Logotipo

Descripción generada automáticamente

**Benemérita Universidad**

**Autónoma de Puebla**

Facultad de Ciencias de la Computación

**MANUAL DE PRACTICAS**

COMPUTO DISTRIBUIDO

Evelyn Dolores Flores Lechuga

Itzel Daniela Martínez Carrera

# Presentación

2023

Logotipo

Descripción generada automáticamente

**BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA**

2023

ÍNDICE GENERAL

[Presentación 0](https://correobuap-my.sharepoint.com/personal/evelyn_floresl_alumno_buap_mx/Documents/Tareas%206to%20semestre/Proyecto%20Final%20Computo%20Distribuido.docx#_Toc151421750)

[1.Práctica: Peer to Peer con Archivos Compartidos y Algoritmo Bully 3](#_Toc151421751)

[1.1 Introducción 3](#_Toc151421752)

[1.1.1 Concepto de Peer to Peer (P2P) 3](#_Toc151421753)

[1.1.2 Concepto de par o nodo 4](#_Toc151421754)

[1.1.3 Concepto de Bully 5](#_Toc151421755)

[1.2 Justificación 5](#_Toc151421756)

[1.3 Objetivo general de la práctica 5](#_Toc151421757)

[1.3.1 Objetivos específicos de la práctica 5](#_Toc151421758)

[1.4 Metodología a estudiar 5](#_Toc151421759)

[1.5 Recursos y materiales del equipo de cómputo (infraestructura) 6](#_Toc151421760)

[1.6 Descripción del desarrollo de la práctica 6](#_Toc151421761)

[1.6.1 Archivo: Peer.java 6](#_Toc151421762)

[1.6.2 Archivo: RedP2P.java 6](#_Toc151421763)

[1.6.3 Archivo: Bully.java 6](#_Toc151421764)

[1.7 Evaluación 6](#_Toc151421765)

[1.7.1 Diagnostico 6](#_Toc151421766)

[1.7.2 Formativas 6](#_Toc151421767)

[1.7.3 Sumativas 6](#_Toc151421768)

[1.8 Resultados 6](#_Toc151421769)

[1.8.1 Evidencias 6](#_Toc151421770)

[1.8.2 Pruebas 6](#_Toc151421771)

[1.9 Conclusiones 6](#_Toc151421772)

[Bibliografía 6](#_Toc151421773)

# **Práctica: Peer to Peer con Archivos Compartidos y Algoritmo Bully**

## **1.1 Introducción**

En el vasto panorama de las redes informáticas, la arquitectura peer-to-peer (P2P) ha emergido como un paradigma poderoso y descentralizado para compartir recursos entre múltiples nodos de una red. En este contexto, el desarrollo de un sistema en Java que facilite la comunicación y el intercambio de archivos entre pares resulta fundamental para comprender los principios de esta arquitectura y la implementación de algoritmos de coordinación como el algoritmo Bully (Abusón).

El objetivo de este proyecto es diseñar y construir una red P2P, donde los nodos individuales actúen como iguales, compartiendo recursos, en este caso, archivos, de manera eficiente y descentralizada. Además, se integrará un coordinador utilizando el algoritmo Bully, una estrategia dinámica que permite la elección y gestión del coordinador dentro de la red.

La red P2P implementada en Java proporcionará a los nodos la capacidad de conectarse entre sí, descubrir recursos disponibles y permitir la descarga de archivos desde otros pares.

El algoritmo Bully, que se implementará como parte integral de este proyecto, desencadenará la elección de un coordinador entre los nodos disponibles. Este algoritmo permite identificar y promover dinámicamente a un nodo como coordinador en caso de que el actual falle o sea inaccesible, asegurando así la continuidad de las operaciones en la red sin interrupciones significativas.

El proyecto se desarrollará utilizando Java, un lenguaje de programación ampliamente utilizado en la industria de la informática. Se emplearán conceptos de programación de sockets y multihilos para establecer conexiones entre los peers y gestionar las solicitudes de descarga de archivos.

