



INSTITUTO POLITECNICO NACIONAL “IPN”



ESCUELA SUPERIOR DE COMPUTO “ESCOM”

LANDEROS CORTES PEDRO JONAS

ANAYA PEREZ MARCO ANTONIO

APLICACIONES PARA COMUNICACIONES EN RED

ULTIMO RETO

Indice

Introducción	4
Desarrollo.....	4
1. Descripción general del programa	4
2. Librerías utilizadas y su propósito.....	5
3. Estructuras de datos principales	6
3.1. Peer	6
3.2. SharedFile	6
3.3. SeenId	6
4. Sincronización y concurrencia	7
5. Protocolo de aplicación.....	7
5.1. Handshake / Autenticación.....	7
5.2. Mensajes cifrados (post-auth).....	8
6. Funciones importantes del código (explicación por módulos)	8
6.1. Utilidades y hashing	8
6.2. Manejo de peers	9
6.3. Compartición y transferencia de archivos	9
6.4. Búsqueda distribuida (flooding TTL).....	10
6.5. Bucle de red	10
7. Pruebas funcionales (comandos y evidencias)	10
7.1. Prueba 1: Conexión y autenticación (handshake)	10
7.2. Prueba 2: Broadcast.....	12
7.3. Prueba 3: Mensaje directo (/msg)	12
7.4. Prueba 4: Compartir archivo y listar.....	14
7.5. Prueba 5: Búsqueda distribuida (/search)	14
7.6. Prueba 6: Descarga de archivo (/get).....	15
7.7. Prueba 7: Desconexión controlada.....	15
7.8. Prueba 8: Seguridad (PSK incorrecta)	16
Conclusiones	16

Flujo completo.....	18
Consola A	18
Consola B	19
Referencias	20

Introducción

En esta práctica se desarrolló e implementó una mini-plataforma P2P (peer-to-peer) en lenguaje C para el sistema operativo Windows, utilizando la librería WinSock2 para la comunicación de red. El objetivo principal fue construir un programa capaz de establecer conexiones entre nodos sin depender de un servidor central, permitiendo el intercambio de mensajes (chat directo y broadcast), búsqueda distribuida de recursos (flooding con TTL), transferencia de archivos mediante solicitudes, un mecanismo básico de autenticación con llave compartida (PSK), cifrado demostrativo con XOR, y un envío/recepción de datos por UDP simulando un “stream” de voz.

El diseño P2P se caracteriza porque cada nodo puede actuar simultáneamente como cliente y como servidor: un nodo puede escuchar conexiones entrantes (servidor) mientras se conecta activamente a otros nodos (cliente). Esto representa un escenario realista de redes distribuidas donde no existe un único punto de falla y donde el intercambio de información se hace directamente entre participantes.

Además, se integró un protocolo de aplicación sencillo basado en mensajes de texto delimitados por saltos de línea, usando comandos con prefijo “/” para facilitar la interacción desde consola. Finalmente, se realizaron pruebas funcionales para verificar el comportamiento correcto de cada módulo: conexión y autenticación, mensajería, broadcast, compartición y búsqueda de archivos, descarga, manejo de desconexiones, envío UDP y validación de seguridad por PSK.

Desarrollo

1. Descripción general del programa

El programa implementa un nodo P2P ejecutable desde consola con la siguiente forma:

```
p2p.exe <listenPort> <username> <psk>
```

- **listenPort**: puerto TCP donde el nodo escuchará conexiones entrantes.
- **username**: nombre del usuario/nodo (se usa en los mensajes).
- **psk**: “Pre-Shared Key”, una clave compartida que todos los nodos deben usar para autenticarse.

Al iniciar, el nodo:

1. Inicializa WinSock2 (WSAStartup).
2. Crea un socket TCP de escucha (bind + listen).
3. Inicia un hilo para leer comandos de consola.
4. En el ciclo principal, usa select() para manejar simultáneamente:
 - o Nuevas conexiones entrantes.
 - o Datos recibidos de peers existentes.
 - o Comandos escritos por el usuario.

2. Librerías utilizadas y su propósito

En el código se incluyen las siguientes librerías:

- `#include <winsock2.h>`
Librería principal de sockets en Windows. Provee socket, bind, listen, accept, connect, send, recv, select, etc.
- `#include <ws2tcpip.h>`
Extiende WinSock con utilidades modernas como inet_pton, inet_ntop para conversión de IP y soporte de direcciones.
- `#include <windows.h>`
Necesaria para funciones del sistema y sincronización: CRITICAL_SECTION, InitializeCriticalSection, EnterCriticalSection, LeaveCriticalSection, Sleep, GetTickCount.
- `#include <process.h>`
Permite crear hilos en C de Windows mediante _beginthreadex, recomendado para evitar problemas con el runtime de C.
- `#include <stdint.h>`
Tipos enteros de tamaño fijo (uint32_t, uint64_t) para evitar ambigüedades.
- `#include <stdio.h>`
Entrada/salida estándar (printf, fgets, fopen, etc.).
- `#include <stdlib.h>`
Memoria dinámica y utilidades (calloc, free, atoi, etc.).
- `#include <string.h>`
Manejo de strings (strncpy, strcmp, strtok_s, strlen, etc.).

Además:

- `#pragma comment(lib, "ws2_32.lib")`
Indica al compilador/linker de MSVC que linkee ws2_32. En GCC generalmente se usa -lws2_32 al compilar.

3. Estructuras de datos principales

3.1. Peer

Representa un nodo remoto conectado:

- SOCKET sock: socket TCP asociado.
- sockaddr_in addr: dirección IP/puerto remoto.
- active: indica si el slot está en uso.
- authed: si ya está autenticado.
- session_key: llave derivada para cifrado XOR.
- user: nombre del usuario del peer.
- rbuf / rbuf_len: buffer acumulador para “framing” por líneas.

El buffer es importante porque recv() puede devolver:

- fragmentos de líneas
- múltiples líneas en una sola llamada
Entonces el programa va acumulando hasta encontrar \n y luego procesa línea por línea.

3.2. SharedFile

Lista de archivos compartidos localmente:

- path: ruta completa.
- name: nombre del archivo (basename).
- size: tamaño en bytes.

3.3. SeenId

IDs de búsqueda ya vistos para evitar loops en flooding:

- idhex: identificador de 8 hex.

4. Sincronización y concurrencia

El programa usa:

- CRITICAL_SECTION g_lock
Protege estructuras globales como:

- g_peers[]
- cola de comandos
- listas de “seen ids”
- etc.

Esto es necesario porque:

- El hilo de consola inserta comandos en una cola.
- El hilo principal consume comandos y maneja sockets.
Sin sincronización habría condiciones de carrera.

