TP1 - File Transfer

[75.43] Intro. a los Sistemas Distribuidos
Primer cuatrimestre de 2023

Grupo 10

Alumno	Email	Padrón
Pedro Aguilar	paguilar@fi.uba.ar	104221
Victoria Lopez	vlopez@fi.uba.ar	103927
Kevin Mendoza	kmendoza@fi.uba.ar	98038
Ezequiel Vilardo	evilardo@fi.uba.ar	104980
Lautaro Fritz	lfritz@fi.uba.ar	102320

Índice

Introducción

En el presente trabajo práctico se busca implementar una aplicación de red que asegure la transferencia de datos de manera confiable, utilizando como base el protocolo UDP. Para este fin, se utiliza una arquitectura cliente-servidor, en la que cada cliente podrá cargar o descargar archivos de manera concurrente. Las pruebas se realizaron a través de Mininet, posibilitando simular una red aislada en la que varios clientes interactúan con un servidor. Para el desarrollo del mismo se utilizó el lenguaje de programación Python.

Hipótesis y suposiciones

Para la realización del trabajo practico se tomaron las siguientes hipótesis:

- El cliente cuenta con espacio en su disco para almacenar el archivo a descargar.
- El servidor cuenta con espacio en su disco para almacenar el archivo a descargar.
- El servidor no se va a caer, no era necesario un sistema de replicas que asumieran el control en caso de algún fallo del servidor.
- Un servidor puede recibir clientes que quieran conectarse utilizando stop and wait o selective repeat (se configura al levantar el mismo), pero ambos tipos de clientes no pueden conectarse a un mismo servidor.
- El tamaño de paquetes enviados a través de stop and wait es de 1471 bytes.
- El tamaño de paquetes enviados a través de selective repeat es de 1469 bytes.
- Cada cliente conectado puede descargar o cargar únicamente un archivo.
- No se le puede pedir para descargar al servidor un archivo que otro cliente este subiendo en ese mismo momento.
- Si un archivo ya existe en el servidor se reemplaza.
- El tamaño máximo de archivo que se puede subir o bajar es de 1 GB.
- No se permite subir archivos al servidor ni solicitar la descarga de archivos cuyo nombre contenga el caracter #, ya que se usa como separador en los mensajes de protocolo de aplicación.

Implementación

El protocolo de aplicación desarrollado permite levantar un servidor para que múltiples clientes puedan conectarse concurrentemente, y así descargar o cargar archivos a través de la red. Debido a la implementación elegida, cada cliente podrá cargar o descargar un sólo archivo; en el momento en que termina la transferencia el cliente se desconecta, y el servidor seguirá esperando futuros clientes. Los servidores que se levanten deben elegir a través de un flag la variante que utilicen (stop and wait o selective repeat). De esta manera, los clientes sólo podrán conectarse a los servidores de su misma variante del protocolo.

Ejecución

Instalar mininet

Para poder ejecutar el trabajo es necesario tener mininet.

```
sudo apt install mininet
sudo pip install mininet
```

Ejecución servidor

El servidor consta de un sólo comando start-server, que permite iniciar el servidor. Para correrlo, ejecutar en el directorio del archivo start-server.py:

```
$ python3 start-server.py [-h] [-v | -q] (-w | -r) [-H ADDR] [-p PORT] [-s DIRPATH]
```

Pueden utilizarse distintos flags:

- -h , –help permite mostrar el mensaje de ayuda y detalle de los distintos flags.
- -v , –verbose | -q , –quiet muestra detalles extra sobre el envío de archivos.
- -H , –host permite indicar el host donde se quiere levantar el servidor.
- -p , –port permite indicar el puerto donde se quiere levantar el servidor.
- -s , –storage permite indicar el directorio donde se quieren bajar los archivos.
- -w, –saw | -r, –sr permite elegir el protocolo de capa de transporte. Este flag es obligatorio.

Se puede frenar la ejecución del servidor en cualquier momento presionando la letra **q**.

Ejecución cliente

El cliente cuenta con dos comandos distintos:

- **upload.py**: permite subir un archivo al servidor.
- **download.py**: permite descargar un archivo del servidor.

upload.py
Para correr este módulo, ejecutar en el directorio del archivo upload.py:

```
$ python3 upload.py [-h] [-v | -q] (-w | -r) [-H ADDR] [-p PORT] [-s FILEPATH] [-n FILENAME]
```

Pueden utilizarse distintos flags:

- -h , –help permite mostrar el mensaje de ayuda y detalle de los distintos flags.
- -v , –verbose | -q , –quiet muestra detalles extra sobre el envío de archivos.