Cada peer actuará como tanto un cliente como un servidor, permitiendo la transferencia de archivos entre diferentes peers. Además, se implementará un sistema de chat para permitir a los usuarios comunicarse y coordinar la transferencia de archivos.

### **Concepto de Peer to Peer (P2P)**

Una red P2P (Peer-to-Peer) representa un ecosistema de comunicación donde todos los nodos o usuarios de la red operan en igualdad de condiciones, siendo capaces de asumir tanto funciones de servidor como de cliente de manera simultánea. En contraposición a los sistemas tradicionales de cliente-servidor, en una red P2P no se observa una estructura centralizada; en su lugar, los datos se dispersan descentralizada mente, permitiendo que los usuarios intercambien recursos y servicios entre sí de forma directa y distribuida.

Estas redes P2P presentan una serie de características:

* **Descentralización**: No existe un servidor central que coordine las interacciones entre los usuarios. En cambio, todos los nodos de la red operan como servidores y clientes a la vez, posibilitando que la información y los servicios circulen y sean compartidos de forma descentralizada.
* **Participación** **activa**: Los usuarios deben involucrarse activamente en la red para contribuir con recursos y recibirlos a su vez. En otras palabras, un usuario no solo consume recursos, sino que también actúa como proveedor de estos para otros participantes.
* **Gestión de la conexión**: La gestión de conexiones y la transferencia de archivos se llevan a cabo mediante acuerdos entre los usuarios, sin requerir la intervención de un servidor central que dicte estas interacciones.
* **Compartición de recursos**: Los usuarios tienen la capacidad de compartir una variedad de recursos, desde archivos hasta servicios y conocimientos, con otros miembros de la red.

Las redes P2P tienen una amplia gama de aplicaciones, incluyendo la compartición de archivos, el streaming de música y vídeos, los juegos en línea y servicios de colaboración. Sin embargo, también han sido asociadas con actividades que infringen los derechos de autor, ya que algunas redes P2P posibilitan el intercambio de contenidos sin el consentimiento de sus legítimos propietarios.

### **1.1.2 Concepto de par o nodo**

En informática y en telecomunicaciones, un nodo es un punto de intersección o unión de varios elementos que confluyen en el mismo lugar. En el contexto de las redes de computadoras, cada máquina conectada a la red se considera un nodo. Los nodos pueden ser dispositivos físicos, como computadoras, servidores, impresoras, enrutadores, switches, hubs, teléfonos IP, cámaras de seguridad, entre otros. En estructuras de datos dinámicas, un nodo es un registro que contiene un dato de interés y al menos un puntero para referenciar (apuntar) a otro nodo. Si la estructura tiene solo un puntero, la única estructura que se puede construir con él es una lista, y si el nodo tiene más de un puntero se pueden construir estructuras más complejas como árboles o grafos.

En resumen, un nodo es un punto de intersección o unión de varios elementos que confluyen en el mismo lugar, ya sea en el contexto de las redes de computadoras o en estructuras de datos dinámicas. En el primer caso, cada máquina conectada a la red se considera un nodo, mientras que, en el segundo caso, un nodo es un registro que contiene un dato de interés y al menos un puntero para referenciar a otro nodo. En el contexto de las redes de computadoras, un nodo se refiere a un dispositivo o punto de conexión que forma parte de una red y que puede enviar, recibir o retransmitir datos.

* Cada nodo tiene una dirección única en la red, que se utiliza para identificarlo y comunicarse con él.
* Los nodos son esenciales para el funcionamiento de una red, ya que permiten la comunicación entre los diferentes dispositivos que forman parte de ella.
* Cada nodo puede actuar como un punto de origen o destino de los datos, y también puede retransmitir los datos a otros nodos en la red
* Los nodos permiten que los dispositivos se comuniquen entre sí y compartan recursos, como archivos, impresoras y conexiones a Internet.

