

Universidad de Los Andes

Infraestructura de Comunicaciones – 202220

Laboratorio 3.2 - Implementación de servidor y cliente UDP

Juan Alegría – 202011282

Juan Andrés Romero – 202013449

### **Proceso de solución implementado de los requerimientos**

Se tomó como punto de inicio del desarrollo el sistema de transferencia de archivos TCP utilizado en el laboratorio anterior. En primer lugar, se realizó un mirror del repositorio inicial para comenzar a modificar el protocolo desde ese punto de inicio.

#### **Tecnologías utilizadas:**

Como tecnologías para implementar nuestro sistema de transferencia de archivos UDP se utilizó principalmente Python 3.10 como lenguaje de programación de la aplicación y la librería de Socket, nativa del core de Python para la comunicación entre cliente y servidor.

#### **Desarrollo del Servidor:**

La principal diferencia para configurar el servidor UDP respecto al servidor TCP es el cambio en el tipo de socket de SOCK\_STREAM a SOCK\_DGRAM. Luego de esto, se modificó ligeramente la arquitectura del servidor ya que como la tecnología es UDP, no hay conexiones ni necesidad de manejarlas. Sin embargo, la concurrencia se trabajó de forma similar al servidor TCP pasado, donde se creó un thread por cada cliente que necesita recibir los archivos. De igual manera, el servidor estará listo a responder todas las peticiones de transferencia en el puerto configurado para enviar automáticamente a cada uno de los sub-clientes.

```

# Server Variables
file_to_be_sent = None
self.path = PROJECT_PATH + "/files"
if not "/server" in self.path:
    self.path = PROJECT_PATH + "/server/files"
files = [f for f in os.listdir(self.path) if os.path.isfile(os.path.join(self.path, f))]

# Server config
self.server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
self.server.bind(ADDR)
log.info(f"[PROCESS] Server process ID: {os.getpid()}")
log.info(f"[LISTENING] Server is listening on {IP}:{PORT}")
while True:
    data, addr = self.server.recvfrom(1024)
    data = data.decode(FORMAT)
    log.info(f"[RECEIVED] Received data from client: {data}")
    if "LIST" == data:
        self.server.sendto(str(files).encode(FORMAT), addr)
    elif "CONFIG" in data:
        commands = data.split(":")
        if len(commands) == 1 or len(commands) == 0:
            self.server.sendto("Invalid config format (Empty config). Please use the following format: !CONFIG <file>".encode(FORMAT), addr)
        elif len(commands) != 3:
            msg_str = ""
            for string in commands:
                msg_str += string + " "
            self.server.sendto(f"Invalid config format! {msg_str}. Please use the following format: !CONFIG <file> :<num_clients>".encode(FORMAT), addr)
        elif commands[1].rstrip() not in files:
            self.server.sendto(f"File {commands[1].rstrip()} does not exist".encode(FORMAT), addr)
        else:
            file_to_be_sent = commands[1].rstrip()
            self.server.sendto("Config set".encode(FORMAT), addr)
    elif "TRANSFER" == data:
        thread = threading.Thread(target=self.send_file, args=(file_to_be_sent, addr))
        thread.start()

```

Figura 1. Configuración inicial del servidor y funcionamiento de comandos.

```

def send_file(self, file_to_be_sent, addr):
    with open(self.path + '/' + file_to_be_sent, 'rb') as f:
        log.info(f"[TRANSFER] Sending file {file_to_be_sent} to client {addr}")
        log.info(f"[TRANSFER] File size: {os.path.getsize(self.path + '/' + file_to_be_sent)} bytes. Client {addr}")
        t1 = time.time()
        data = f.read(65507)
        while data:
            if self.server.sendto(data, addr):
                data = f.read(65507)
        t2 = time.time()
    log.info(f"[TRANSFER] File {file_to_be_sent} sent to client {addr}")
    log.info(f"[TRANSFER] Time taken to send file: {t2 - t1} seconds. Client {addr}")

```

Figura 2. Envío de archivos a los clientes.

