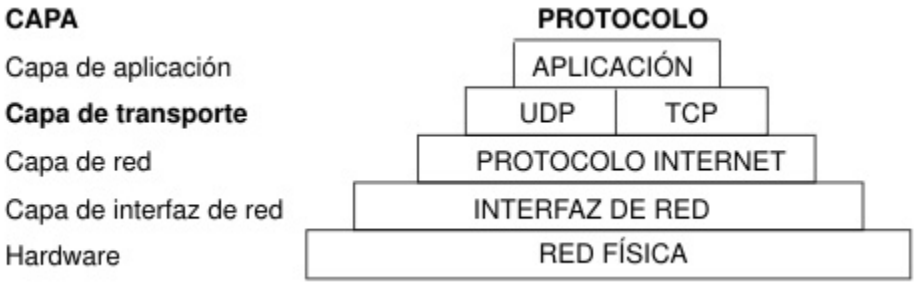
Servicio UDP de la capa de transporte

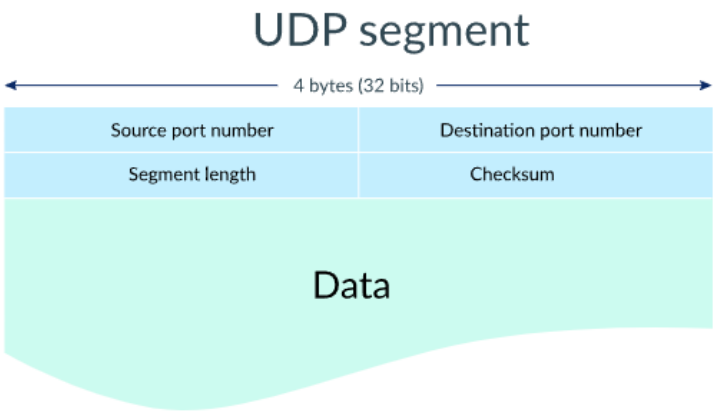
La capa de transporte es la cuarta capa del modelo de referencia OSI (Interconexión de Sistemas Abiertos). Su principal función es proporcionar una transferencia de datos confiable y eficiente entre dos dispositivos en una red. Esta capa se encarga de segmentar los datos recibidos de la capa de sesión en porciones manejables, y reensamblar estos datos en el destino. La capa de transporte también ofrece servicios de control de errores, control de flujo y, en algunos casos, control de congestión.



UDP (User Datagram Protocol) es un protocolo de la capa de transporte que se utiliza para la transmisión de datos. Es uno de los principales protocolos de la capa de transporte en el modelo TCP/IP, junto con TCP (Transmission Control Protocol). UDP es un protocolo simple, sin conexión y no fiable, lo que significa que no establece una conexión antes de enviar datos, no garantiza la entrega de los datos, y no asegura el orden de los paquetes.

UDP proporciona un mecanismo para detectar datos corruptos en paquetes, pero no intenta resolver otros problemas que surgen con paquetes, como cuando se pierden o llegan fuera de orden. Por eso, a veces UDP es conocido como el protocolo de datos no confiable.

UDP es simple pero rápido, al menos en comparación con otros protocolos que funcionan sobre IP. A menudo se usa para aplicaciones sensibles al tiempo (como "streaming" de vídeo en tiempo real) donde la velocidad es más importante que la precisión.



Desarrollo:

Para esta práctica se desarrollara un cliente y un servidor al que se le pasará un mensaje estático a través del protocolo UDP y que el servidor lo pueda escuchar y recibir.

Inicialmente se debe de instalar Node JS en tu ordenador de la siguiente manera:

Comenzaremos por instalar las dependencias necesarias para que se pueda crear el proyecto, para ello se puede utilizar un entorno de desarrollo libre como es el caso de Visual Studio Code o el que sea de la preferencia del lector.

Comenzaremos por descargar node.js

Una vez descargado creamos un proyecto en Visual Studio Code, basta con crear una carpeta y dos scripts con extensión .js para esta práctica se nombraron de la siguiente manera:

- P_udp_cliente.js
- P_udp_servidor.js

Comenzaremos por comprender que conforma el servidor:

Para ello se ocupará el módulo dgram que es parte de la biblioteca estándar de Node.js y que sus funcionalidades principales están orientadas a crear y procesar datos.