### **1.1.3 Concepto de Bully**

El algoritmo Bully es un algoritmo de elección utilizado en sistemas distribuidos para seleccionar un coordinador o líder dinámicamente a partir de un grupo de procesos informáticos distribuidos. Este algoritmo es especialmente útil en entornos donde los procesos deben ser capaces de elegir un coordinador en caso de que el coordinador actual falle o se vuelva inaccesible. El algoritmo Bully asume que cada proceso tiene un número de prioridad único en el sistema, y el proceso con la prioridad más alta será elegido primero como nuevo coordinador. Cuando el proceso coordinador actual falla, el algoritmo inicia una nueva elección para seleccionar un nuevo coordinador entre los procesos restantes. El proceso de elección en el algoritmo Bully sigue los siguientes pasos:

* Un proceso que detecta la ausencia del coordinador actual inicia una elección enviando un mensaje a los procesos con una prioridad más baja que la suya.
* Los procesos con prioridades más bajas responden al mensaje, renunciando a su candidatura y reconociendo al proceso que inició la elección como el nuevo coordinador.
* Si no hay respuesta de los procesos con prioridades más bajas, el proceso que inició la elección se convierte en el nuevo coordinador.

El algoritmo Bully es un método para elegir dinámicamente un coordinador en un sistema distribuido, asumiendo que cada proceso tiene un número de prioridad único y que el proceso con la prioridad más alta será elegido como el nuevo coordinador en caso de que el coordinador actual falle.

## **1.2 Justificación**

El proyecto de implementar una red peer-to-peer que comparte archivos, junto con el uso del algoritmo Bully para designar un coordinador, ofrece la oportunidad de explorar tecnologías emergentes y relevantes en el ámbito de las redes distribuidas, permitiendo una comprensión profunda de su aplicación práctica. Además, implica un aprendizaje detallado sobre conceptos fundamentales de redes, protocolos de comunicación y gestión de conexiones, ofreciendo una experiencia valiosa en estas áreas.

Por último, este proyecto no solo contribuye a la formación académica personal, sino que también puede ser un recurso educativo valioso para otros interesados en aprender sobre redes distribuidas y algoritmos de coordinación. En resumen, esta iniciativa ofrece una oportunidad única para adquirir conocimientos sólidos, desarrollar habilidades prácticas y contribuir al entendimiento y aplicación de tecnologías innovadoras en el campo de las redes y la computación distribuida.

## **1.3 Objetivo general de la práctica**

El objetivo general de este proyecto es implementar una red peer-to-peer en Java que permita el intercambio de archivos entre nodos, mientras se incorpora el algoritmo Bully para la selección y gestión dinámica de un coordinador en el entorno distribuido de la red.

### **1.3.1 Objetivos específicos de la práctica**

* Desarrollar un sistema funcional en Java que permita la comunicación entre nodos para compartir archivos de manera descentralizada.
* Definir un protocolo de comunicación eficiente entre los nodos de la red para la transferencia de archivos y la gestión de conexiones.
* Establecer un mecanismo para la gestión y almacenamiento de archivos compartidos, incluyendo la capacidad de búsqueda y descarga por parte de otros nodos.
* Integrar el algoritmo Bully en la red para la elección y gestión dinámica de un coordinador, asegurando su funcionamiento ante posibles fallos o cambios en la red.

## **1.4 Metodología a estudiar**

Los pasos que se realizaron para llevar a cabo el correcto desarrollo de la red peer to peer:

1. Obtener el conocimiento básico sobre una red distribuida.
2. Instalar Apache NetBeans IDE 17 con su respectivo jdk.
3. Crear la interfaz gráfica del programa para que el usuario pueda interactuar con él (archivo Peer.java).
4. Crear la lógica del funcionamiento de la red peer to peer (archivo RedP2P.java).
5. Crear y enlazar la lógica del algoritmo bully a la red peer to peer (archivo Bully.java).
6. Configurar la red en la que se conectarían los dispositivos que van a ejecutar el programa.
7. Conectar los dispositivos a la red local creada (modem) y comenzar con las pruebas.
8. Documentar los resultados.