- -H , –host permite indicar la dirección IP del servidor.
- -p , –port permite indicar el puerto en el que está escuchando el servidor.
- -s , –src permite indicar la ruta del archivo a enviar.
- -n , –name permite indicar el nombre con el cual el archivo se guardará en el servidor.
- -w, –saw | -r, –sr permite elegir el protocolo de capa de transporte. Este flag es obligatorio.

download.py
Para correr este módulo, ejecutar en el directorio del archivo download.py:

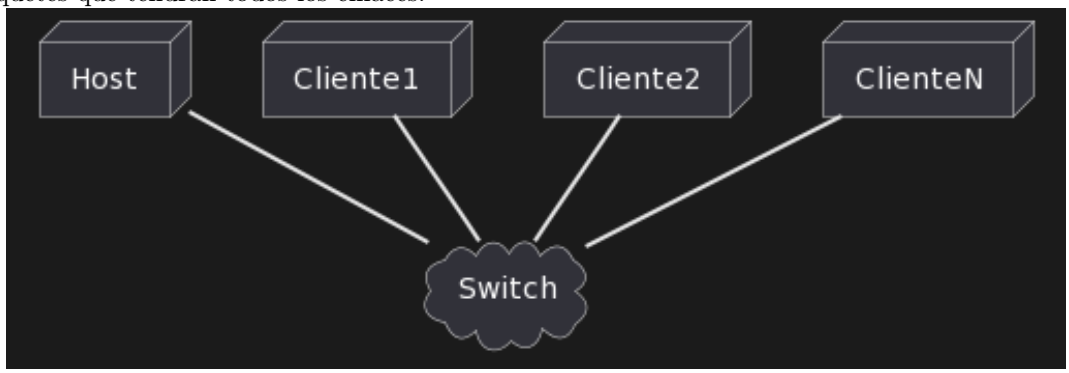
```
$ python3 download.py [-h] [-v | -q] (-w | -r) [-H ADDR] [-p PORT] [-d FILEPATH] [-n FILENAME]
```

Pueden utilizarse distintos flags:

- -h , –help permite mostrar el mensaje de ayuda y detalle de los distintos flags.
- -v , –verbose | -q , –quiet muestra detalles extra sobre el envío de archivos.
- -H , –host permite indicar la dirección IP del servidor.
- -p , –port permite indicar el puerto en el que está escuchando el servidor.
- -d , –dst permite indicar la ruta donde será guardado el archivo descargado.
- -n , –name permite indicar el nombre del archivo alojado en el servidor a descargar.
- -w, –saw | -r, –sr permite elegir el protocolo de capa de transporte. Este flag es obligatorio.

Topología

Para el trabajo se pidió la elaboración de una topología parametrizable sobre la cual probar diferentes funcionalidades. La parametrizacion a través de numero de clientes permite variar la cantidad de clientes diferentes que tendremos en nuestra red, independientes al servidor, que tendremos. Con el parámetro porcentaje de perdida determinaremos el porcentaje de perdida de paquetes que tendrán todos los enlaces.



Podemos ejecutarla mediante:

```
./run_topology.sh numero_de_clientes porcentaje_de_perdida
```

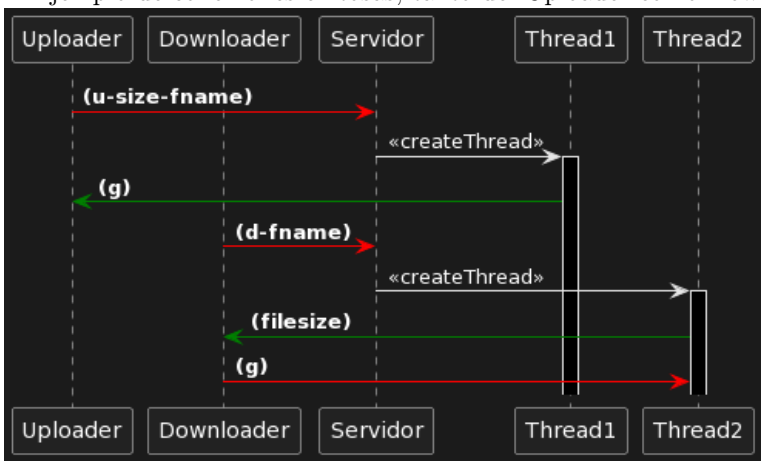
Para el caso de querer 2 clientes y un porcentaje de perdida en los enlaces del 10 por ciento, la linea debería ser:

```
./run_topology.sh 2 10
```