5. Protocolo de aplicación

El protocolo está basado en mensajes de texto:

5.1. Handshake / Autenticación

- Cliente envía (en plano):
- AUTH|noncehex8|hashhex8|username

Donde:

- noncehex8 es un valor pseudoaleatorio de 32 bits en hex.
 - hashhex8 = FNV(noncehex + psk).
 - username es el nombre del cliente.
- Servidor valida el hash. Si falla:
 - AUTHFAIL

y cierra conexión.

- Si pasa, servidor responde (en plano):
- AUTHOK|serverUser

y después marca authed=1 en ambos lados.

Importancia del cambio aplicado:

Durante las pruebas se detectó que si AUTHOK se enviaba cifrado antes de que el cliente estuviera autenticado, el cliente lo ignoraba (porque todavía no aceptaba ENC). Por eso, la respuesta AUTHOK debe viajar en plano para completar el handshake.

5.2. Mensajes cifrados (post-auth)

Una vez autenticado, se usa formato:

ENC|<hex>

donde <hex> es el mensaje original cifrado con XOR + keystream (LCG) usando session_key.

Dentro del mensaje descifrado, se procesan comandos de aplicación como:

- CHAT|from|text
- BCAST|from|text
- SRCH|...
- SRCHRESP|...
- GET|filename
- FILE|filename|size

6. Funciones importantes del código (explicación por módulos)

6.1. Utilidades y hashing

- fnv1a_32()
Implementa hash FNV-1a para generar IDs y validar autenticación.
- u32_to_hex8()
Convierte un entero de 32 bits a hex de 8 caracteres.
- hex_to_bytes() y bytes_to_hex()
Para representar bytes como hex en mensajes ENCI.
- xor_crypt()
Cifrado demostrativo. Genera un “keystream” con LCG y aplica XOR.
No es criptografía fuerte, pero sirve para mostrar el concepto de cifrado simétrico con llave.

- `strcasestr_simple()`
Implementa búsqueda substring case-insensitive para que /search funcione sin depender de funciones específicas de Windows.

6.2. Manejo de peers

- `add_peer()`
Guarda un socket nuevo en el arreglo `g_peers`.
- `remove_peer()`
Cierra socket y libera el slot.
- `peer_send_line()`
Envía mensajes. Si `authed=1`, manda ENC|.... Si no, manda plano.
- `peer_send_plain_line()`
Fuerza envío sin cifrado, usado en handshake.
- `peer_send_line_all()`
Envía a todos los peers autenticados.

6.3. Compartición y transferencia de archivos

- `share_add()`
Agrega archivo a la lista de compartidos, obteniendo tamaño y nombre.
- `share_list()`
Imprime lista de compartidos.
- `share_find()`
Busca por nombre para atender solicitudes GET.
- `handle_app_message()` caso GET|:
 1. Valida si existe el archivo.
 2. Envía header FILE|name|size cifrado.
 3. Luego manda bytes crudos (no cifrados) para simplificar.
- `handle_file_incoming()`
Al recibir FILE|name|size, guarda exactamente size bytes siguientes en un archivo local.

6.4. Búsqueda distribuida (flooding TTL)

- cmd_search()
Genera un ID de búsqueda y envía:
- SRCH|id|ttl|origin_ip|origin_port|pattern

a todos los peers.

- handle_app_message() caso SRCH|:
 1. Si ID ya fue visto, ignora.
 2. Si no, guarda en seen.
 3. Busca coincidencias en archivos compartidos con substring case-insensitive.
 4. Responde con SRCHRESP|...
 5. Si ttl > 0, reenvía a otros peers (menos al emisor).

Esto evita loops infinitos con:

- lista seen
- decremento de TTL.

6.5. Bucle de red

- process_peer_data()
Lee datos con recv(), acumula en buffer, separa por líneas y procesa:
 - control pre-auth (AUTH, AUTHOK, AUTHFAIL)
 - app post-auth (CHAT, BCAST, etc.)
- Uso de select() en el main loop:
Permite no bloquearse y manejar múltiples sockets en un solo hilo.

7. Pruebas funcionales (comandos y evidencias)

A continuación, se describen pruebas recomendadas. **Aquí es donde yo agrego capturas (screenshots) de las consolas.**

Importante: Para la prueba se abren dos consolas (A y B) con puertos distintos.

7.1. Prueba 1: Conexión y autenticación (handshake)

Consola A

p2p.exe 9001 usuario1 123

Consola B

p2p.exe 9002 usuario2 123

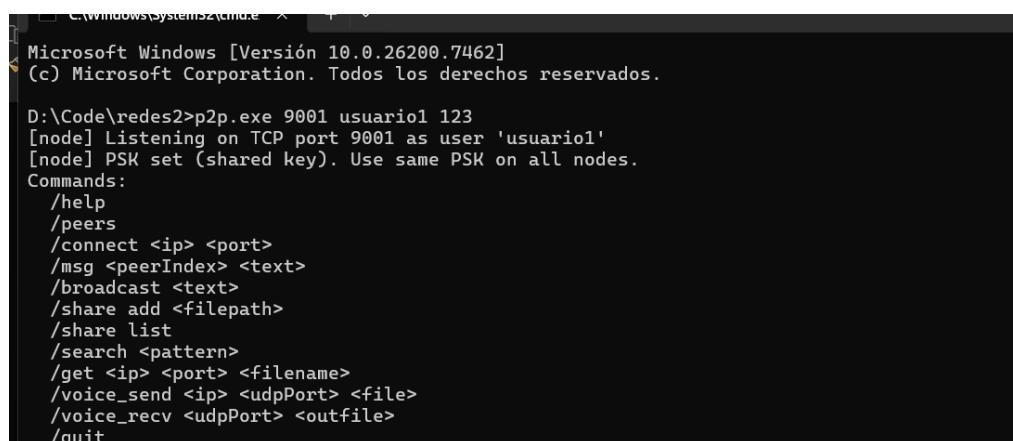
/connect 127.0.0.1 9001

/peers

Resultado esperado:

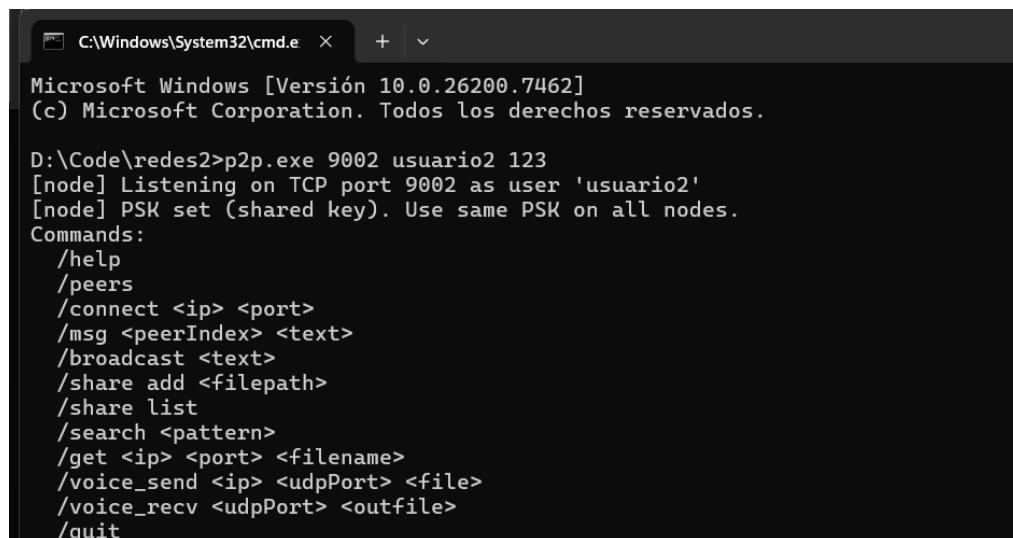
En B debe aparecer:

authed=1 user=usuario1



```
C:\Windows\System32\cmd.e × + ▾
Microsoft Windows [Versión 10.0.26200.7462]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\Code\redes2>p2p.exe 9001 usuario1 123
[node] Listening on TCP port 9001 as user 'usuario1'
[node] PSK set (shared key). Use same PSK on all nodes.
Commands:
/help
/peers
/connect <ip> <port>
/msg <peerIndex> <text>
/broadcast <text>
/share add <filepath>
/share list
/search <pattern>
/get <ip> <port> <filename>
/voice_send <ip> <udpPort> <file>
/voice_recv <udpPort> <outfile>
/quit
```



```
C:\Windows\System32\cmd.e × + ▾
Microsoft Windows [Versión 10.0.26200.7462]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\Code\redes2>p2p.exe 9002 usuario2 123
[node] Listening on TCP port 9002 as user 'usuario2'
[node] PSK set (shared key). Use same PSK on all nodes.
Commands:
/help
/peers
/connect <ip> <port>
/msg <peerIndex> <text>
/broadcast <text>
/share add <filepath>
/share list
/search <pattern>
/get <ip> <port> <filename>
/voice_send <ip> <udpPort> <file>
/voice_recv <udpPort> <outfile>
/quit
```

7.2. Prueba 2: Broadcast

En **Consola B:**

```
/broadcast hola
```

Resultado esperado:

En A:

```
[bcast] usuario2: hola
```

```
Microsoft Windows [Versión 10.0.26200.7462]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\Code\redes2>p2p.exe 9002 usuario2 123
[node] Listening on TCP port 9002 as user 'usuario2'
[node] PSK set (shared key). Use same PSK on all nodes.
Commands:
/help
/peers
/connect <ip> <port>
/msg <peerIndex> <text>
/broadcast <text>
/share add <filepath>
/share list
/search <pattern>
/get <ip> <port> <filename>
/voice_send <ip> <udpPort> <file>
/voice_recv <udpPort> <outfile>
/quit
/connect 127.0.0.1 9001
[connect] Connected peer idx=0 127.0.0.1:9001 (sent AUTH)
[auth] Auth OK with peer 0 (server user: usuario1).
/peers
Peers:
[0] 127.0.0.1:9001  authed=1 user=usuario1
/broadcast hola
```

```
C:\Windows\System32\cmd.e + v

Microsoft Windows [Versión 10.0.26200.7462]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\Code\redes2>p2p.exe 9001 usuario1 123
[node] Listening on TCP port 9001 as user 'usuario1'
[node] PSK set (shared key). Use same PSK on all nodes.
Commands:
/help
/peers
/connect <ip> <port>
/msg <peerIndex> <text>
/broadcast <text>
/share add <filepath>
/share list
/search <pattern>
/get <ip> <port> <filename>
/voice_send <ip> <udpPort> <file>
/voice_recv <udpPort> <outfile>
/quit
[net] accepted peer idx=0 from 127.0.0.1:51832 (waiting AUTH)
[auth] Peer 0 authed as 'usuario2'.
[bcast] usuario2: hola
```

7.3. Prueba 3: Mensaje directo (/msg)

En **Consola B:**

/msg 0 hola usuario1, mensaje directo

Esperado en A:

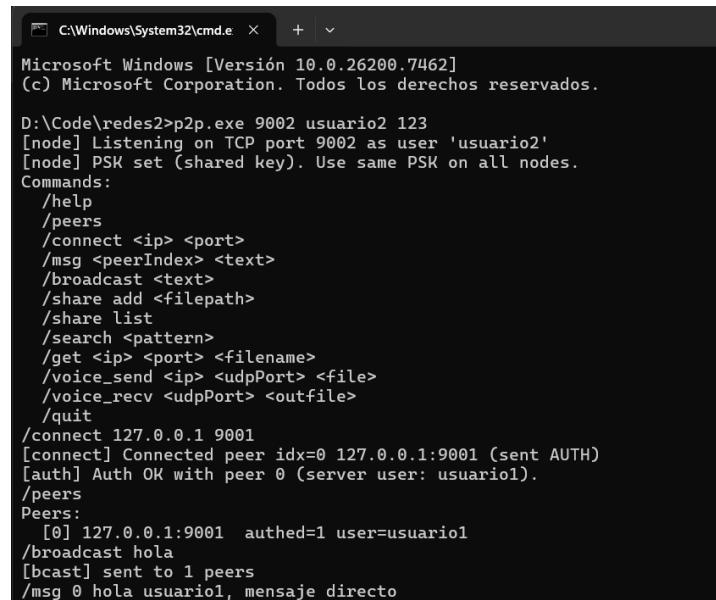
[chat] usuario2: hola usuario1, mensaje directo

Luego en **Consola A**:

/msg 0 recibido

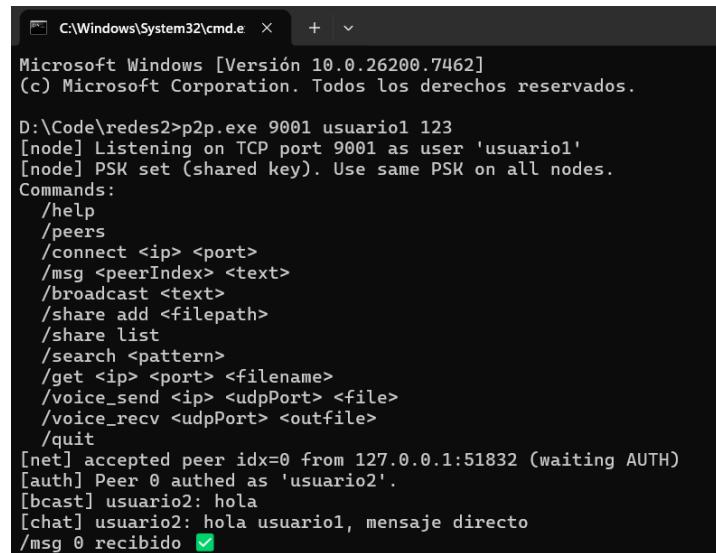
Esperado en B:

[chat] usuario1: recibido ✓



```
C:\Windows\System32\cmd.e × + ▾
Microsoft Windows [Versión 10.0.26200.7462]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\Code\redes2>p2p.exe 9002 usuario2 123
[node] Listening on TCP port 9002 as user 'usuario2'
[node] PSK set (shared key). Use same PSK on all nodes.
Commands:
/help
/peers
/connect <ip> <port>
/msg <peerIndex> <text>
/broadcast <text>
/share add <filepath>
/share list
/search <pattern>
/get <ip> <port> <filename>
/voice_send <ip> <udpPort> <file>
/voice_recv <udpPort> <outfile>
/quit
/connect 127.0.0.1 9001
[connect] Connected peer idx=0 127.0.0.1:9001 (sent AUTH)
[auth] Auth OK with peer 0 (server user: usuario1).
/peers
Peers:
[0] 127.0.0.1:9001 authed=1 user=usuario1
/broadcast hola
[bcast] sent to 1 peers
/msg 0 hola usuario1, mensaje directo
```



```
C:\Windows\System32\cmd.e × + ▾
Microsoft Windows [Versión 10.0.26200.7462]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\Code\redes2>p2p.exe 9001 usuario1 123
[node] Listening on TCP port 9001 as user 'usuario1'
[node] PSK set (shared key). Use same PSK on all nodes.
Commands:
/help
/peers
/connect <ip> <port>
/msg <peerIndex> <text>
/broadcast <text>
/share add <filepath>
/share list
/search <pattern>
/get <ip> <port> <filename>
/voice_send <ip> <udpPort> <file>
/voice_recv <udpPort> <outfile>
/quit
[net] accepted peer idx=0 from 127.0.0.1:51832 (waiting AUTH)
[auth] Peer 0 authed as 'usuario2'.
[bcast] usuario2: hola
[chat] usuario2: hola usuario1, mensaje directo
/msg 0 recibido ✓
```

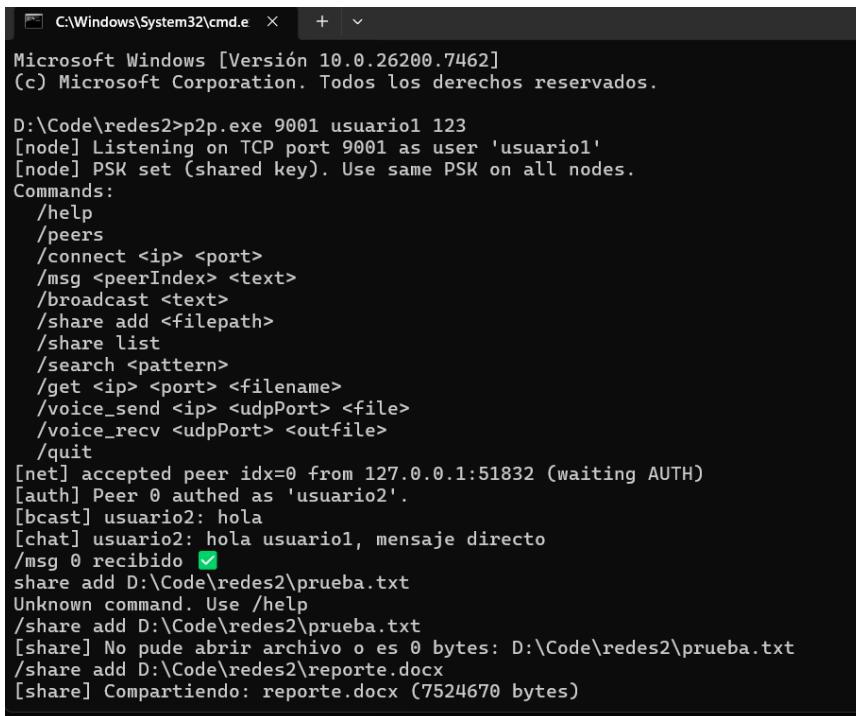
7.4. Prueba 4: Compartir archivo y listar

En **Consola A:**

```
/share add C:\Users\TUUSUARIO\Desktop\prueba.txt
```

```
/share list
```

Debe listar el archivo y tamaño.



```
C:\Windows\System32\cmd.e × + ▾
Microsoft Windows [Versión 10.0.26200.7462]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\Code\redes2>p2p.exe 9001 usuario1 123
[node] Listening on TCP port 9001 as user 'usuario1'
[node] PSK set (shared key). Use same PSK on all nodes.
Commands:
/help
/peers
/connect <ip> <port>
/msg <peerIndex> <text>
/broadcast <text>
/share add <filepath>
/share list
/search <pattern>
/get <ip> <port> <filename>
/voice_send <ip> <udpPort> <file>
/voice_recv <udpPort> <outfile>
/quit
[net] accepted peer idx=0 from 127.0.0.1:51832 (waiting AUTH)
[auth] Peer 0 authed as 'usuario2'.
[bcast] usuario2: hola
[chat] usuario2: hola usuario1, mensaje directo
/msg 0 recibido ✓
share add D:\Code\redes2\prueba.txt
Unknown command. Use /help
/share add D:\Code\redes2\prueba.txt
[share] No pude abrir archivo o es 0 bytes: D:\Code\redes2\prueba.txt
/share add D:\Code\redes2\reporte.docx
[share] Compartiendo: reporte.docx (7524670 bytes)
```

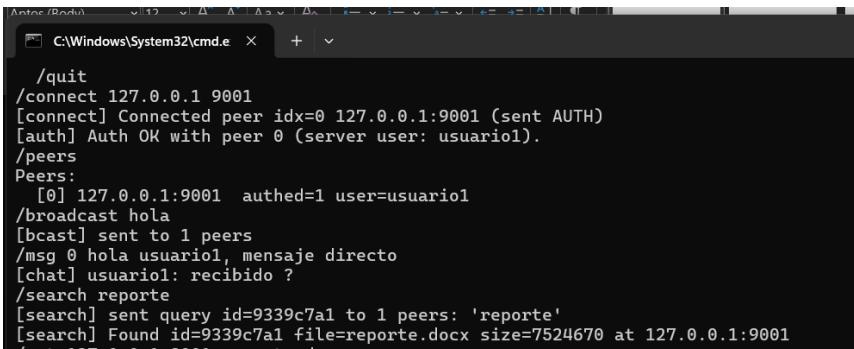
7.5. Prueba 5: Búsqueda distribuida (/search)