## **1.5 Recursos y materiales del equipo de cómputo (infraestructura)**

* Sistema Operativo Windows 11 Pro
* Lenguaje de programación Java
* Entorno de desarrollo NetBeans IDE 17 con su respectivo jdk.
* Computadora HP con procesador AMD Ryzen 5 5600U with Radeon Graphics, 2301 Mhz
* Computadora HP con procesador AMD Ryzen 3 3300U with Radeon Vega Mobile Gfx, 2.10 GHz

## **1.6 Descripción del desarrollo de la práctica**

El desarrollo de esta práctica implica una serie de etapas clave que van desde la planificación hasta la implementación y pruebas del sistema de red peer-to-peer con el algoritmo Bully.

1. Planificación y Diseño:

Análisis de Requisitos: Definición clara de las funcionalidades requeridas para la red P2P y el comportamiento del algoritmo Bully en el entorno distribuido.

Diseño de la Arquitectura: Establecimiento de la estructura de la red, definición de los protocolos de comunicación y planificación de la integración del algoritmo Bully.

1. Implementación de la Red P2P:

Desarrollo de la Comunicación entre Nodos: Codificación de los métodos y procedimientos que permitan a los nodos de la red intercambiar información, compartir archivos y gestionar conexiones.

Gestión de Archivos Compartidos: Creación de un sistema para almacenar y compartir archivos entre los nodos, incluyendo funciones de búsqueda y descarga.

1. Integración del Algoritmo Bully:

Implementación del Algoritmo: Codificación del algoritmo Bully para la elección y gestión del coordinador en el contexto distribuido de la red P2P.

Manejo de Fallos y Cambios: Incorporación de mecanismos para manejar situaciones de fallo del coordinador actual y la elección de un nuevo líder en caso de necesidad.

1. Pruebas y Depuración:

Pruebas Unitarias: Verificación del funcionamiento individual de cada componente de la red y del algoritmo.

Pruebas de Integración: Evaluación de la interacción entre los diferentes módulos y la coherencia del sistema en su conjunto.

Depuración y Optimización: Identificación y corrección de errores, así como la optimización de rendimiento del sistema.

1. Documentación y Presentación:

Elaboración de Documentación: Creación de documentación detallada que incluya el diseño, implementación, pruebas y funcionamiento del sistema.

Presentación del Proyecto: Preparación para compartir los hallazgos, desafíos y soluciones encontradas durante el desarrollo del proyecto.

Esta descripción resalta los pasos principales a seguir, desde la concepción y diseño hasta la implementación, pruebas y documentación del sistema de red P2P con el algoritmo Bully, asegurando así un enfoque estructurado y completo en el desarrollo del proyecto para así lograr representar una red peer to peer como se muestra en la Fig 1.1.

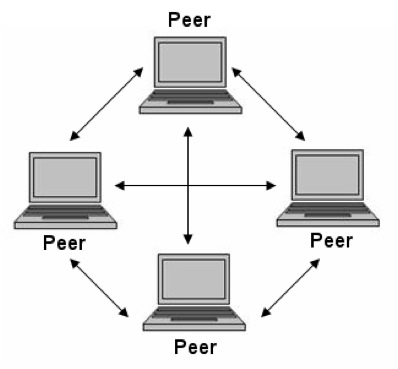


Fig 1.1 Representación gráfica de una Red peer to peer.

### **1.6.1 Archivo: Peer.java**

El archivo Peer.java representa la interfaz de una aplicación en Java basada en Swing que funciona como un nodo para cada peer en una red peer-to-peer (P2P).

Se cuenta con un constructor Peer que inicializa los componentes de la interfaz gráfica mediante el método initComponents().Al inicio de la ejecución, el programa solicita al usuario su nombre o le asigna uno anónimo con la hora en que inicio, que será tomada como su identificador y con ello crea una instancia de la red P2P (peer) con una dirección IP multicast, un puerto y el nombre del nodo. De tal forma que asocia áreas de texto de la GUI con instancias específicas de la red P2P e inicia un nuevo hilo para la ejecución de la red P2P.