Inicio de conexión

El cliente envía al servidor un mensaje para iniciar la conexión. Este mensaje contiene el nombre y el tamaño del archivo que se quiere cargar o descargar. Entonces, el cliente queda esperando una respuesta del servidor. En el momento en que el servidor reciba uno de estos mensajes de algún cliente, verificará si se trata de una operación de carga (upload) o de descarga (download). Dependiendo de si se trata de una o de otra, enviará una respuesta distinta. En el caso de una carga, el servidor enviará al cliente un mensaje de confirmación, que contiene la letra "G". El cliente, entonces, comenzará la transferencia. En el caso de una descarga, el servidor envía un mensaje con el tamaño del archivo a descargar. Una vez recibido por el cliente, este le envía un mensaje de confirmación ("G"). El servidor, entonces comenzará la transferencia.

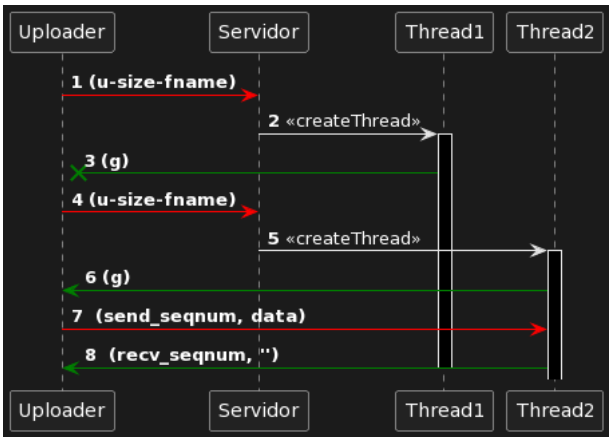
Figura 1: Ejemplo de conexiones exitosas, tanto del Uploader como Downloader



Si no llega, vuelve a mandar el mensaje. Este proceso puede durar hasta que el cliente reciba una respuesta o hasta que se hayan realizado 5 intentos de envío.

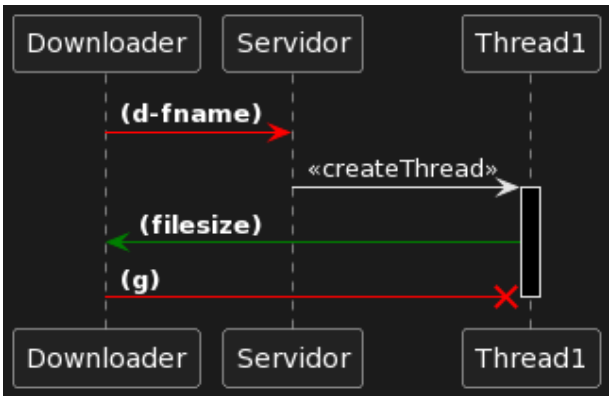
Figura 2: Ejemplo de problema al enviar el primer mensaje, donde el cliente especifica la operación

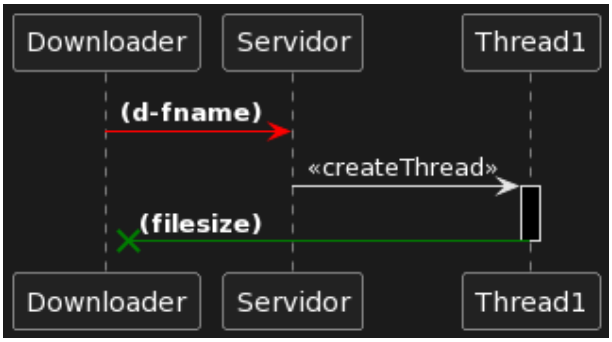
Puede darse la situación donde se pierda la confirmación de subida envía por el servidor



Para solucionar este problema, el cliente reintentara conectarse al servidor, que le abrirá un nuevo thread para atenderlo, el thread antiguo luego de acusar un tiempo (10 segs) sin recibir paquetes, se desbloqueara y procesadara a terminarse.

Ahora en caso de tratarse de un cliente del tipo descarga, si se perdiese el filesize enviado desde el servidor o la confirmación de descarga enviada por el cliente, ambos quedaran esperando momentáneamente hasta que los liberemos luego de un tiemout.