El módulo dgram de Node.js proporciona funcionalidades para crear y manejar sockets UDP, incluyendo eventos como 'message' y 'error', métodos para enviar y recibir datos, y soporte para configuraciones multicast. Permite gestionar la conexión, configuración de buffers, TTL, y la membresía en grupos multicast. Puedes ver más sobre este módulo en:

https://nodejs.org/api/dgram.html

Ahora comenzaremos definiendo la lógica que implica al servidor:

Primero, se importa el módulo dgram para utilizar las funcionalidades de sockets UDP. Luego, se definen el puerto y la dirección IP en las constantes PORT y HOST. A continuación, se crea un socket de tipo udp4 para IPv4. El servidor se configura para escuchar en el puerto y dirección especificados, y se definen manejadores de eventos para cuando el servidor comienza a escuchar ('listening') y para cuando recibe mensajes ('message'). Finalmente, se vincula el servidor al puerto y dirección especificados para empezar a recibir mensajes UDP.

A continuación, tenemos el siguiente código para P_udp_servidor.js:

```
const dgram = require('dgram');
const PORT = 8080;
const HOST = '127.0.0.1';

const servidor = dgram.createSocket('udp4');

servidor.on('listening', () => {
    console.log(`Servidor UDP escuchando en ${HOST}:${PORT}`);
});

servidor.on('message', (mensaje, rinfo) => {
    console.log(`El servidor dice: ${mensaje} de
${rinfo.address}:${rinfo.port}`);
});

servidor.bind(PORT, HOST);
```

Después establecemos la lógica del cliente que va a mandar el mensaje estático:

Primero, se importa el módulo dgram para utilizar las funcionalidades de sockets UDP. Luego, se definen el puerto y la dirección IP en las constantes PORT y HOST. Se utiliza setInterval para ejecutar una función cada segundo. Dentro de esta función, se crea un socket de tipo udp4 para IPv4, y se envía un mensaje 'Mando paquete UDP' al puerto y dirección especificados. Si hay un error durante el envío, se lanza una excepción. Si el envío es exitoso, se imprime un mensaje en la consola y se cierra el socket.

A continuación, tenemos el siguiente código para P_udp_cliente.js:

```
const dgram = require('dgram');
const PORT = 8080;
const HOST = '127.0.0.1';

setInterval(function () {
    const cliente = dgram.createSocket('udp4');
    cliente.send('Mando paquete UDP', PORT, HOST, (err) => {
        if (err) {
            throw err;
        }
        console.log('Mensaje UDP enviado');
        cliente.close();
    });
}, 1000);
```

Ahora se ejecutara en dos terminales que se encuentren en la ruta donde están su cliente y su servidor de la siguiente manera:

```
\Aplicaciones Comunicaciones en red\UDP> node .\P_udp_servidor.js
```

```
\Aplicaciones Comunicaciones en red\UDP> node .\P_udp_cliente.js
```