Él envió de mensajes se hace con el jButton2ActionPerformed que se activa cuando el usuario hace clic en el botón "Send" que sirve para enviar el mensaje a la red, ese mensaje se obtiene el mensaje del campo de texto jTextField4 y utiliza la instancia de la red P2P (peer) para enviar el mensaje a los otros nodos de la red.

Para la carga de archivos en el jButtonUploadActionPerformed se activa cuando el usuario hace clic en el botón "Cargar Archivo", ahí mismo llama al método runUpload de la instancia de la red P2P (peer), que permite al usuario cargar un documento de su sistema de archivos.

Por último, el método main configura el aspecto visual de la aplicación e inicializa una instancia de la clase Peer, lo que hace que la interfaz gráfica sea visible al usuario. El contenido de este archivo se muestra en la Fig 1.2.

Texto

Descripción generada automáticamente

Fig 1.2 Contenido del archivo Peer.java

### **Archivo: RedP2P.java**

### Este archivo RedP2P.java representa la lógica de red para una aplicación peer-to-peer (P2P). El constructor RedP2P inicializa el socket multicast, la dirección IP, el puerto y el nombre del nodo además con la clase interna FileData representa la información de un archivo compartido, incluyendo el nombre del archivo y los datos del archivo en formato de bytes.

Los métodos set\_Area, set\_AreaPeers, set\_AreaArchivos, se encargan de asociar las áreas de texto con instancias específicas de la red P2P.

Los métodos que se utilizaron para manejar la lógica del proyecto:

* enviar\_Mensaje, este envía mensajes al chat mediante un hilo separado, incluyendo el nombre del remitente y el contenido del mensaje.
* runUpload, inicia un hilo para manejar la carga de archivos mediante un cuadro de diálogo de selección de archivos.
* enviarArchivo, envía un archivo a la red P2P mediante un hilo separado.
* guardarArchivoEnCarpeta, se encarga de guarda el archivo cargado en una carpeta local llamada "Upload”.
* solicitarArchivo, envía una solicitud para descargar el archivo.
* descargarArchivo, descarga y guarda un archivo en una carpeta local llamada "Download".
* run, hace la ejecución del hilo para la comunicación de la red P2P con esto maneja la recepción de mensajes, solicitudes de archivos, archivos compartidos y actualiza las áreas de texto de la interfaz gráfica según sea necesario.

El contenido de este archivo se muestra en la Fig 1.3.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Fig 1.3 Contenido del archivo RedP2P.java

### **1.6.3 Archivo: Bully.java**

El archivo Bully.java implementa un sistema distribuido que utiliza un enfoque al algoritmo de Bully para elegir un coordinador entre los nodos. Cada instancia de la clase Peer representa un nodo en la red y se inicializa con un identificador único (hora en que inicio) y un indicador de coordinador. Utiliza un socket multicast para la comunicación entre nodos y se une a un grupo multicast con una dirección y un puerto.

El método msg\_Coordinador, envía un mensaje a todos los nodos informando que este nodo es el coordinador.

El método msg\_Eleccion, envía un mensaje indicando la intención de este nodo de ser coordinador.

El método run, implementa el hilo principal del nodo, donde el nodo coordinador envía periódicamente mensajes, y los nodos no coordinadores esperan mensajes del coordinador actual o participan en el proceso de elección según el algoritmo de Bully.

El método to\_Split\_Datagram, se encarga de dividir la información de los datagramas para su procesamiento. El contenido de este archivo se muestra en la Fig 1.4.

Texto

Descripción generada automáticamente

Fig 1.4 Contenido del archivo Bully.java

## **1.7 Evaluación**

### **1.7.1 Diagnostico**

Previo a esta práctica se desarrolló un programa en Java que realizaba funciones similares al de chat de mensajes. Este programa solicita un nombre de usuario y le asigna un ID aleatorio, después muestra el área de chat, donde el usuario puede introducir un mensaje y enviarlo a la red (en un hilo) en espera de que otro peer conectado responda. En la Fig 1.5 se aprecia el programa antes descrito.

