

INSTITUTO POLITÉCNICO NACIONAL



Práctica 5

Integrantes:

- **Flores Morales Aldahir Andrés**
- **Velasco Jiménez Luis Antonio**

Fecha de entrega: 06/01/2025

Materia: Aplicaciones para comunicaciones en red

Grupo: 6CM1

Introducción

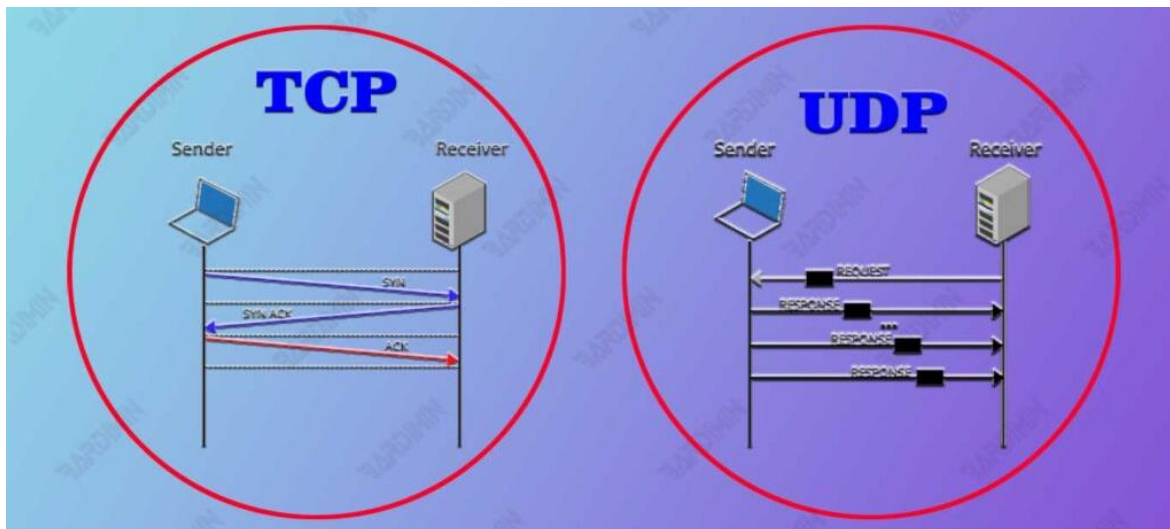
En el ámbito de las comunicaciones en redes, los sockets de datagrama juegan un papel crucial para implementar sistemas de comunicación rápida y ligera entre dispositivos. A diferencia de los sockets orientados a conexión (como TCP), los sockets de datagrama (basados en el protocolo UDP, User Datagram Protocol) no requieren de una conexión establecida y garantizada entre el emisor y el receptor, lo que los hace ideales para aplicaciones donde la velocidad y eficiencia son más importantes que la confiabilidad absoluta, como en transmisiones de audio o video en tiempo real, y sistemas de control en red.

Esta práctica se centra en comprender el funcionamiento de los sockets de datagrama utilizando sockets no bloqueantes. El servidor será capaz de recibir mensajes de uno o varios clientes, y los clientes, a su vez, podrán enviar datos en forma de paquetes al servidor sin necesidad de mantener una conexión permanente. Esto implica el uso de paquetes pequeños que se envían y reciben de manera independiente, permitiendo explorar conceptos como la fragmentación, el manejo de pérdida de paquetes y los mecanismos de retransmisión en caso de errores.

Los Sockets no bloqueantes en UDP permiten que el programa continúe ejecutándose mientras espera datos, en lugar de quedarse detenido ("bloqueado") hasta que ocurra una operación de E/S (entrada/salida). Esto es útil en aplicaciones donde se requiere manejar múltiples tareas simultáneamente, como un servidor que debe procesar varias solicitudes de clientes sin detenerse.