## Desarrollo del Cliente:

Para el desarrollo del cliente se utilizó una arquitectura similar a la del servidor, se crea una conexión principal y esta le indica al servidor qué archivo enviar y a cuántas conexiones. Una vez se configura y se le da la orden de empezar, el cliente genera X conexiones diferentes que le avisan al servidor que están listas para recibir el archivo (donde X es la cantidad especificada por

el cliente). Del mismo modo que en el servidor, los únicos cambios notables respecto al sistema de transferencia de TCP es la configuración del tipo de socket.

```
def main():
    client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    config = False
    global num_clients
    global ADDR
    IP = input("Enter the server IP: ")
    ADDR = (IP, PORT)
    msg = MENU.split("\n")
    log.info(f"[STARTING] Client is starting...")
    for i in msg:
        log.info(f"[MENU] {i}")

    threads = list()
    connected = True

    while connected:
        msg = input("[MENU] Enter a command: ")
        if not msg in COMMANDS and not "!CONFIG" in msg:
            log.info(f"[MENU] Invalid command {msg}")
            continue
        elif msg == "!LIST":
            client.sendto("LIST".encode(FORMAT), ADDR)
            data, addr = client.recvfrom(1024)
            log.info(f"[SERVER] {data.decode(FORMAT)}")
        elif "!CONFIG" in msg and not config:
            client.sendto(msg.encode(FORMAT), ADDR)
            data, addr = client.recvfrom(1024)
            if data.decode(FORMAT) == "Config set":
                config = True
                num_clients = int(msg.split(":")[2])
                log.info(f"[SERVER] {data.decode(FORMAT)}")
        elif "!CONFIG" in msg:
            log.info("[MENU] Server already configured")
        elif "!DISCONNECT" == msg:
            sys.exit(0)
        elif "!START" == msg:
            for i in range(num_clients):
                client = threading.Thread(target=connect_client, args=(i+1,))
                threads.append(client)
                client.start()
            connected = False
```

Figura 3. Configuración y menú del cliente principal

```

def connect_client(client_num):
    timeout = 5
    client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    log.info(f"[CONNECTED] Client #{client_num} - Connected to {IP}:{PORT}")
    try:
        os.makedirs(PROJECT_PATH+ '/client/ArchivosRecibidos')
    except FileExistsError:
        pass
    client.sendto(f"TRANSFER".encode(FORMAT), ADDR)
    t1 = time.time()
    data, addr = client.recvfrom(65507)
    if data:
        with open(PROJECT_PATH + f"/client/ArchivosRecibidos/Cliente{client_num}-Prueba{num_clients}.mp4", "wb") as f:
            log.info(f"[RECEIVING] Receiving file from server...")
            receiving = True
            f.write(data)
            while receiving:
                receiving = select.select([client], [], [], timeout)[0]
                if receiving:
                    data, addr = client.recvfrom(65507)
                    f.write(data)
                else:
                    f.close()
                    receiving = False
    t2 = time.time()
    file_path = PROJECT_PATH + f'/client/ArchivosRecibidos/Cliente{client_num}-Prueba{num_clients}.mp4'
    time_taken = t2 - t1
    log.info(f"[RECEIVED] Client #{client_num} - File received")
    log.info(f"[TIME] Client #{client_num} - Time elapsed: {t2-t1} seconds")
    log.info(f"[RECEIVED] Client #{client_num} - File: {os.path.getsize(file_path)} bytes")
    log.info(f"[TRANSFER SPEED] Client #{client_num} - Transfer speed: {(os.path.getsize(file_path)/1024**2)/(time_taken)} MB/s")