Interfaz de usuario gráfica

Descripción generada automáticamente

Fig 1.5 Programa de chat sencillo Peer Multicast.

Evidentemente el programa anterior no implementa de manera completa todo el concepto de una red distribuida. Sin embargo, fue tomado como base para mejorar y complementar en gran medida todo lo que abarca el concepto de una red peer to peer, agregando funcionalidades extras como la lista de los peers conectados a la red, la distribución de archivos y la elección de un coordinado.

### **1.7.2 Formativas**

Como estudiantes de la carrera de Ingeniería en Tecnologías de la Información, familiarizados con el contexto cliente-servidor, el desarrollo de este proyecto representó un auténtico desafío. Nos vimos en la necesidad de dejar atrás la noción convencional de un servidor para adoptar el concepto “distribuida”, comprendiendo que la información no reside en un único punto, sino que fluye a través de una red interconectada. La creación de una red peer-to-peer, la transferencia de archivos en formato PDF y la designación de un coordinador resultó ser un proceso fascinante. Este enfoque no solo amplió nuestros conocimientos, sino que también nos proporcionó una visión única sobre la organización y la comunicación en entornos de red distribuidos.

### **1.7.3 Sumativas**

Para llevar a cabo con éxito el proyecto, fue necesario implementar la siguiente lógica:

* Él envió de información siempre se realiza mediante un socket en la red multicast.
* Los hilos nos permiten realizar tareas en paralelo y facilitan la comunicación entre procesos.
* Es esencial identificar el tipo de información que se transmite en la red para después poder clasificarla. Implementamos un prefijo para cada tipo de mensaje:
  + "[FILE]:", mensajes de tipo archivos.
  + "[REQUEST\_FILE]:", archivos que eran solicitados para ser descargados.
  + "[FILE\_SHARE]:", archivos que eran compartidos en la red.
  + "[BYE]:", mensajes que indican una desconexión.
  + "[HoraPeer]:", señalaría una conexión y proporcionaría hora y nombre del peer que está conectado a la red.
  + “nombre”, mensajes normales del chat.

Gracias a esta estrategia, pudimos identificar y procesar la información para después determinar en qué sección de la interfaz debía aparecer, según el tipo de mensaje que fuera.

* Para mantener actualizada la lista de los pares conectados:
  + Implementamos un hilo que enviaba los datos del peer (nombre y hora de inicio) cada segundo.
  + Estos datos se almacenaban en un HashMap (tiemposConectados).
  + Cuando un peer se desconectaba, enviaba un mensaje de despedida ([BYE]:) a la red multicast, indicando la desconexión.
  + Eliminábamos la información correspondiente a ese peer del HashMap.
  + Finalmente, actualizábamos la información en la interfaz para reflejar los cambios.
* Para realizar exitosamente el proceso de compartir archivos en la red fue necesario seguir con los siguientes pasos:
  + Un peer carga un archivo de formato PDF y lo envía a la red, este archivo se guarda en su carpeta local Upload.
  + Los demás peers pueden visualizar el nombre del archivo en el área de recursos compartidos.
  + Si un peer desea descargar el archivo, da click en el nombre del recurso compartido y esto enviara un mensaje de solicitud al peer que subió el archivo.
  + El peer que recibe la solicitud del archivo, envía el archivo al peer que lo solicito.
  + El peer solicitante recibe el archivo y lo guarda en su carpeta local Download.
* Para realizar el proceso de elección del coordinador:
* Cuando el programa se ejecuta automáticamente se inicia el proceso de elección.
* El algoritmo bully tomara la hora de inicio del peer como un identificador y tomara como coordinador aquel que lleve más tiempo conectado a la red.
* La variable esCoordinador será true cuando se haya identificado la hora de inicio más temprana.
* Cuando sea nombrado un coordinador este enviara mensajes a la red multicast para informar sobre su nombramiento, los peers que estén conectados escuchan el mensaje y reconocen al coordinador.
* Si el coordinador se desconecta el proceso de elección debe reiniciarse en busca de un nuevo coordinador.