Sockets de Datagrama

Los sockets de datagrama son una herramienta fundamental en el ámbito de la comunicación de redes de computadoras y se utilizan ampliamente para implementar aplicaciones de red que requieren transmisión rápida y eficiente de datos sin la necesidad de establecer una conexión continua entre los dispositivos que participan en la comunicación. A diferencia de los sockets orientados a conexión (como los de TCP), los sockets de datagrama operan sobre el protocolo UDP (User Datagram Protocol), que es un protocolo de transporte no orientado a conexión. Esto significa que los datos se envían en paquetes independientes llamados datagramas, sin confirmación de llegada o garantías de reordenamiento en la entrega.



Características de los Sockets de Datagrama

1. **Transmisión sin conexión** : Los sockets de datagrama permiten la transmisión de datos sin establecer una conexión continua entre el emisor y el receptor. Cada datagrama enviado es independiente y no necesita de una sesión o enlace entre el cliente y el servidor. Esto contrasta con TCP, que requiere establecer y mantener una conexión para el intercambio de información.
2. **Protocolo UDP (User Datagram Protocol)** : Los sockets de datagramas se basan en UDP, un protocolo de transporte que no garantiza la entrega confiable de los paquetes. Los paquetes enviados pueden llegar a su destino en el orden incorrecto, llegar duplicados o incluso perderse sin aviso. Aunque esto puede parecer una limitación, resulta ventajoso en aplicaciones donde la velocidad es más crítica que la confiabilidad, como en transmisiones de video en tiempo real, juegos en línea o sistemas de monitoreo en red.
3. **Unidad de Transmisión: el Datagrama** : En los sockets de datagrama, la información se empaqueta en pequeñas unidades llamadas datagramas. Cada datagrama es un paquete que contiene una porción de datos, así como la información de encabezado necesaria para ser dirigido al destino correcto. Los datagramas no tienen una longitud fija, aunque su tamaño generalmente se limita a 65.535 bytes debido a las restricciones del protocolo UDP. Los datagramas permiten a las aplicaciones de red dividir grandes cantidades de datos en paquetes más pequeños que se pueden enviar de manera independiente.
4. **Velocidad y Eficiencia** : La ausencia de mecanismos de control de flujo y verificación en los sockets de datagrama permite una transmisión de datos rápida y con baja latencia. Esto los hace ideales para aplicaciones en tiempo real que requieren que los datos lleguen rápidamente sin importar si algunos paquetes se pierden en el camino. Por ejemplo, en una transmisión de voz o video en tiempo real, perder algunos paquetes no suele afectar significativamente la calidad de la transmisión, mientras que retrasar los paquetes puede causar un impacto negativo.
5. **Simpleza de Implementación** : Los sockets de datagrama son simples de

implementar, ya que no se requiere el código adicional para gestionar la creación, mantenimiento y cierre de una conexión. Solo es necesario crear el socket, especificar el destino y enviar el datagrama. Esto hace que los sockets de datagrama sean especialmente útiles para aplicaciones donde se requieren muchas comunicaciones breves con diferentes destinos, como en sistemas de detección de red o envío de comandos simples a Múltiples dispositivos.

Funcionamiento Básico de los Sockets de Datagramas

El proceso de envío y recepción de información a través de sockets de datagrama implica los siguientes pasos:

1. **Creación del Socket** : Tanto el emisor como el receptor deben crear un socket de datagrama. En la mayoría de los lenguajes de programación, este proceso es sencillo y solo requiere especificar el tipo de socket (en este caso, un socket de datagrama basado en UDP).
2. **Creación y Envío de Datagramas** : Para enviar datos, el emisor debe empaquetar los datos en un datagrama, que incluye tanto la información de los datos como la dirección IP y el número de puerto del receptor. Una vez configurado el datagrama, se envía al receptor sin esperar ninguna respuesta.
3. **Recepción del Datagrama** : El receptor, por su parte, escucha en un puerto específico, esperando recibir datagramas. Al recibir uno, el receptor desempaqueta el datagrama para obtener los datos y la información sobre el emisor. En la mayoría de las implementaciones, el receptor debe especificar el tamaño máximo del datagrama que espera recibir.
4. **Procesamiento de los Datos** : Una vez que el datagrama es recibido, la aplicación puede procesar los datos según sea necesario. Si el protocolo de la aplicación requiere algún tipo de confirmación de llegada o reordenamiento de los paquetes, estos mecanismos deben implementarse a nivel de aplicación, ya que UDP no los proporciona.
5. **Reintentos y Control de Errores (si es necesario)** : Si la aplicación requiere confiabilidad en la entrega de los datagramas, es necesario agregar mecanismos adicionales. Esto puede incluir la solicitud de reenvío de paquetes faltantes o la implementación de un número de secuencia para ordenar los paquetes en el receptor.