```

*Figura 4. Creación de subclientes y manejo de transferencias.*

### Puesta en marcha del sistema:

Una vez se haya iniciado el cliente, se le pedirá al usuario la IP del servidor a conectarse, para después de esto mostrar un menú donde podrá seleccionar los comandos de ejecución para listar archivos, configurar el servidor para el envío de archivos específicos dado un número de clientes, o iniciar la transferencia.

```

2022-10-23 20:17:35 INFO    client Enter the server IP:
172.17.0.3
2022-10-23 20:17:45 INFO    client [STARTING] Client is starting...
2022-10-23 20:17:45 INFO    client [MENU] Server Commands:
2022-10-23 20:17:45 INFO    client [MENU] !LIST - List all the available files
2022-10-23 20:17:45 INFO    client [MENU] !CONFIG - Set up the server file transfer configuration
2022-10-23 20:17:45 INFO    client [MENU] !START - Start the file transfer to all clients
2022-10-23 20:17:45 INFO    client [MENU] !DISCONNECT - Disconnect from the server
2022-10-23 20:17:45 INFO    client [MENU]
2022-10-23 20:17:45 INFO    client [MENU] Enter a command:
!LIST
2022-10-23 20:17:47 INFO    client [SERVER] ['100mb.mp4', '250mb.mp4']
2022-10-23 20:17:47 INFO    client [MENU] Enter a command:
!CONFIG :250mb.mp4 :4
2022-10-23 20:17:57 INFO    client [SERVER] Config set
2022-10-23 20:17:57 INFO    client [MENU] Enter a command:
!START
2022-10-23 20:18:04 INFO    client [CONNECTED] Client #1 - Connected to 172.17.0.2:6969
2022-10-23 20:18:04 INFO    client [CONNECTED] Client #2 - Connected to 172.17.0.2:6969
2022-10-23 20:18:04 INFO    client [CONNECTED] Client #3 - Connected to 172.17.0.2:6969
2022-10-23 20:18:04 INFO    client [RECEIVING] Receiving file from server...
2022-10-23 20:18:04 INFO    client [CONNECTED] Client #4 - Connected to 172.17.0.2:6969
2022-10-23 20:18:05 INFO    client [RECEIVING] Receiving file from server...
2022-10-23 20:18:06 INFO    client [RECEIVING] Receiving file from server...
2022-10-23 20:18:06 INFO    client [RECEIVING] Receiving file from server...
2022-10-23 20:18:10 INFO    client [RECEIVED] Client #1 - File received
2022-10-23 20:18:10 INFO    client [TIME] Client #1 - Time elapsed: 5.815106630325317 seconds
2022-10-23 20:18:10 INFO    client [RECEIVED] Client #1 - File: 263776025 bytes
2022-10-23 20:18:10 INFO    client [TRANSFER SPEED] Client #1 - Transfer speed: 43.25912426341183 MB/s
2022-10-23 20:18:11 INFO    client [RECEIVED] Client #2 - File received
2022-10-23 20:18:11 INFO    client [TIME] Client #2 - Time elapsed: 6.326212406158447 seconds
2022-10-23 20:18:11 INFO    client [RECEIVED] Client #2 - File: 221655024 bytes
2022-10-23 20:18:11 INFO    client [TRANSFER SPEED] Client #2 - Transfer speed: 33.41441765145761 MB/s
2022-10-23 20:18:11 INFO    client [RECEIVED] Client #3 - File received
2022-10-23 20:18:11 INFO    client [TIME] Client #3 - Time elapsed: 6.856896638870239 seconds
2022-10-23 20:18:11 INFO    client [RECEIVED] Client #3 - File: 208684638 bytes
2022-10-23 20:18:11 INFO    client [TRANSFER SPEED] Client #3 - Transfer speed: 29.02438084904928 MB/s
2022-10-23 20:18:12 INFO    client [RECEIVED] Client #4 - File received
2022-10-23 20:18:12 INFO    client [TIME] Client #4 - Time elapsed: 7.290682554244995 seconds
2022-10-23 20:18:12 INFO    client [RECEIVED] Client #4 - File: 177261942 bytes
2022-10-23 20:18:12 INFO    client [TRANSFER SPEED] Client #4 - Transfer speed: 23.18715156007787 MB/s