En **Consola B:**

```
/search prueba
```

Esperado en B (respuesta del nodo que tiene el archivo):

```
[search] Found id=... file=prueba.txt size=... at 127.0.0.1:9001
```



```
C:\Windows\System32\cmd.e
/quit
/connect 127.0.0.1 9001
[connect] Connected peer idx=0 127.0.0.1:9001 (sent AUTH)
[auth] Auth OK with peer 0 (server user: usuario1).
/peers
Peers:
[0] 127.0.0.1:9001 authed=1 user=usuario1
/broadcast hola
[bcast] sent to 1 peers
/msg 0 hola usuario1, mensaje directo
[chat] usuario1: recibido ?
/search reporte
[search] sent query id=9339c7a1 to 1 peers: 'reporte'
[search] Found id=9339c7a1 file=reporte.docx size=7524670 at 127.0.0.1:9001
```

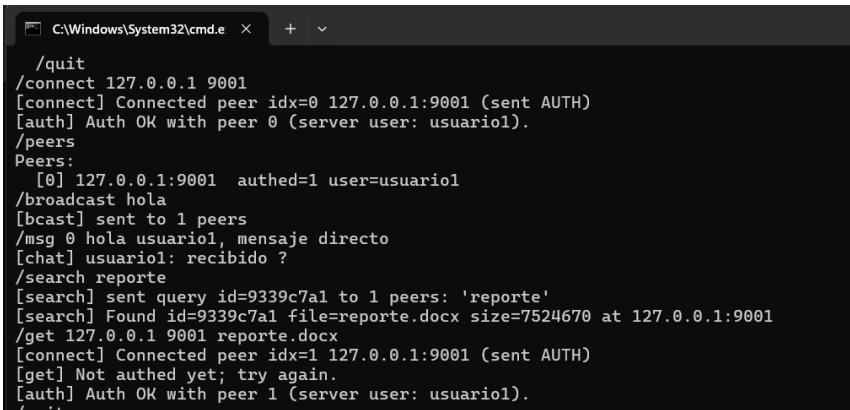
7.6. Prueba 6: Descarga de archivo (/get)

En **Consola B:**

/get 127.0.0.1 9001 prueba.txt

Esperado:

- En A:
 - [file] Sent 'prueba.txt' (...) to peer 0
- En B:
 - [file] Received 'prueba.txt' (...) saved as .\prueba.txt



```
C:\Windows\System32\cmd.e
/quit
/connect 127.0.0.1 9001
[connect] Connected peer idx=0 127.0.0.1:9001 (sent AUTH)
[auth] Auth OK with peer 0 (server user: usuario1).
/peers
Peers:
[0] 127.0.0.1:9001 authed=1 user=usuario1
/broadcast hola
[bcast] sent to 1 peers
/msg 0 hola usuario1, mensaje directo
[chat] usuario1: recibido ?
/search reporte
[search] sent query id=9339c7a1 to 1 peers: 'reporte'
[search] Found id=9339c7a1 file=reporte.docx size=7524670 at 127.0.0.1:9001
/get 127.0.0.1 9001 reporte.docx
[connect] Connected peer idx=1 127.0.0.1:9001 (sent AUTH)
[get] Not authed yet; try again.
[auth] Auth OK with peer 1 (server user: usuario1).
/quit
```

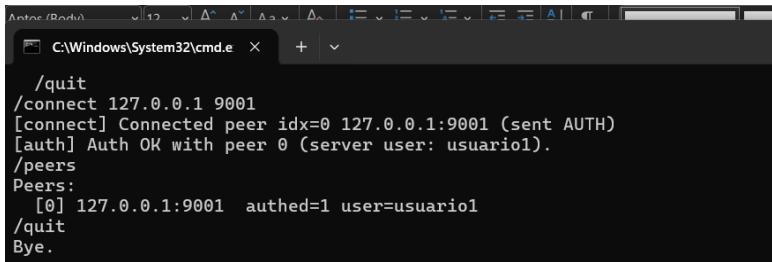
7.7. Prueba 7: Desconexión controlada

En **Consola B:**

/quit

Esperado en A:

[net] Peer 0 disconnected.



```
Antes/Rdida C:\Windows\System32\cmd.e + 
/quit
/connect 127.0.0.1 9001
[connect] Connected peer idx=0 127.0.0.1:9001 (sent AUTH)
[auth] Auth OK with peer 0 (server user: usuario1).
/peers
Peers:
[0] 127.0.0.1:9001 authed=1 user=usuario1
/quit
Bye.
```

7.8. Prueba 8: Seguridad (PSK incorrecta)

Abrir un tercer nodo con una clave distinta:

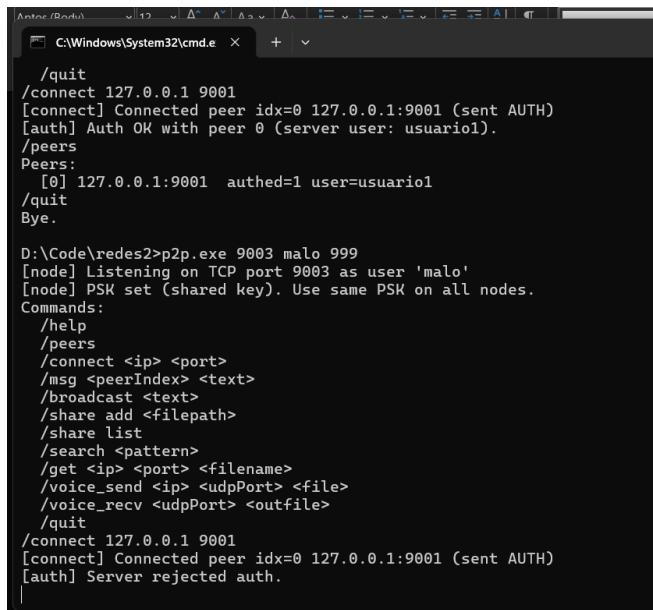
p2p.exe 9003 malo 999

/connect 127.0.0.1 9001

Esperado en A:

[auth] Peer X auth FAILED.

