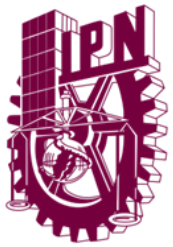


Instituto Politécnico Nacional

IPN

Escuela Superior de Cómputo

ESCOM

Academia de Redes Aplicaciones para comunicaciones en red

6CV2

Práctica 5

“Servicio TCP de la capa de transporte en Java”

Alumna:

Navarrete Becerril Sharon Anette

Fecha de entrega: “Lunes, 21-Octubre-2024”

Profesor: Ojeda Santillan Rodrigo

OBJETIVO

Desarrollar una comprensión técnica y práctica del Protocolo de Control de Transmisión (TCP) a través de la implementación de un servidor y un cliente utilizando Node.js, con el fin de observar el comportamiento y flujo de datos en una red TCP, incluyendo el análisis de tráfico mediante Wireshark.

INTRODUCCIÓN

El Protocolo de Control de Transmisión (TCP) es un protocolo fundamental en la capa de transporte de los modelos OSI y TCP/IP. A diferencia de otros protocolos como UDP, TCP está orientado a la conexión y garantiza una transmisión fiable y en orden de los datos. Su capacidad para gestionar conexiones, realizar seguimiento de la entrega de datos y controlar el flujo de información lo convierte en una opción preferida para aplicaciones que requieren integridad y consistencia en la transmisión de datos. En este proyecto, se implementará un servidor y un cliente TCP utilizando el módulo `net` de Node.js para observar cómo funcionan las conexiones TCP y analizar el tráfico de red generado utilizando la herramienta Wireshark.

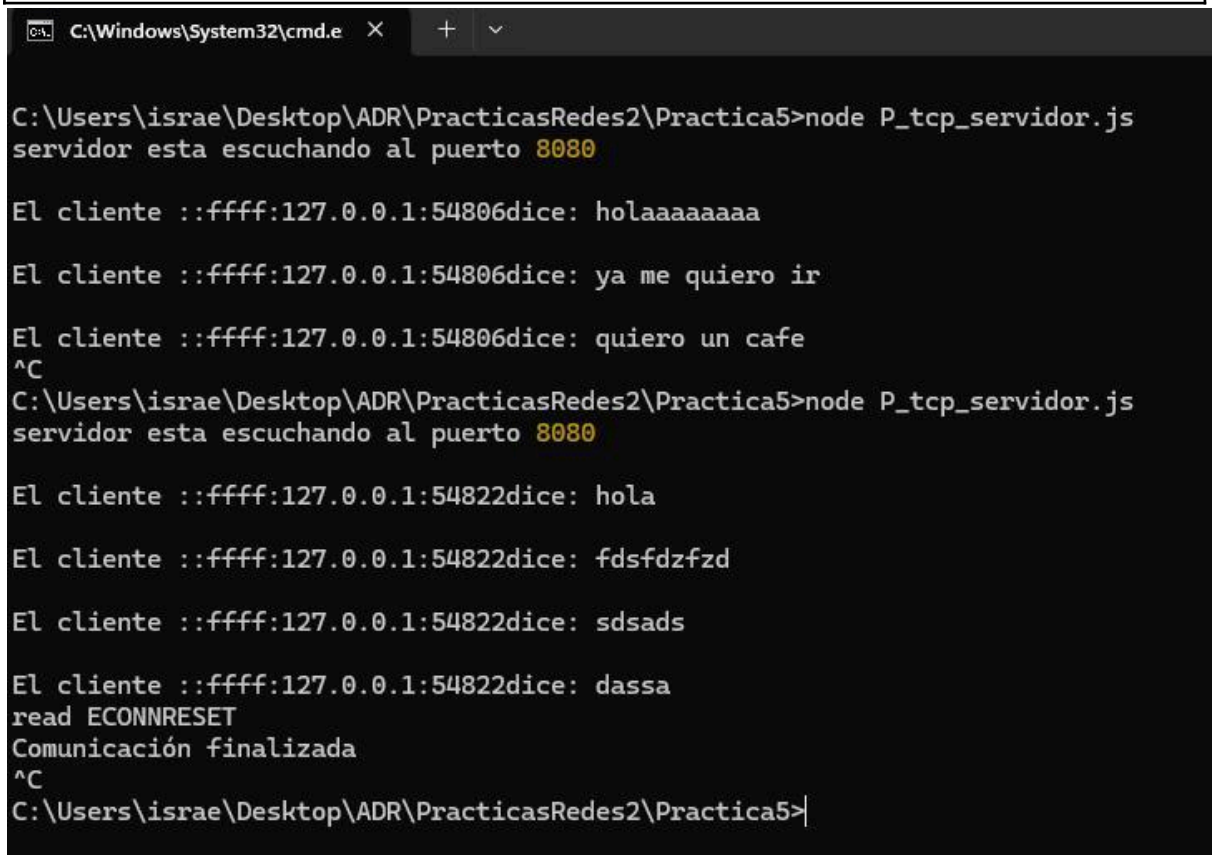
DESARROLLO

Para poner en práctica el concepto de TCP, se crearon dos scripts en Node.js: uno para el servidor (`P_tcp_servidor.js`) y otro para el cliente (`P_tcp_cliente.js`). El servidor utiliza el módulo `net` para crear un servicio TCP que escucha en el puerto 8080. Una vez que un cliente establece la conexión, el servidor gestiona los mensajes entrantes y envía respuestas confirmando la recepción de los datos.