## **1.8 Resultados**

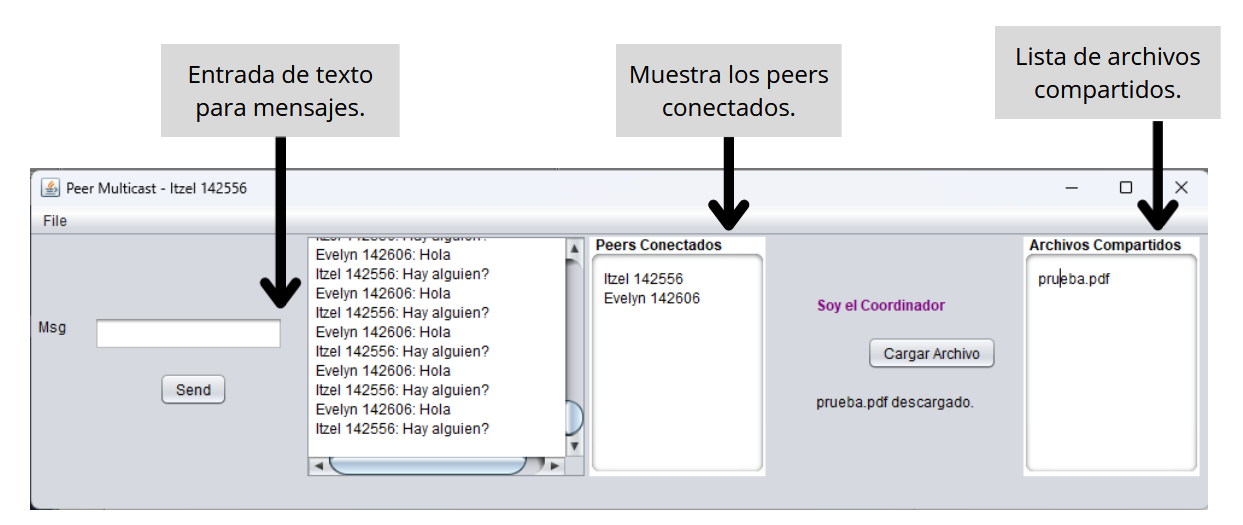
Cuando el programa se ejecuta por primera vez, es necesario ingresar un nombre de usuario, en caso de dejar vacío el programa lo registrará como anónimo, este nombre de usuario será concatenado con la hora en que inicio que a la vez será su ID para identificarlo en la red. Como se muestra en la Fig 1.6.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Fig 1.6 Cuadro de dialogo para ingresar el usuario.

Una vez ingresado el nombre de usuario se mostrará la interfaz gráfica principal, que contiene tres secciones: área de chat para enviar mensajes, el área donde se muestran los peers conectados y el área de recursos compartidos. En la interfaz también se muestran mensajes de texto para indicar quien es el coordinador y los archivos que se han descargado. Lo anterior descrito se muestra en la Fig 1.7.