Sockets No Bloqueantes

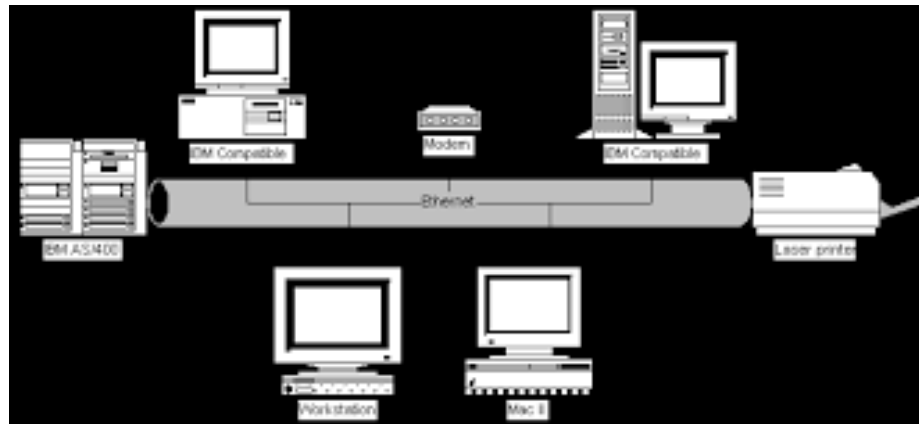
Los **sockets no bloqueantes** son una técnica utilizada en redes para evitar que un programa se quede detenido esperando que una operación de entrada/salida (E/S) se complete, como enviar o recibir datos. En lugar de bloquear el flujo del programa, las operaciones de E/S se realizan de manera asíncrona, devolviendo un error si no están listas en ese momento.

Características principales:

1. **Sin espera activa**: Permiten que el programa realice otras tareas mientras espera datos o conexiones.
2. **Flexibilidad**: Son útiles para aplicaciones que necesitan manejar múltiples conexiones simultáneamente, como servidores o aplicaciones en tiempo real.
3. **Uso**: Se configuran llamando al método `setblocking(False)` en un socket.

4. **Control del flujo:** Las operaciones como `recv` o `send` pueden generar excepciones como `BlockingIOError` si no están listas para ejecutarse.

El uso de UDP con sockets no bloqueantes permite construir aplicaciones rápidas y eficientes que pueden manejar múltiples tareas simultáneamente. Aunque no garantiza la confiabilidad, su simplicidad y velocidad lo convierten en una herramienta poderosa para aplicaciones en tiempo real y de alto rendimiento.



Pruebas:

Primero se debe ejecutar el servidor para que inicie el programa. Una vez que inicia nuestro servidor, empieza a esperar los datagramas:

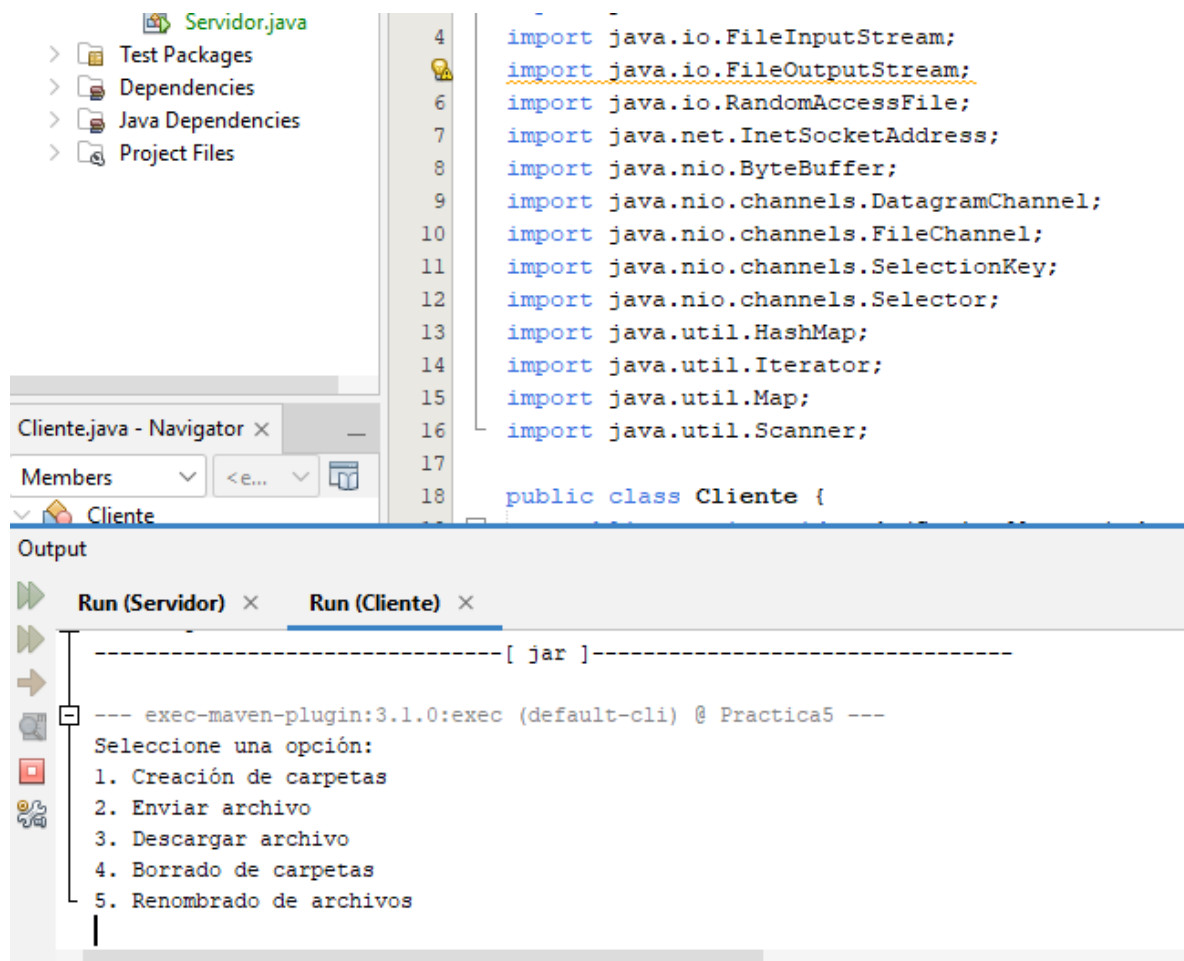
```
Output - Run (Servidor)

cd C:\Users\aldahir\Documents\NetBeansProjects\Practica5; "JAVA_HOME=C:\\Program Files\\
Running NetBeans Compile On Save execution. Phase execution is skipped and output direc
Scanning for projects...

-----< com.mycompany:Practica5 >-----
Building Practica5 1.0-SNAPSHOT
-----[ jar ]-----

--- exec-maven-plugin:3.1.0:exec (default-cli) @ Practica5 ---
Bienvenido a su Drive
Servidor iniciado, esperando cliente...
```

Cuando el cliente se ejecuta, despliega un menú en la consola para que el usuario pueda realizar la operación correspondiente:



```
4 import java.io.FileInputStream;
5 import java.io.FileOutputStream;
6 import java.io.RandomAccessFile;
7 import java.net.InetSocketAddress;
8 import java.nio.ByteBuffer;
9 import java.nio.channels.DatagramChannel;
10 import java.nio.channels.FileChannel;
11 import java.nio.channels.SelectionKey;
12 import java.nio.channels.Selector;
13 import java.util.HashMap;
14 import java.util.Iterator;
15 import java.util.Map;
16 import java.util.Scanner;
17
18 public class Cliente {
```

