



Instituto Politécnico Nacional

IPN

Escuela Superior de Cómputo

ESCOM

6CV2

Práctica 6

“Servicio UDP de la capa de transporte”

Academia de Redes Aplicaciones para comunicaciones
en red

Alumna:

Navarrete Becerril Sharon Anette

Fecha de entrega: “22-Octubre-2024”

Profesor: Ojeda Santillan Rodrigo

OBJETIVO

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.

INTRODUCCIÓN

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 desarrollará 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

Código del servidor:

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

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

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

servidor.on('message', (mensaje, rinfo) => {
```

```

console.log(`Mensaje recibido: ${mensaje} de ${rinfo.address}:${rinfo.port}`);

// Responder al cliente (opcional)
const respuesta = Buffer.from('Mensaje recibido correctamente');
servidor.send(respuesta, rinfo.port, rinfo.address, (err) => {
  if (err) {
    console.error('Error al enviar la respuesta:', err);
  } else {
    console.log('Respuesta enviada al cliente');
  }
});

// Manejador de errores
servidor.on('error', (err) => {
  console.error(`Error en el servidor: ${err.stack}`);
  servidor.close();
});

servidor.bind(PORT, HOST);

```

```

C:\Windows\System32\cmd.e  X  +  v
C:\Users\israe\Desktop\ADR\PracticasRedes2\Practica6>node P_udp_servidor.js
Servidor UDP escuchando en 127.0.0.1:8080
Mensaje recibido: Mando paquete UDP de 127.0.0.1:60205
Respuesta enviada al cliente
Mensaje recibido: Mando paquete UDP de 127.0.0.1:63330
Respuesta enviada al cliente
Mensaje recibido: Mando paquete UDP de 127.0.0.1:63331
Respuesta enviada al cliente
Mensaje recibido: Mando paquete UDP de 127.0.0.1:63332
Respuesta enviada al cliente
Mensaje recibido: Mando paquete UDP de 127.0.0.1:63333
Respuesta enviada al cliente
Mensaje recibido: Mando paquete UDP de 127.0.0.1:63334
Respuesta enviada al cliente
Mensaje recibido: Mando paquete UDP de 127.0.0.1:63335
Respuesta enviada al cliente
Mensaje recibido: Mando paquete UDP de 127.0.0.1:63336
Respuesta enviada al cliente
Mensaje recibido: Mando paquete UDP de 127.0.0.1:63337
Respuesta enviada al cliente
Mensaje recibido: Mando paquete UDP de 127.0.0.1:51083
Respuesta enviada al cliente
Mensaje recibido: Mando paquete UDP de 127.0.0.1:51084
Respuesta enviada al cliente
Mensaje recibido: Mando paquete UDP de 127.0.0.1:51085
Respuesta enviada al cliente
Mensaje recibido: Mando paquete UDP de 127.0.0.1:51086
Respuesta enviada al cliente
^C
C:\Users\israe\Desktop\ADR\PracticasRedes2\Practica6>

```

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

setInterval(() => {
  const cliente = dgram.createSocket('udp4');

  // Enviar mensaje al servidor
  const mensaje = Buffer.from('Mando paquete UDP');
  cliente.send(mensaje, PORT, HOST, (err) => {
    if (err) {
      console.error('Error al enviar el mensaje:', err);
    } else {
      console.log('Mensaje UDP enviado');
    }
  });

  // Escuchar respuesta del servidor
  cliente.on('message', (respuesta, rinfo) => {
    console.log(`Respuesta del servidor: ${respuesta} de ${rinfo.address}:${rinfo.port}`);
    cliente.close();
  });

  // Manejador de errores en el cliente
  cliente.on('error', (err) => {
    console.error('Error en el cliente:', err);
    cliente.close();
  });
}, 1000); // Envía un mensaje cada segundo
```


CONCLUSIONES

Colin Garcia Beatriz Adriana:

Implementación práctica del modelo cliente-servidor: La práctica descrita en el documento enseña a configurar un sistema cliente-servidor utilizando UDP en Node.js. A través de la creación de scripts simples para enviar y recibir mensajes, se demuestra la facilidad con la que se puede implementar comunicación en red utilizando este protocolo, haciendo énfasis en su capacidad para manejar eventos y errores

Mendoza Munguia Braulio Adriel:

Simplicidad y eficiencia del protocolo UDP: El protocolo UDP es una opción ideal para aplicaciones donde la rapidez es prioritaria sobre la confiabilidad, como en transmisiones en tiempo real. Su diseño sin conexión y sin mecanismos de corrección de errores lo hace más rápido que otros protocolos como TCP, aunque a costa de no garantizar la entrega o el orden de los paquetes

Navarrete Becerril Sharon Anette:

Uso de herramientas de monitoreo como Wireshark: La práctica nos enseña el uso de herramientas como Wireshark para observar y analizar el tráfico de red generado por las comunicaciones UDP. Esto no solo ayuda a entender cómo funcionan los paquetes a nivel técnico, sino que también fomenta buenas prácticas en el análisis y depuración de la red

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