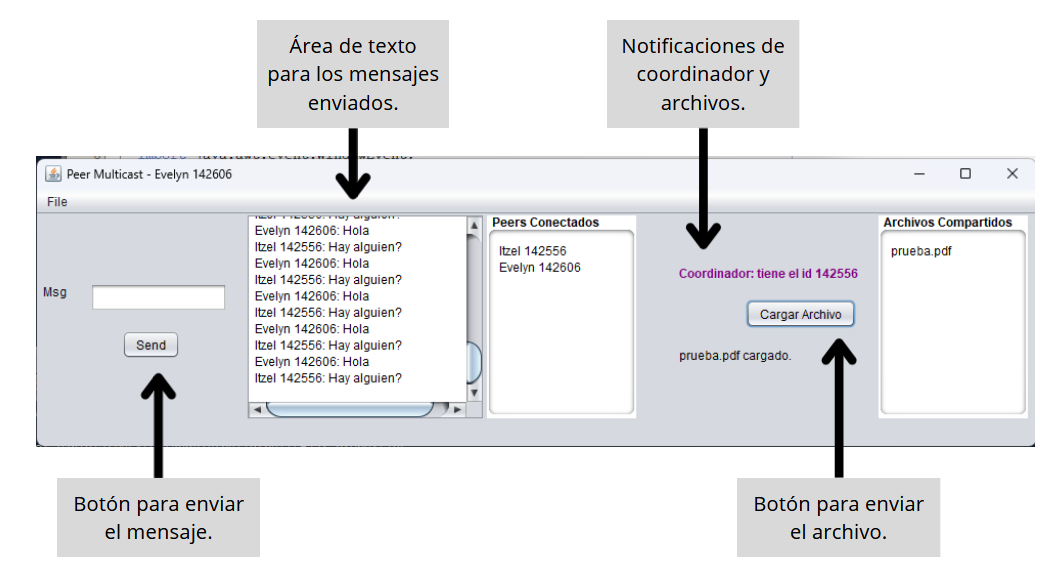


Fig 1.7 Elementos de la interfaz gráfica de la Red peer to peer.

### **1.8.1 Evidencias**

Como evidencias, se tomarán las ejecuciones del programa realizadas durante la clase asegurándonos de que cumplan con todas las especificaciones requeridas, y contaremos con la supervisión directa del docente como testigo confiable de su correcto funcionamiento. Además, este manual y los códigos implementados se considerarán igualmente como evidencias de la implementación del proyecto.

### **1.8.2 Pruebas**

Inicialmente, se llevó a cabo una prueba exhaustiva en un solo ordenador, donde el sistema demostró un rendimiento satisfactorio y cumplió con todas las expectativas previstas.

Posteriormente, se amplió la prueba al conectar dos PC con capacidades distintas, ambas vinculadas a un módem y configuradas en la misma red privada. Estos escenarios más complejos también arrojaron resultados satisfactorios, evidenciando la capacidad del programa para operar de manera efectiva en entornos de red más amplios y heterogéneos. Los resultados exitosos de esta última prueba se presentan en la Fig 1.8 y la Fig 1.9, para proporcionar una representación visual de la estabilidad y la eficiencia del programa.

Captura de pantalla de computadora

Descripción generada automáticamente

Fig 1.8 Captura de pantalla PC1

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Fig 1.9 Captura de pantalla PC2

## **1.9 Conclusiones**

Al concluir el desarrollo de esta práctica se logro identificar la gran diferencia que existe entre un sistema centralizado y uno descentralizado identificando también las ventajas y desventajas de cada uno de ellos. Nos enfrentamos a desafíos al implementar el algoritmo Bully, que demanda una comunicación constante para su correcto funcionamiento. La implementación de la transferencia de archivos implicó la gestión de diversos métodos cuya interacción podía resultar confusa. En consecuencia, la clave residía en contar con pasos claros y completos para llevar a cabo esta actividad de manera eficiente. Este proyecto no solo consolidó nuestra comprensión de los sistemas distribuidos, sino que también resaltó la importancia de una planificación meticulosa y una ejecución precisa en el desarrollo de soluciones descentralizadas.

# **Bibliografía**

1. ¿Qué es P2P (peer to peer)? (2023, mayo 12). IONOS Digital Guide; IONOS. https://www.ionos.es/digitalguide/servidores/know-how/que-es-p2p-peer-to-peer/
2. Dos Santos, M. (2023, junio 23). Nodo de red: qué es, para qué sirve y cómo funciona. tfmfwH84@NjhsqeEJ834Muyh. https://polaridad.es/nodo-de-red-que-es-para-que-sirve-y-como-funciona/
3. de Asís López Fuentes, F. (2015). Sistemas distribuidos. UNIVERSIDAD AUTÓNOMA METROPOLITANA. http://dccd.cua.uam.mx/libros/archivos/03IXStream\_sistemas\_distribuidos.pdf