```

*Figura 5. Funcionamiento del cliente*

Por otro lado, el servidor informa por medio de logs acerca de todas las operaciones ejecutadas, entre ellas se encuentran: el inicio del servidor; información de puertos y dirección y puerto donde está funcionando; apertura de nuevas conexiones realizadas ya sea de un cliente principal o de conexiones secundarias; estado de conexiones activas; cambios en la configuración del servidor; desconexiones de clientes; información y estado final de la transferencia de archivos; y finalización del servidor.

```

2022-10-23 20:17:38 INFO UDPServer [STARTING] server is starting...
2022-10-23 20:17:38 INFO UDPServer [MAIN THREAD] Press Enter to exit...
2022-10-23 20:17:38 INFO UDPServer [PROCESS] Server process ID: 8
2022-10-23 20:17:38 INFO UDPServer [LISTENING] Server is listening on 172.17.0.3:6969
2022-10-23 20:17:47 INFO UDPServer [RECEIVED] Received data from client: LIST
2022-10-23 20:17:57 INFO UDPServer [RECEIVED] Received data from client: !CONFIG :250mb.mp4:4
2022-10-23 20:18:04 INFO UDPServer [RECEIVED] Received data from client: TRANSFER
2022-10-23 20:18:04 INFO UDPServer [TRANSFER] Sending file 250mb.mp4 to client
2022-10-23 20:18:04 INFO UDPServer [TRANSFER] File size: 266789347 bytes
2022-10-23 20:18:05 INFO UDPServer [TRANSFER] File 250mb.mp4 sent to client
2022-10-23 20:18:05 INFO UDPServer [TRANSFER] Time taken to send file: 0.8048744201660156 seconds
2022-10-23 20:18:05 INFO UDPServer [RECEIVED] Received data from client: TRANSFER
2022-10-23 20:18:05 INFO UDPServer [TRANSFER] Sending file 250mb.mp4 to client
2022-10-23 20:18:05 INFO UDPServer [TRANSFER] File size: 266789347 bytes
2022-10-23 20:18:06 INFO UDPServer [TRANSFER] File 250mb.mp4 sent to client
2022-10-23 20:18:06 INFO UDPServer [TRANSFER] Time taken to send file: 0.5016963481903076 seconds
2022-10-23 20:18:06 INFO UDPServer [RECEIVED] Received data from client: TRANSFER
2022-10-23 20:18:06 INFO UDPServer [TRANSFER] Sending file 250mb.mp4 to client
2022-10-23 20:18:06 INFO UDPServer [TRANSFER] File size: 266789347 bytes
2022-10-23 20:18:06 INFO UDPServer [TRANSFER] File 250mb.mp4 sent to client
2022-10-23 20:18:06 INFO UDPServer [TRANSFER] Time taken to send file: 0.5521278381347656 seconds
2022-10-23 20:18:06 INFO UDPServer [RECEIVED] Received data from client: TRANSFER
2022-10-23 20:18:06 INFO UDPServer [TRANSFER] Sending file 250mb.mp4 to client
2022-10-23 20:18:06 INFO UDPServer [TRANSFER] File size: 266789347 bytes
2022-10-23 20:18:07 INFO UDPServer [TRANSFER] File 250mb.mp4 sent to client
2022-10-23 20:18:07 INFO UDPServer [TRANSFER] Time taken to send file: 0.4242851734161377 seconds

```

*Figura 6. Funcionamiento del servidor*

### Enlace al repositorio de GitHub

- <https://github.com/communications-infrastructure/simple-udp-transfer-system>

### Enlace al video de explicación de desarrollo, aplicaciones, y resultados

- <https://youtu.be/QRmwivxeAvE>

### Enlace de descarga a capturas de tráfico realizadas durante la práctica

- [https://uniandes-my.sharepoint.com/:f/g/personal/j\\_alegria\\_uniandes\\_edu\\_co/EI5rmDBd4ZFBsVN4X87S34UB3SUwIYldqj5ILeeyTbZmFw?e=Lgnbi5](https://uniandes-my.sharepoint.com/:f/g/personal/j_alegria_uniandes_edu_co/EI5rmDBd4ZFBsVN4X87S34UB3SUwIYldqj5ILeeyTbZmFw?e=Lgnbi5)