Esto valida que el sistema no acepta nodos con PSK distinto.



```
Antes/Rdida C:\Windows\System32\cmd.e + 
/quit
/connect 127.0.0.1 9001
[connect] Connected peer idx=0 127.0.0.1:9001 (sent AUTH)
[auth] Auth OK with peer 0 (server user: usuario1).
/peers
Peers:
[0] 127.0.0.1:9001 authed=1 user=usuario1
/quit
Bye.

D:\Code\redes2>p2p.exe 9003 malo 999
[node] Listening on TCP port 9003 as user 'malo'
[node] PSK set (shared key). Use same PSK on all nodes.
Commands:
/help
/peers
/connect <ip> <port>
/msg <peerIndex> <text>
/broadcast <text>
/share add <filepath>
/share list
/search <pattern>
/get <ip> <port> <filename>
/voice_send <ip> <udpPort> <file>
/voice_recv <udpPort> <outfile>
/quit
/connect 127.0.0.1 9001
[connect] Connected peer idx=0 127.0.0.1:9001 (sent AUTH)
[auth] Server rejected auth.
```

Conclusiones

En conclusión, esta práctica permitió construir un sistema P2P funcional en C sobre Windows, integrando múltiples conceptos fundamentales de redes: comunicación TCP cliente-servidor, multiplexación de sockets con select(), manejo de múltiples peers, diseño de un protocolo de aplicación por mensajes, y transferencia de archivos. El resultado fue un programa donde cada nodo puede operar

simultáneamente como servidor y cliente, logrando una arquitectura distribuida sin necesidad de un servidor central para mensajería o intercambio de recursos.

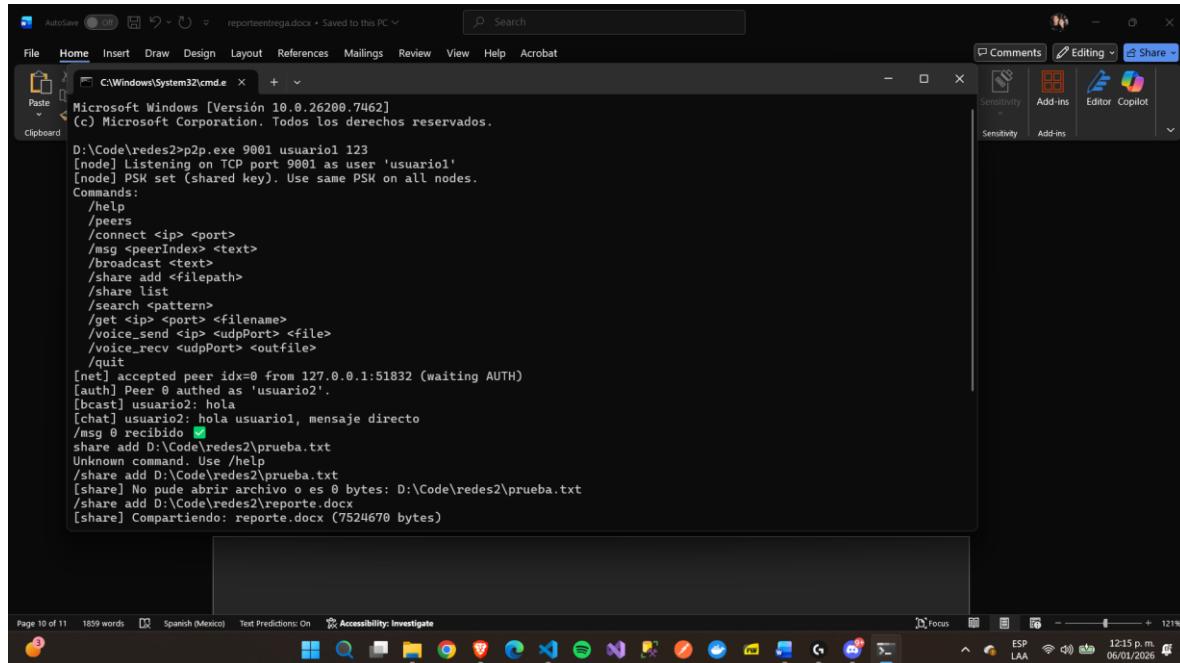
Uno de los aprendizajes más importantes fue comprender que en TCP los datos no llegan necesariamente “por mensajes” sino como un flujo; por ello, fue indispensable implementar un mecanismo de framing basado en salto de línea y un buffer acumulador. Esto hizo evidente por qué no se puede asumir que un recv() corresponde exactamente a un mensaje enviado. De la misma manera, se reforzó la importancia de mantener estados de conexión por peer (activo, autenticado, llave de sesión, buffer parcial), ya que el nodo interactúa con varios peers en paralelo.

En el apartado de seguridad, aunque el cifrado usado es demostrativo, se implementó correctamente el concepto de autenticación por llave compartida (PSK) y derivación de una llave de sesión, aplicando el principio de que ambos nodos deben poder comprobar que poseen la misma clave sin enviarla directamente. Además, se observó en pruebas que el orden del handshake es crucial: durante la depuración se detectó que enviar AUTHOK cifrado antes de que el cliente estuviera en estado autenticado causaba que el cliente nunca completara el proceso. Esto fue una evidencia clara de que en protocolos reales se debe definir cuidadosamente qué mensajes viajan en claro y cuáles se cifran, así como el momento exacto en que se activa el canal seguro.

La búsqueda distribuida con flooding y TTL mostró cómo se puede propagar una consulta a través de una red de nodos y evitar bucles usando un registro de IDs ya vistos y un contador de saltos. Esto es un concepto esencial en redes P2P y se relaciona con sistemas como redes de descubrimiento y consulta distribuida. Finalmente, las pruebas de transferencia de archivos confirmaron que es posible solicitar recursos directamente a un peer y recibirlas de forma confiable por TCP, manejando tamaños exactos de lectura y escritura.