Output

Run (Servidor) × Run (Cliente) ×

-----[jar]-----

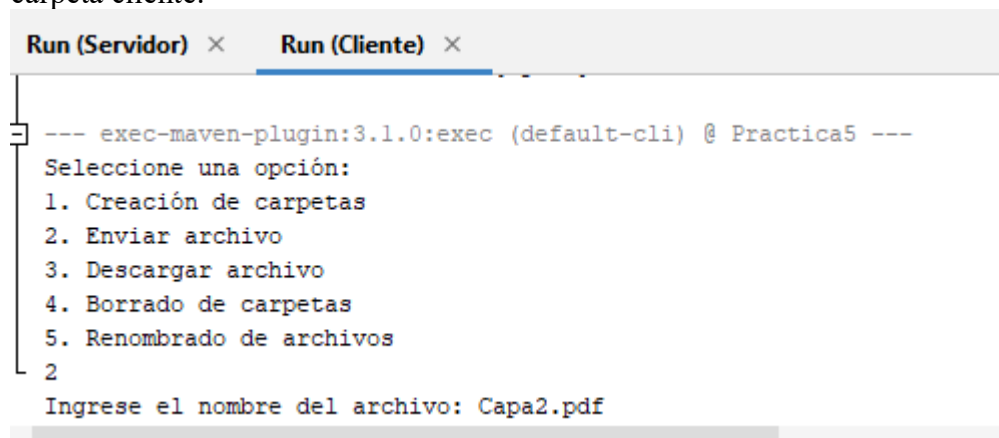
--- exec-maven-plugin:3.1.0:exec (default-cli) @ Practica5 ---

Seleccione una opción:

1. Creación de carpetas
2. Enviar archivo
3. Descargar archivo
4. Borrado de carpetas
5. Renombrado de archivos

Subir archivos:

Para poder subir un archivo, se coloca la opción 2 en el menú, luego nos aparecerá un mensaje para ingresar un archivo, el cual debe ir seguido de su extensión y debe encontrarse en la carpeta cliente.



Run (Servidor) × Run (Cliente) ×

--- exec-maven-plugin:3.1.0:exec (default-cli) @ Practica5 ---

Seleccione una opción:

1. Creación de carpetas
2. Enviar archivo
3. Descargar archivo
4. Borrado de carpetas
5. Renombrado de archivos

2

Ingrese el nombre del archivo: Capa2.pdf

Una vez que se selecciona, se fragmenta el archivo en datagramas y se empiezan a mandar al servidor en donde aparece un mensaje con el número de paquete y el porcentaje de envío:

Run (Servidor) ×	Run (Cliente) ×
Enviando paquete 598 con 1024 bytes. Progreso: 98%	Paquete 442 recibido. Progreso: 72%
Enviando paquete 599 con 1024 bytes. Progreso: 98%	Paquete 443 recibido. Progreso: 72%
Enviando paquete 600 con 1024 bytes. Progreso: 98%	Paquete 444 recibido. Progreso: 72%
Enviando paquete 601 con 1024 bytes. Progreso: 98%	Paquete 445 recibido. Progreso: 73%
Enviando paquete 602 con 1024 bytes. Progreso: 98%	Paquete 446 recibido. Progreso: 73%
Enviando paquete 603 con 1024 bytes. Progreso: 98%	Paquete 447 recibido. Progreso: 73%
Enviando paquete 604 con 1024 bytes. Progreso: 99%	Paquete 448 recibido. Progreso: 73%
Enviando paquete 605 con 1024 bytes. Progreso: 99%	Paquete 449 recibido. Progreso: 73%
Enviando paquete 606 con 1024 bytes. Progreso: 99%	Paquete 450 recibido. Progreso: 73%
Enviando paquete 607 con 1024 bytes. Progreso: 99%	Paquete 451 recibido. Progreso: 74%
Enviando paquete 608 con 1024 bytes. Progreso: 99%	Paquete 452 recibido. Progreso: 74%
Enviando paquete 609 con 1024 bytes. Progreso: 99%	Paquete 453 recibido. Progreso: 74%
Enviando paquete 610 con 714 bytes. Progreso: 99%	Paquete 454 recibido. Progreso: 74%
	Paquete 455 recibido. Progreso: 74%
	Paquete 456 recibido. Progreso: 74%
	Paquete 457 recibido. Progreso: 74%
	Paquete 458 recibido. Progreso: 75%
	Paquete 459 recibido. Progreso: 75%
	Paquete 460 recibido. Progreso: 75%
	Paquete 461 recibido. Progreso: 75%
	Paquete 462 recibido. Progreso: 75%
	Paquete 463 recibido. Progreso: 75%
	Paquete 464 recibido. Progreso: 76%
	Paquete 465 recibido. Progreso: 76%
	Paquete 466 recibido. Progreso: 76%