### Análisis de las pruebas de desempeño obtenidas del servicio de transferencia

Prueba	1C/100MB	1C/250MB	5C/100MB	5C/250MB	10C/100MB	10C/250MB
Transferencia exitosa	Sí	Sí	Sí	Sí	Sí	Sí
Valor total de bytes recibidos por cada cliente	113563354	261614294	114152917, 114087410, 113694368, 113563354, 113301326	266265291, 266330798, 264955151, 266003263, 265937756	113890889, 114087410, 114021903, 113563354, 114087410,	262400378, 262989941, 262727913, 263645011, 263055448, 261417773, 264300081, 264758630, 262858927, 263645011

<b>Tiempo de la transferencia para cada cliente</b>	20.25s	36.514s	44.52s, 45.47s, 46.92s, 48.09s, 48.71s	100.61s, 102.38s, 104.48s, 105.01s, 105.57s	88.78s, 91.62s, 93.28s, 93.46s, 93.83s, 94.28s, 95.74s, 95.93s, 96.12s, 96.17s	168.73s, 181.53s, 184.80s, 189.62s, 191.98s, 192.74s, 198.09s, 199.65s, 199.96s, 200.30s
<b>Tasa de transferencia a cada cliente</b>	5.3471 MB/s	6.8326 MB/s	2.44 MB/s, 2.39 MB/s, 2.310 MB/s, 2.25 MB/s, 2.12 MB/s	2.52 MB/s, 2.48 MB/s, 2.41 MB/s, 2.41 MB/s, 2.40 MB/s	1.22 MB/s, 1.18 MB/s, 1.16 MB/s, 1.15 MB/s, 1.15 MB/s, 1.15 MB/s, 1.13 MB/s, 1.13 MB/s, 1.13 MB/s, 1.13 MB/s	1.48 MB/s, 1.38 MB/s, 1.35 MB/s, 1.32 MB/s, 1.30 MB/s, 1.29 MB/s, 1.27 MB/s, 1.26 MB/s, 1.25 MB/s, 1.25 MB/s
<b>Numero Puerto utilizado para la conexión de cada cliente (Aplicación Cliente)</b>	55755	51693	55270, 55271, 55272, 55273, 55274	55141, 55142, 55143, 55144, 15145	53554, 53555, 53556, 53557, 53558, 53559, 53560, 53561, 53562, 53563	60437, 60438, 60439, 60440, 60441, 60442, 60443, 60444, 60445, 60446
<b>Valor total de bytes transmitidos por el servidor a cada cliente</b>	114283931	266789347	114283931	266789347	114283931	266789347
<b>Número Puerto utilizado para la conexión con cada cliente (Aplicación Servidor)</b>	6969	6969	6969	6969	6969	6969

**Prueba 1:** Se puede ver que en esta prueba se le está dando el ancho de banda completo al único usuario que está conectado, de esta manera, es capaz de aprovechar al máximo la red y la descarga del archivo.

**Prueba 2:** Como en esta prueba sigue siendo una conexión, se le da el ancho de banda completo a la única conexión, la diferencia en la tasa de transferencia puede deberse a fluctuaciones en la

red o congestión en ella, de igual manera se puede ver que la velocidad es más rápida. Esto puede deberse a que la cantidad de paquetes perdidos fue mucho mayor en esta prueba que en la anterior y no hay encolamiento que cause retraso en la llegada de paquetes.

**Prueba 3:** Al comparar esta prueba con la anterior, se puede ver que las tasas de transferencia disminuyen bastante, esto se debe a que ahora hay 5 conexiones concurrentes recibiendo un archivo al mismo tiempo, con lo cual el ancho de banda de la red es compartido.

**Prueba 4:** Para esta prueba se notó un comportamiento bastante similar al de la prueba anterior, y adicionalmente, también se identificó una pérdida de paquetes mayor con el archivo de 250MB que con el de 100. Esto puede causar que la velocidad de transferencia haya sido ligeramente mayor en esta prueba que en la otra.