Código del servidor:

```
const net = require('net');
const server = net.createServer();
server.on('connection', (socket)=>{
socket.on('data', (data)=>{
console.log('\nEl cliente ' + socket.remoteAddress + ":" +
socket.remotePort + "dice: " + data);
socket.write('Mensaje recibido');
});
socket.on('close', ()=>{
console.log('Comunicación finalizada');
});

socket.on('error', (err)=>{
    console.log(err.message);
});
});
```

```
server.listen(8080, ()=>{
console.log('servidor esta escuchando al puerto', server.address().port);
});
```



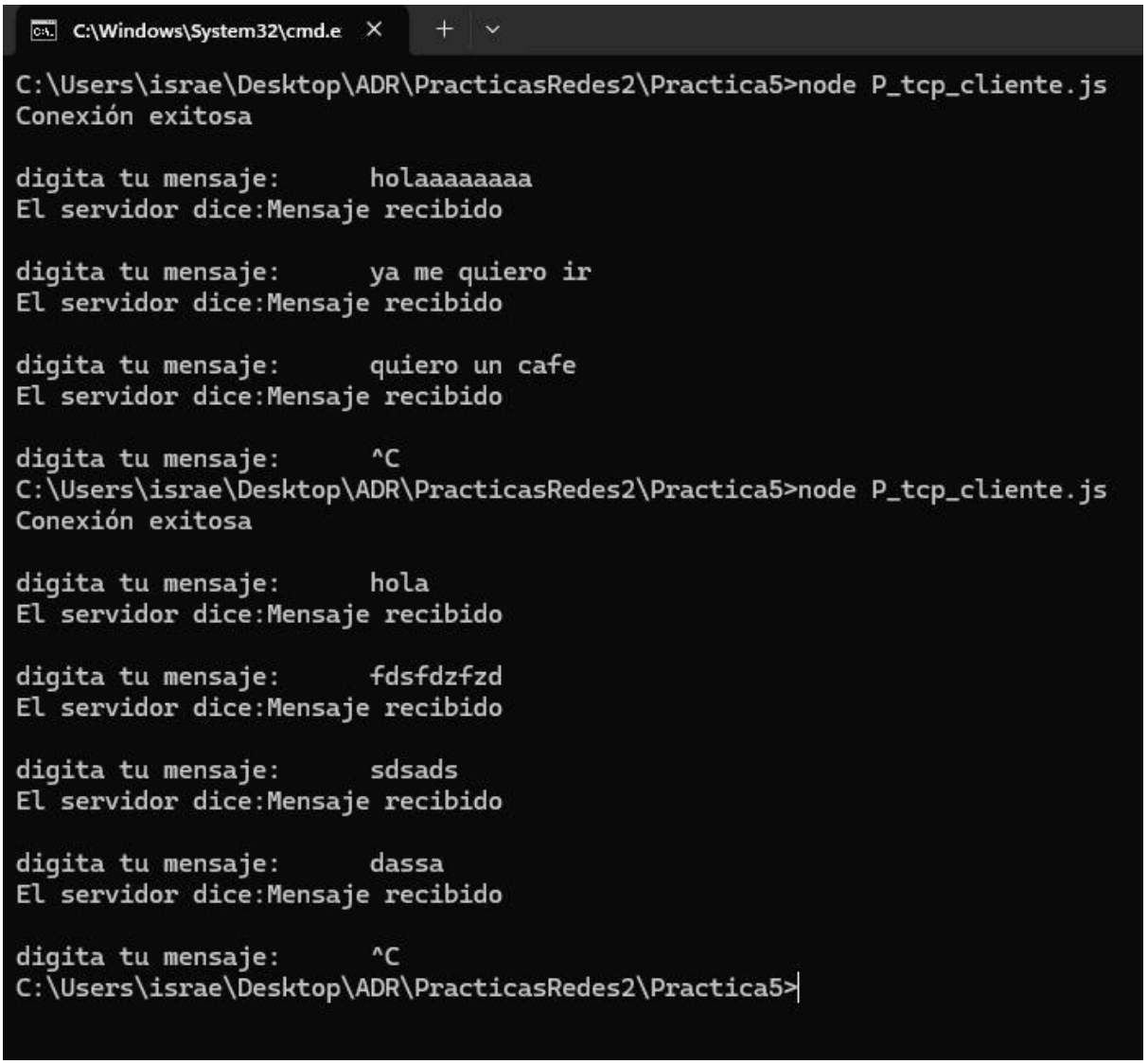
Cliente

Por otro lado, el cliente también utiliza el módulo `net` junto con la biblioteca `readline-sync` para permitir la entrada de datos desde la línea de comandos, y posteriormente envía estos datos al servidor. Ambos scripts permiten observar en tiempo real el intercambio de datos entre el cliente y el servidor. Para obtener una visión más detallada de los paquetes de datos transmitidos