Descargar archivo

Para descargar archivos se coloca la opción 3 y luego de esto se ingresa el nombre del archivo. Se debe agregar la extensión del archivo a descargar.

```

Seleccione una opción:
1. Creación de carpetas
2. Enviar archivo
3. Descargar archivo
4. Borrado de carpetas
5. Renombrado de archivos
3
Ingrese el nombre del archivo a descargar: Nuevo.txt

```


Luego de esto se empieza a fragmentar el archivo en datagramas y se empieza a enviar del servidor al cliente.

```

--- exec-maven-plugin:3.1.0:exec (default-cli) @ Practica5 ---
Bienvenido a su Drive
Servidor iniciado, esperando cliente...
Enviando archivo: Nuevo.txt de tamaño 54 bytes.
Enviando paquete 0 con 54 bytes. Progreso: 100%
Archivo enviado completamente.

5. Renombrado de archivos
3
Ingrese el nombre del archivo a descargar: Nuevo.txt
Solicitud de descarga enviada para el archivo: Nuevo.txt
Tamaño del archivo recibido: 54 bytes.
Paquete 0 recibido. Progreso: 100%
Archivo descargado completamente.
Seleccione una opción:
1. Creación de carpetas
2. Enviar archivo

```


Crear carpeta:


Para crear una carpeta, se coloca la opción 1 y luego se ingresa el nombre de la carpeta. El servidor recibe el nombre de la carpeta y la crea en la carpeta servidor.


```


Archivo descargado completamente.
Seleccione una opción:
1. Creación de carpetas
2. Enviar archivo
3. Descargar archivo
4. Borrado de carpetas
5. Renombrado de archivos
1
Ingrese el nombre de la carpeta a crear: NuevaCarpeta
Solicitud de creación enviada: NuevaCarpeta
Paquete 0 enviado y recibido correctamente
Seleccione una opción:
1. Creación de carpetas

```

 NuevaCarpeta
04/01/2025 12:47 p. m.
Carpeta de archivos

 Capa2
04/01/2025 12:45 p. m.
Microsoft Edge P...

 Ldf
03/01/2025 12:26 p. m.
Documento de te...

 Nuevo
03/01/2025 08:18 p. m.
Documento de te...

Run (Servidor) ×
Run (Cliente) ×

```

cd C:\Users\aldahir\Documents\NetBeans\
Running NetBeans Compile On Save execu
Scanning for projects...

-----< com.mycompany:|
Building Practica5 1.0-SNAPSHOT
-----[ jar ]
--- exec-maven-plugin:3.1.0:exec (defa
Bienvenido a su Drive
Servidor iniciado, esperando cliente..
Enviando archivo: Nuevo.txt de tamaño !
Enviando paquete 0 con 54 bytes. Progre
Archivo enviado completamente.
Paquete 0 recibido correctamente
Carpeta creada: NuevaCarpeta

```


Borrar carpeta:

Para borrar una carpeta se coloca el número 4 y luego el nombre de la carpeta a borrar, una vez hecho esto, se envía el nombre por medio de datagrama y se utiliza una función para borrar la carpeta.

```
Seleccione una opción:
1. Creación de carpetas
2. Enviar archivo
3. Descargar archivo
4. Borrado de carpetas
5. Renombrado de archivos
4
Ingrese el nombre de la carpeta a borrar: Nt
Solicitud de creación enviada: NuevaCarpeta
Paquete 0 enviado y recibido correctamente
Seleccione una opción:
1. Creación de carpetas
```

Al recibir el nombre de la carpeta se borra automáticamente:

Herramientas > NetBeansProjects > Practicas > Servidor				▼ 🔍		Buscar en servidor
📁 Photo Print				^		
Nombre	Fecha de modificación	Tipo	Tamaño			
📁 Adios	04/01/2025 12:32 p. m.	Carpeta de archivos				
📁 hgffhjkjhjg	03/01/2025 03:15 p. m.	Carpeta de archivos				
📁 Hola	04/01/2025 12:32 p. m.	Carpeta de archivos				
📁 HolaAmigos	03/01/2025 01:18 p. m.	Carpeta de archivos				
📄 Capa2	04/01/2025 12:45 p. m.	Microsoft Edge P...	611 K			
📄 Ldf	03/01/2025 12:26 p. m.	Documento de te...	1 K			
📄 Nuevo	03/01/2025 08:18 p. m.	Documento de te...	1 K			

Renombrar archivos:

Para poder renombrar un archivo, se coloca el número 5, luego se coloca el nombre del archivo a renombrar, seguido de un punto y su extensión. Luego de esto aparece un mensaje en donde se debe ingresar el nuevo nombre del archivo seguido de la extensión

Seleccione una opción:

1. Creación de carpetas
2. Enviar archivo
3. Descargar archivo
4. Borrado de carpetas
5. Renombrado de archivos

5

Ingrese el nombre del archivo a modificar seguido de la extensión: Ldf.txt

Ingrese el nuevo nombre seguido de la extensión: ArchivoRenombrado.txt

Solicitud de modificación enviada.

Nombre actual: Ldf.txt

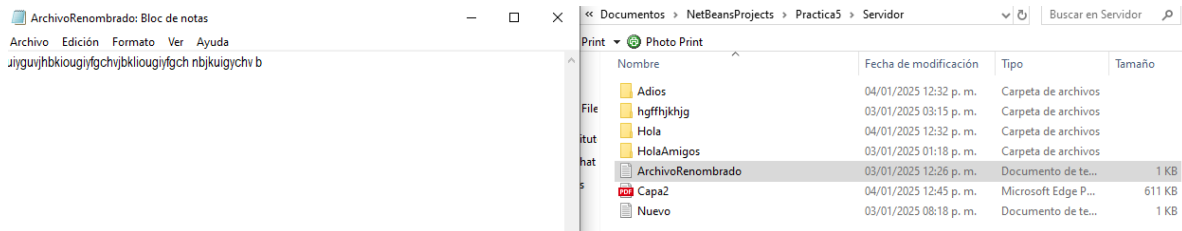
Nuevo nombre: ArchivoRenombrado.txt

Paquete 0 enviado y recibido correctamente

Seleccione una opción:

1. Creación de carpetas
2. Enviar archivo
3. Descargar archivo

A continuación se observa el archivo renombrado y no se afecta su conetnido:



Conclusión

Los sockets no bloqueantes que usan datagrama ofrecen una forma rápida y eficiente de comunicación en red para aplicaciones que requieren baja latencia y pueden tolerar la pérdida de algunos paquetes. Al no estar orientados a conexión, permiten que los desarrolladores diseñen aplicaciones ligeras y de alto rendimiento, aunque en caso de requerir confiabilidad, es necesario implementar un manejo adicional a nivel de aplicación. Esto hace que los sockets de datagrama sean una elección versátil para una amplia variedad de aplicaciones de red, desde streaming en tiempo real hasta videojuegos en línea y consultas de DNS, destacándose especialmente en entornos donde la rapidez y el bajo uso de recursos son prioritarios.