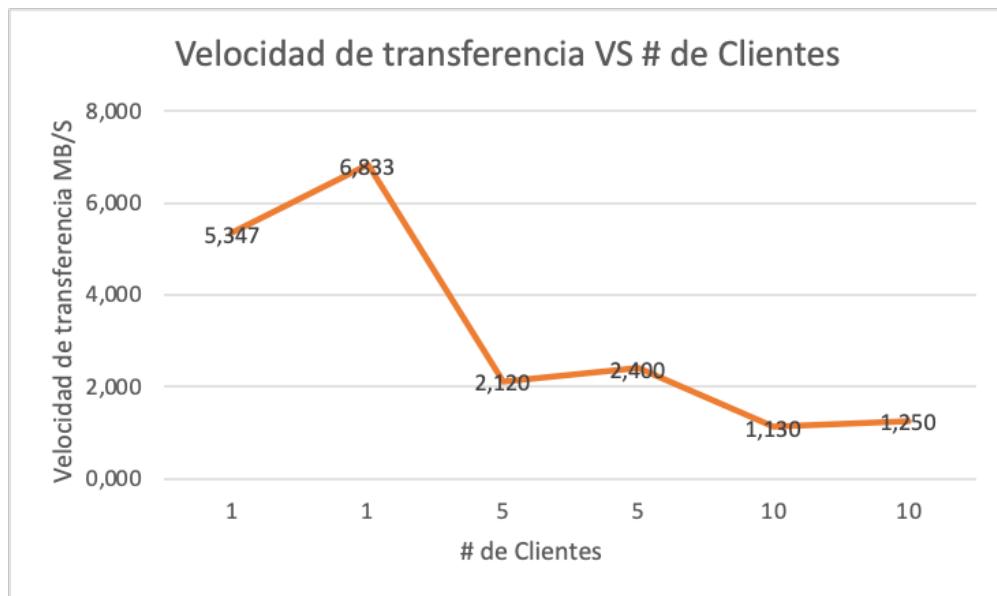
**Prueba 5:** En la prueba 5, se puede evidenciar un cambio en la tasa de transferencia similar debido a la cantidad de conexiones recibiendo paquetes al mismo tiempo. Se puede ver que la disminución fue casi inversamente proporcional al aumento de la cantidad de clientes, pues hay el doble de ellos, pero la velocidad se redujo en casi un 50%

**Prueba 6:** En esta prueba se puede evidenciar un comportamiento bastante similar a las anteriores pruebas con archivos de 250 MB en cuanto a velocidad y porcentaje de pérdida de paquetes. La velocidad es mayor que la de la prueba 5, pero hubo una pérdida mucho mayor. Mientras que en la prueba 5 se llegó a perder como máximo 1 MB, en esta lo que más se perdió fue alrededor de 4 MB