Código del cliente:

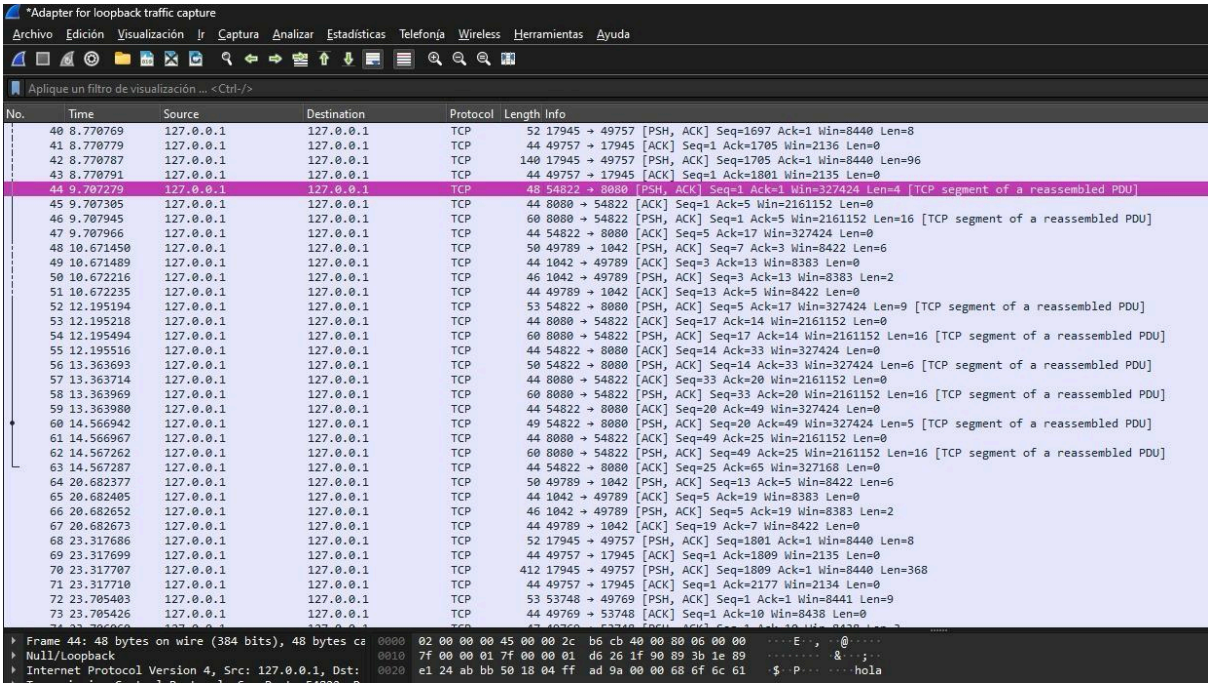
```
const net = require('net');
const readline = require('readline-sync');

const configuracion = {
    port: 8080,
    host: '127.0.0.1'
};

const cliente = net.createConnection(configuracion);

cliente.on('connect', ()=>{
    console.log('Conexión exitosa');
    sendLine();
```

```
});

cliente.on('data', (data)=>{
    console.log('El servidor dice:' + data);
    sendLine();
});

cliente.on('error', (err)=>{
    console.log(err.message);
});

function sendLine() {
    var line = readline.question('\ndigita tu mensaje: \t');
    if (line == "0") {
        cliente.end();
    }else{
        cliente.write(line);
     }

    }
```



Análisis de tráfico con Wireshark

Se utilizó la herramienta Wireshark, que permite analizar los paquetes TCP y evaluar la calidad de la transmisión.



CONCLUSIONES

Navarrete Becerril Sharon Anette:

Implementar TCP en Node.js es eficiente y flexible: Node.js, con su módulo `net`, facilita la creación de aplicaciones de red TCP, permitiendo a los desarrolladores crear servidores y clientes TCP de manera sencilla. Además, proporciona flexibilidad para manejar errores y gestionar múltiples conexiones de manera eficiente.

Bibliografía

Stevens, W. Richard. Programación en red con Unix: La API de Sockets. Addison-Wesley Professional, 2003.

Forouzan, Behrouz A. Comunicaciones de Datos y Redes. McGraw-Hill, 2012. Tanenbaum, Andrew S., y David J. Wetherall. Redes de Computadoras. Pearson, 2010.

Comer, Douglas E. Internetworking con TCP/IP Volumen Uno. Prentice Hall, 2006. Kurose, James F., y Keith W. Ross. Redes de Computadoras: Un Enfoque Descendente. Pearson, 2017.

Beazley, David, y Brian K. Jones. Python Cookbook: Recetas para Dominar Python 3. O'Reilly Media, 2013. Begg, A. Sockets TCP/IP en C: Guía Práctica para Programadores. Morgan Kaufmann, 2000