INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL
INTERDISCIPLINARIA DE INGENIERÍAS
ZACATECAS

Práctica 1

Aplicaciones para Comunicación en Red

Ramiro Estrada García

Contenido

Introducción

Una red de comunicaciones es un conjunto de medios técnicos que permiten la comunicación a distancia entre equipos autónomos. Normalmente se trata de transmitir datos, audio y vídeo. La información se puede transmitir de forma analógica, digital o mixta, pero en cualquier caso las conversiones, si las hay, siempre se realizan de forma transparente al usuario, el cual maneja la información de forma analógica exclusivamente.

Objetivos

Construcción de un servicio de transferencia de archivos utilizando sockets orientados a conexión.

Desarrollo

Se usó de referencia el código hecho en clase, pero al usar Kotlin tuve que cambiar el código fuente un poco. La ventaja de usar Kotlin es que corre en la máquina Virtual de Java por lo que puedo usar sus librerías, pero con todas las ventajas que provee Kotlin.

Servidor

```
import java.io.DataInputStream
import java.io.DataOutputStream
import java.io.File
import java.net.ServerSocket
import kotlin.math.min


class ServerTCP {
    init {
        println("Iniciando servidor")
    }

    private val server = ServerSocket(1025)
    private val client = server.accept()
    private val output = DataOutputStream(client.getOutputStream())
    private val input = DataInputStream(client.getInputStream())

    init {
        val file = receiveFile("pruebaServidor.txt")
        file.createNewFile()
        output.write(200)
        closeServer()
    }

    private fun closeServer(){
        input.close()
        output.close()
        client.close()
```

```
        println("Cerrando server")
    }

    private fun receiveFile(fileName: String) : File {
        // Obtenemos el tamaño del archivo
        var fileSize = input.readLong()

        // Definimos el tamaño del buffer
        val buffer = ByteArray(16*1024)

        // Escribimos los datos en un archivo
        return File(fileName).also {
            val fileOutput = it.outputStream()

            var bytes = input.read(buffer, 0, min(buffer.size.toLong(),
fileSize).toInt())
            while (fileSize > 0 && bytes != -1){
                fileOutput.write(buffer, 0, bytes)
                fileSize -= bytes
                bytes = input.read(buffer, 0, min(buffer.size.toLong(),
fileSize).toInt())
            }
        }
    }
}

fun main(){
    ServerTCP()
}
```

Cliente

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

class ClientTCP {
    init {
        println("Iniciando cliente")
    }

    private val client = Socket("127.0.0.1", 1025)
    private val output = DataOutputStream(client.getOutputStream())
    private val input = DataInputStream(client.getInputStream())

    init {
        sendFile(File("prueba.txt"))
        if(input.read() == 200){
            closeClient()
        }
    }

    private fun closeClient(){
        input.close()
        output.close()
        println("Cerrando cliente")
    }
```

```
    private fun sendFile(file: File){
        // Mandamos el tamaño del archivo
        output.writeLong(file.length())

        // Creamos un Stream para archivos
        val fileStream = FileInputStream(file)

        // Definimos el tamaño del Buffer
        val buffer = ByteArray(16*1024)

        // Escribimos los datos del archivo en el stream del socket
        var bytes = fileStream.read(buffer)
        while (bytes != -1){
            output.write(buffer, 0, bytes)
            output.flush()
            bytes = fileStream.read(buffer)
        }

        fileStream.close()
    }
}

fun main(){
    ClientTCP()
}
```

Conclusiones

Podemos enviar archivos de tamaño pequeño enviando un ByteArray pero a la hora de trabajar con archivos más grandes no es posible mandar toda la información, por eso se tuvo que enviar la información en partes más pequeñas.

Referencias

[1]I. University, "What is a network operating system (NOS)?", *Knowledge Base*, 2021. [Online]. Available: https://kb.iu.edu/d/ajlq. [Accessed: 26- Feb- 2021].

[2]F. Björn, *Distributed Systems*. Edinburgh: University of Edinburgh, 2021.

[3]S. Zimmer, "Multiprocessing Operating Systems (OS)", *Science.jrank.org*. [Online]. Available: https://science.jrank.org/programming/Multiprocessing_Operating_Syst.html. [Accessed: 26- Feb- 2021].