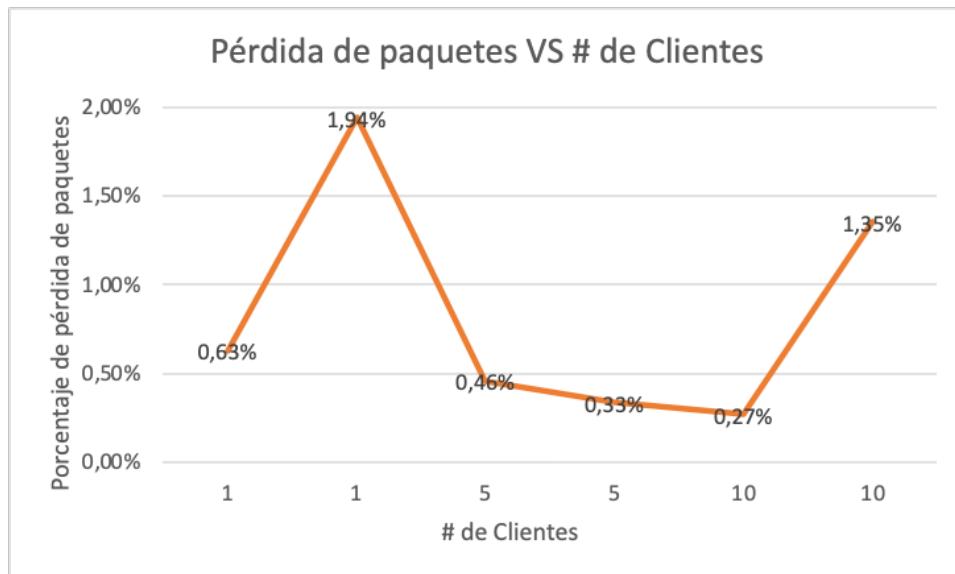
### **Análisis general de las pruebas**

En la anterior tabla se puede ver que a medida que aumenta la cantidad de las conexiones y el tamaño del archivo a enviar, existe una mayor cantidad de tráfico, congestión y pérdida de paquetes en la red, esto causa que las velocidades de transferencia simultáneas disminuyan

sustancialmente y que el tiempo de transferencia de archivos sea mucho mayor que si se tuvieran conexiones secuenciales. Adicionalmente, se identificó que al mandar un archivo mucho más grande existe una mayor pérdida de paquetes en comparación con los envíos de archivos de 100MB.



*Figura 7. Comparación de velocidad de transferencia respecto a clientes activos*



*Figura 8. Comparación tasa de pérdida de paquetes respecto a clientes activos*

## Preguntas

1. Si tuviera que desarrollar un servicio de streaming de video con una arquitectura Cliente/Servidor donde la transmisión fuera en multidifusión. Mencione cuales serían las consideraciones técnicas que tendría en cuenta para el desarrollo del servicio, sea lo más detallado posible.

R// Ya que en la transmisión por UDP habrá pérdida de paquetes se debería considerar manejar desde la capa de aplicación un sistema de retransmisión de paquetes, buffer de carga, relleno de bytes para simular pixeles sin transmitir, o manejar varias versiones de calidad de video para transmitir según la capacidad de cada cliente y que se pueda visualizar el video con la mayor coherencia posible, en términos de usuario. Además, se debería considerar enviar contenido por adelantado para que todas las anteriores operaciones se puedan ejecutar sin que sea notable desde la aplicación. En términos de arquitectura, se debería tener en cuenta un servidor de almacenamiento de los videos en múltiples calidades y formatos de video, un servidor para realizar multicast a un grupo de clientes, y cada uno de los clientes que maneja la correcta recepción de video desde la aplicación.

2. ¿Es posible desarrollar aplicaciones UDP que garanticen la entrega confiable de archivos? Que consideraciones deben tenerse en cuenta para garantizar un servicio de entrega confiable utilizando dicho protocolo. Justifique su respuesta.

R// Por definición, UDP no es confiable, para agregarlo debemos integrar las siguientes características:

1. Tiempo de espera y retransmisión para manejar datagramas descartados.

2. Checksum para que el cliente pueda verificar que una respuesta sea para la solicitud apropiada.

De este modo, nuestra aplicación informará si se recibió exactamente el archivo enviado sin modificaciones por medio del checksum y evitara que el cliente descarte paquetes, por aspectos como buffer lleno, al tener tiempo de espera. Sin embargo, al integrar estas características, se pierde la rapidez propia del protocolo UDP y se convierte TCP en un protocolo más apropiado para estas ocasiones.

3. ¿Se podrían considerar servicios de emisión de contenidos que funcionen con el protocolo TCP? Justifique su respuesta.

R// Sí, de hecho, YouTube funciona en su mayoría con el protocolo TCP, ya que es más apto para streaming de video on-demand al prevenir problemas de audio y video por medio de la retransmisión automática de paquetes perdidos. Mientras que el protocolo UDP es más apropiado para servicios de transmisión en tiempo real como Twitch, Skype o Zoom, donde se tolera la pérdida de audio o video dada su funcionalidad. En muchos servicios de emisión de contenidos existentes en internet, se suele utilizar protocolos como RTMP (Real-Time Messaging Protocol) que garantizan muchos de los requerimientos funcionales que generan la experiencia de usuario.