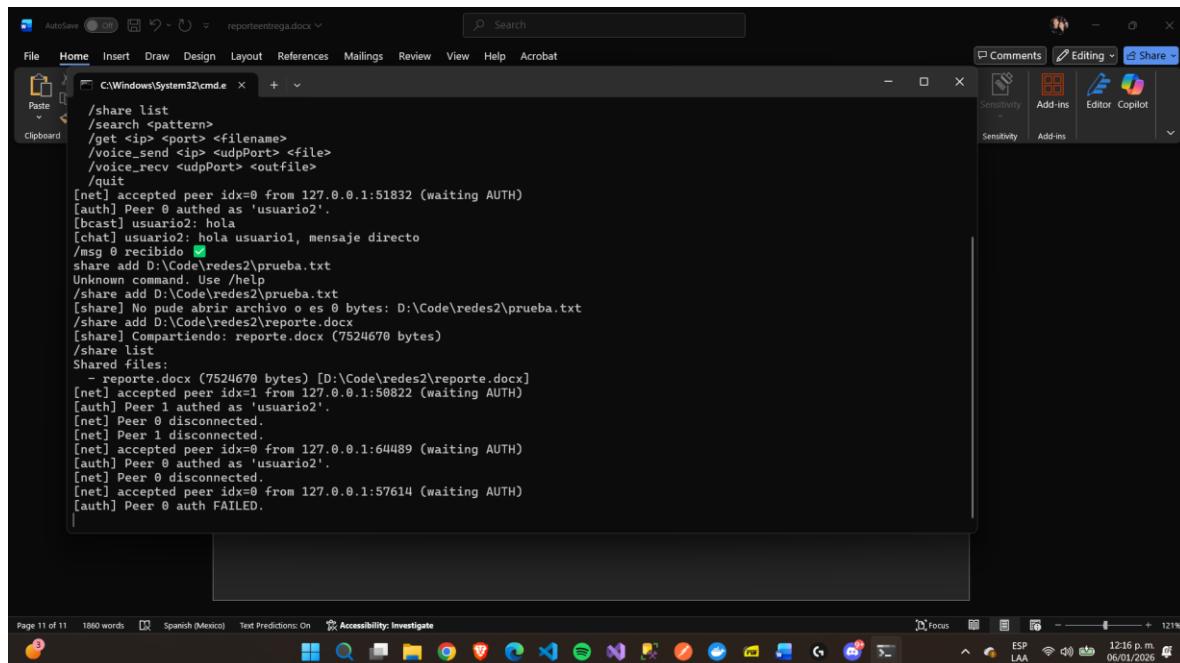
Flujo completo

Consola A



A screenshot of a Microsoft Word document titled "reportenteentrega.docx". The document content is a command-line session from a Windows terminal. The session shows two users, "usuario1" and "usuario2", interacting via a peer-to-peer network. User "usuario1" initiates a connection, sends a message, and shares a file. User "usuario2" receives the message and file share. The session ends with both users disconnecting.

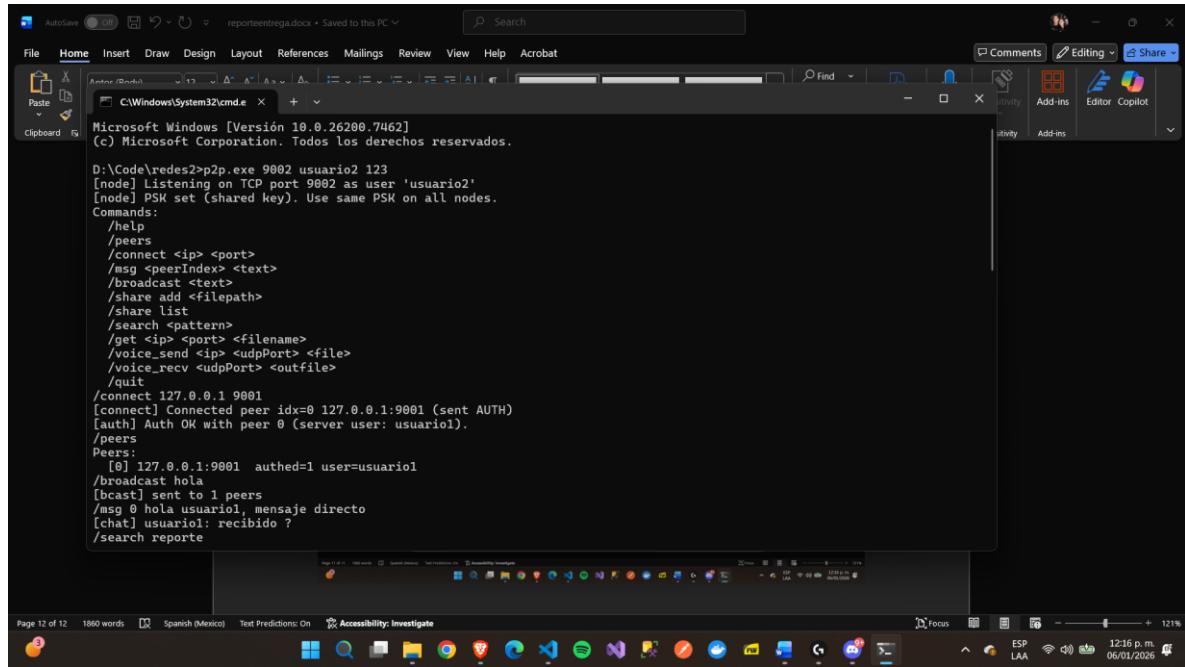
```
D:\Code\redes2>p2p.exe 9001 usuario1
[net] Listening on TCP port 9001 as user 'usuario1'
[net] PSK set (shared key). Use same PSK on all nodes.
Commands:
/help
/peers
/connect <ip> <port>
/msg <peerIndex> <text>
/broadcast <text>
/share add <filepath>
/share list
/search <pattern>
/get <ip> <port> <filename>
/voice_send <ip> <udpPort> <file>
/voice_recv <udpPort> <outfile>
/quit
[net] accepted peer idx=0 from 127.0.0.1:51832 (waiting AUTH)
[auth] Peer 0 authed as 'usuario2'.
[bcast] usuario2: hola
[chat] usuario2: hola usuario1, mensaje directo
[msg 0 recibido ]
share add D:\Code\redes2\prueba.txt
Unknown command. Use /help
/share add D:\Code\redes2\prueba.txt
[share] No pude abrir archivo o es 0 bytes: D:\Code\redes2\prueba.txt
/share add D:\Code\redes2\reporte.docx
[share] Compartiendo: reporte.docx (7524670 bytes)
```



A screenshot of a Microsoft Word document titled "reportenteentrega.docx". The document content is a command-line session from a Windows terminal. This session is identical to the one in the first screenshot, showing users "usuario1" and "usuario2" connecting, messaging, and sharing files. The session concludes with both users disconnecting.

```
/share list
/search <pattern>
/get <ip> <port> <filename>
/voice_send <ip> <udpPort> <file>
/voice_recv <udpPort> <outfile>
/quit
[net] accepted peer idx=0 from 127.0.0.1:51832 (waiting AUTH)
[auth] Peer 0 authed as 'usuario2'.
[bcast] usuario2: hola
[chat] usuario2: hola usuario1, mensaje directo
[msg 0 recibido ]
share add D:\Code\redes2\prueba.txt
Unknown command. Use /help
/share add D:\Code\redes2\prueba.txt
[share] No pude abrir archivo o es 0 bytes: D:\Code\redes2\prueba.txt
/share add D:\Code\redes2\reporte.docx
[share] Compartiendo: reporte.docx (7524670 bytes)
/share list
Shared files:
- reporte.docx (7524670 bytes) [D:\Code\redes2\reporte.docx]
[net] accepted peer idx=1 from 127.0.0.1:50822 (waiting AUTH)
[auth] Peer 1 authed as 'usuario2'.
[net] Peer 0 disconnected
[net] Peer 1 disconnected.
[net] accepted peer idx=0 from 127.0.0.1:64489 (waiting AUTH)
[auth] Peer 0 authed as 'usuario2'.
[net] Peer 0 disconnected
[net] accepted peer idx=0 from 127.0.0.1:57614 (waiting AUTH)
[auth] Peer 0 auth FAILED.
```

