UD4 - 01. Protocolos estándar de comunicación en red a nivel de aplicación (telnet, ftp, http, pop3, smtp, entre otros).

Principales Protocolos de Aplicación

1. Telnet (Telecommunication Network)

- **Función**: Permite conectarse a un dispositivo remoto para controlarlo desde una consola de comandos.
- **Puerto estándar**: 23.

Ejemplo en Java:

```
import java.io.*;
import java.net.Socket;

public class TelnetClient {
    public static void main(String[] args) throws IOException {
        String servidor = "servidor-ejemplo.com";
        int puerto = 23;

        Socket socket = new Socket(servidor, puerto);
        BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        BufferedWriter writer = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));

        writer.write("Comando de prueba\n");
        writer.flush();
        String respuesta;
        while ((respuesta = reader.readLine()) != null) {
            System.out.println(respuesta);
```

```
        }
        socket.close();
    }
}
```

---

2. FTP (File Transfer Protocol)

- **Función**: Protocolo para transferir archivos entre cliente y servidor.
- **Puertos estándar**: 20 y 21.

Ejemplo en Java utilizando Apache Commons Net:

```
import org.apache.commons.net.ftp.FTPClient;

public class FTPExample {
    public static void main(String[] args) {
        FTPClient client = new FTPClient();
        try {
            client.connect("ftp.servidor.com");
            client.login("usuario", "contraseña");

            client.enterLocalPassiveMode();
            client.storeFile("/directorio/archivo.txt", new
FileInputStream("archivo_local.txt"));

            client.logout();
            client.disconnect();
            System.out.println("Archivo subido con éxito.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

---

3. HTTP (Hypertext Transfer Protocol)

- **Función**: Permite la comunicación entre navegadores y servidores web para cargar páginas web.
- **Puertos estándar**: 80 (HTTP), 443 (HTTPS).

Ejemplo en Java con `HttpURLConnection`:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class HttpExample {
    public static void main(String[] args) {
        try {
            URL url = new URL("http://www.example.com");
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setRequestMethod("GET");

            BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
            reader.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

-

---

4. HTTPS (Hypertext Transfer Protocol Secure)

- **Función**: Variante segura de HTTP con cifrado SSL/TLS para proteger la información.
- **Puerto estándar**: 443.
- **Nota**: El ejemplo anterior también puede adaptarse para HTTPS utilizando certificados de seguridad.

5. SMTP (Simple Mail Transfer Protocol)

- **Función**: Protocolo para enviar correos electrónicos.
- **Puertos estándar**: 25, 587 (TLS).

Ejemplo en Java con `javax.mail`:

```
import javax.mail.*;
import javax.mail.internet.*;
import java.util.Properties;

public class SendEmail {
    public static void main(String[] args) {
        String host = "smtp.ejemplo.com";
        String from = "remitente@ejemplo.com";
        String to = "destinatario@ejemplo.com";
        String username = "usuario";
        String password = "contraseña";

        Properties props = new Properties();
        props.put("mail.smtp.host", host);
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");

        Session session = Session.getInstance(props, new
Authenticator() {
            protected PasswordAuthentication
getPasswordAuthentication() {
                return new PasswordAuthentication(username,
password);
            }
        });

        try {
            Message message = new MimeMessage(session);
            message.setFrom(new InternetAddress(from));
            message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(to));
            message.setSubject("Prueba SMTP");
            message.setText("Este es un mensaje de prueba.");
```

```
            Transport.send(message);
            System.out.println("Correo enviado con éxito.");
        } catch (MessagingException e) {
            e.printStackTrace();
        }
    }
}
```

-

6. POP3 (Post Office Protocol v3)

- **Función**: Protocolo para descargar correos electrónicos al cliente desde el servidor.
- **Puertos estándar**: 110 (sin cifrado), 995 (cifrado SSL/TLS).
- **Ejemplo**: Para implementar POP3, se utiliza `JavaMail` con configuración específica del servidor POP3.

7. IMAP (Internet Message Access Protocol)

- **Función**: Protocolo para gestionar correos electrónicos directamente en el servidor, permitiendo acceso desde múltiples dispositivos.
- **Puertos estándar**: 143 (sin cifrado), 993 (SSL/TLS).

Conclusión

Los protocolos de aplicación son fundamentales para el funcionamiento de servicios en red como transferencia de archivos, correo electrónico y navegación web. Con Java, se pueden desarrollar aplicaciones que interactúan con estos protocolos mediante librerías especializadas. Implementar estos servicios de manera segura es crucial para proteger la integridad y la privacidad de los datos.