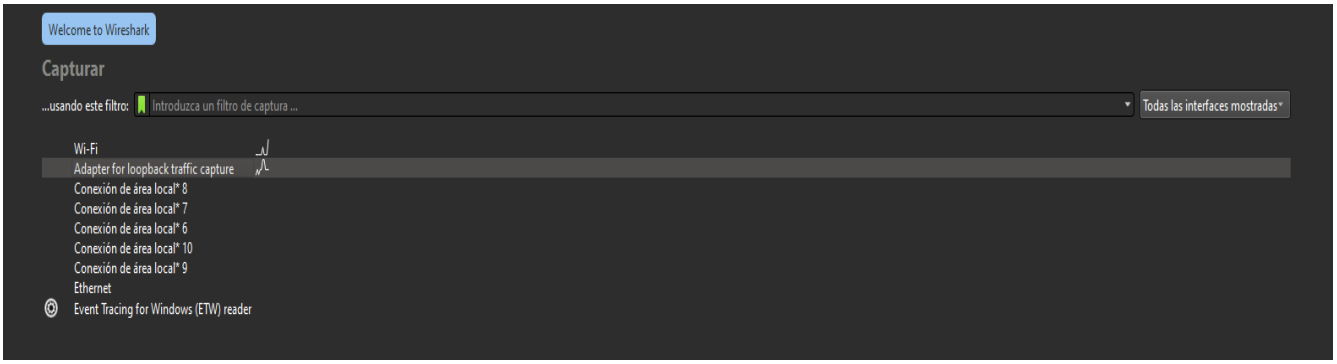
Ahora se comenzará a ejecutar ambos programas y se podrán visualizar la entrega de paquetes entre cliente y servidor mostrando cada uno su respectivo mensaje como se muestra a continuación:

```
Servidor UDP escuchando en 127.0.0.1:8080
El servidor dice: Mando paquete UDP de 127.0.0.1:52128
El servidor dice: Mando paquete UDP de 127.0.0.1:52129
El servidor dice: Mando paquete UDP de 127.0.0.1:52130
El servidor dice: Mando paquete UDP de 127.0.0.1:52131
El servidor dice: Mando paquete UDP de 127.0.0.1:52132
El servidor dice: Mando paquete UDP de 127.0.0.1:52133
El servidor dice: Mando paquete UDP de 127.0.0.1:52134
El servidor dice: Mando paquete UDP de 127.0.0.1:52135
El servidor dice: Mando paquete UDP de 127.0.0.1:52136
El servidor dice: Mando paquete UDP de 127.0.0.1:52137
El servidor dice: Mando paquete UDP de 127.0.0.1:52138
El servidor dice: Mando paquete UDP de 127.0.0.1:52139
El servidor dice: Mando paquete UDP de 127.0.0.1:52140
El servidor dice: Mando paquete UDP de 127.0.0.1:52141
El servidor dice: Mando paquete UDP de 127.0.0.1:52142
El servidor dice: Mando paquete UDP de 127.0.0.1:52143
El servidor dice: Mando paquete UDP de 127.0.0.1:52144
El servidor dice: Mando paquete UDP de 127.0.0.1:52145
```

```
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
Mensaje UDP enviado
```

También se puede hacer uso de wireshark para ver la recepción de paquetes, esta herramienta es opcional pero establece buenas prácticas, puedes decargarla de la siguiente liga: https://www.wireshark.org/download.html

Una vez instalada debes iniciarla en modo administrador y seleccionar la opción de adaptador de loopback



Una vez seleccionada podrás observar una gran cantidad de tráfico en la interfaz seleccionada pero se debe de filtrar para observar los paquetes de datos que se requieren:

udp.port == 8080

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	UDP	49	55719 → 8080 Len=17
2	1.004657	127.0.0.1	127.0.0.1	UDP	49	55720 → 8080 Len=17
3	2.017698	127.0.0.1	127.0.0.1	UDP	49	55721 → 8080 Len=17
4	3.032574	127.0.0.1	127.0.0.1	UDP	49	55722 → 8080 Len=17
5	3.375870	127.0.0.1	127.0.0.1	TCP	45	60166 → 60187 [ACK] Seq=1 Ack=1 Win=10231 Len=1
6	3.375905	127.0.0.1	127.0.0.1	TCP	56	60187 → 60166 [ACK] Seq=1 Ack=2 Win=10232 Len=0 SLE=1 SRE=2
7	4.044922	127.0.0.1	127.0.0.1	UDP	49	55724 → 8080 Len=17
8	5.055890	127.0.0.1	127.0.0.1	UDP	49	55725 → 8080 Len=17
9	6.066385	127.0.0.1	127.0.0.1	UDP	49	55726 → 8080 Len=17
10	7.069280	127.0.0.1	127.0.0.1	UDP	49	55729 → 8080 Len=17
11	8.070331	127.0.0.1	127.0.0.1	UDP	49	55730 → 8080 Len=17
12	9.069118	127.0.0.1	127.0.0.1	TCP	45	60188 → 60164 [ACK] Seq=1 Ack=1 Win=10110 Len=1

Esta herramienta te permitirá ver información detallada sobre los paquetes de datos que se están mandando como se muestra a continuación:

La data que es nuestro mensaje inicialmente se muestra en formato hexadecimal y del lado derecho en un formato de cadena.