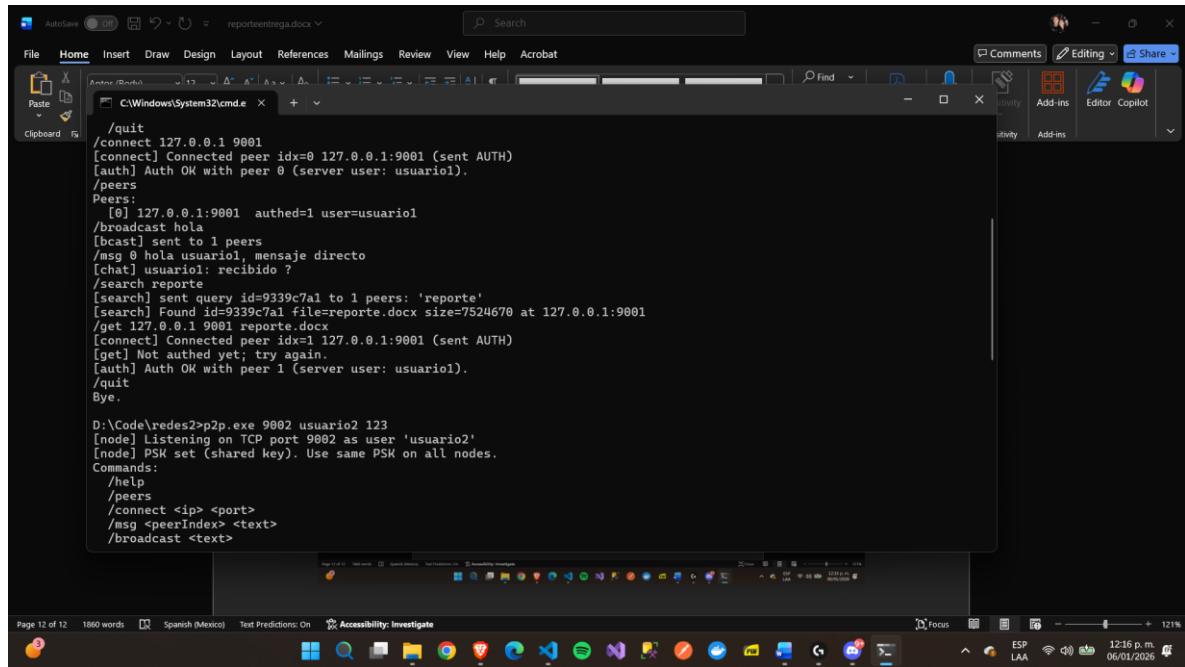
Consola B



Microsoft Windows [Versión 10.0.26200.7462]
(c) Microsoft Corporation. Todos los derechos reservados.

```
D:\Code\redes2>p2p.exe 9002 usuario2 123
[node] Listening on TCP port 9002 as user 'usuario2'
[node] PSK set (shared key). Use same PSK on all nodes.
Commands:
/help
/peers
/connect <ip> <port>
/msg <peerIndex> <text>
/broadcast <text>
/share add <filepath>
/share list
/search <pattern>
/get <ip> <port> <filename>
/voice_send <ip> <udpPort> <file>
/voice_recv <udpPort> <outfile>
/quit
/connect 127.0.0.1 9001
[connect] Connected peer idx=0 127.0.0.1:9001 (sent AUTH)
[auth] Auth OK with peer 0 (server user: usuario1).
/peers
Peers:
[0] 127.0.0.1:9001 authed=1 user=usuario1
/broadcast hola
[bcast] sent to 1 peers
/msg 0 hola usuario1, mensaje directo
[chat] usuario1: recibido ?
/search reporte
```

Page 12 of 12 1860 words Spanish (Mexico) Text Predictions: On Accessibility: Investigate



```
/quit
/connect 127.0.0.1 9001
[connect] Connected peer idx=0 127.0.0.1:9001 (sent AUTH)
[auth] Auth OK with peer 0 (server user: usuario1).
/peers
Peers:
[0] 127.0.0.1:9001 authed=1 user=usuario1
/broadcast hola
[bcast] sent to 1 peers
/msg 0 hola usuario1, mensaje directo
[chat] usuario1: recibido ?
/search reporte
[search] sent query id=9339c7a1 to 1 peers: 'reporte'
[search] Found id=9339c7a1 file=reporte.docx size=7524670 at 127.0.0.1:9001
/get 127.0.0.1 9001 reporte.docx
[connect] Connected peer idx=1 127.0.0.1:9001 (sent AUTH)
[get] Not authed yet; try again.
[auth] Auth OK with peer 1 (server user: usuario1).
/quit
Bye.

D:\Code\redes2>p2p.exe 9002 usuario2 123
[node] Listening on TCP port 9002 as user 'usuario2'
[node] PSK set (shared key). Use same PSK on all nodes.
Commands:
/help
/peers
/connect <ip> <port>
/msg <peerIndex> <text>
/broadcast <text>
```

Page 12 of 12 1860 words Spanish (Mexico) Text Predictions: On Accessibility: Investigate

```
/quit
[connect] Connected peer idx=0 127.0.0.1:9001 (sent AUTH)
[auth] Auth OK with peer 0 (server user: usuariol).
/peers
Peers:
[0] 127.0.0.1:9001 authed=1 user=usuariol
/qut
Bye.

D:\Code\redes2>p2p.exe 9003 malo 999
[node] Listening on TCP port 9003 as user 'malo'
[node] PSK set (shared key). Use same PSK on all nodes.

Commands:
/help
/peers
/connect <ip> <port>
/msg <peerIndex> <text>
/broadcast <text>
/share add <filepath>
/share list
/search <pattern>
/get <ip> <port> <filename>
/voice_send <ip> <udpPort> <file>
/voice_recv <udpPort> <outfile>
/qut
/connect 127.0.0.1 9001
[connect] Connected peer idx=0 127.0.0.1:9001 (sent AUTH)
[auth] Server rejected auth.
```

Referencias

Kurose, J. F., & Ross, K. W. (2021). *Computer networking: A top-down approach* (8th ed.). Pearson Education.

Stevens, W. R., Fenner, B., & Rudoff, A. M. (2003). *UNIX network programming: The sockets networking API* (Vol. 1, 3rd ed.). Addison-Wesley Professional.

Microsoft Corporation. (2023). *Windows Sockets 2 (Winsock) documentation*.

Microsoft Learn.

<https://learn.microsoft.com/en-us/windows/win32/winsock/windows-sockets-start-page>

Tanenbaum, A. S., & van Steen, M. (2017). *Distributed systems: Principles and paradigms* (2nd ed.). Pearson Education.

Comer, D. E. (2018). *Internetworking with TCP/IP: Principles, protocols, and architecture* (6th ed.). Pearson Education.