Stop and wait

Mediante este protocolo el archivo es transferido a través de paquetes de tamaño máximo de 1471 bytes. Esta transferencia se realiza de manera tal que por cada paquete enviado, se espera un mensaje de confirmación (ack) antes de enviar el siguiente.

Figura 3: Ejemplo de subida exitosa con Stop and Wait

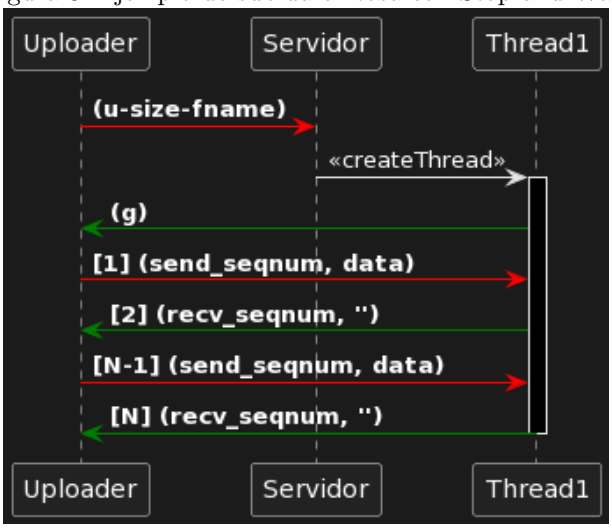
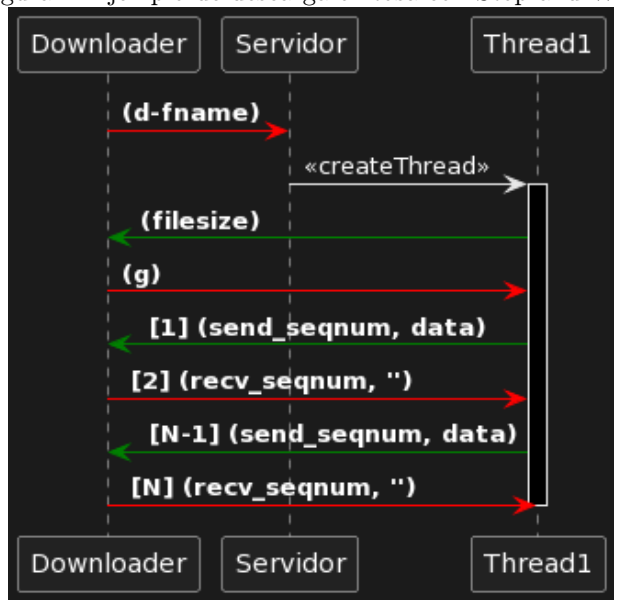
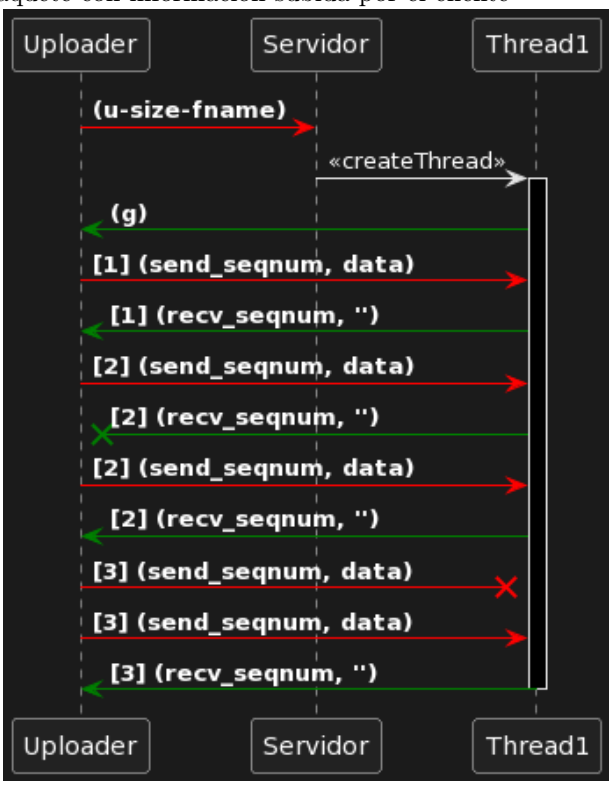


Figura 4: Ejemplo de descarga exitosa con Stop and Wait



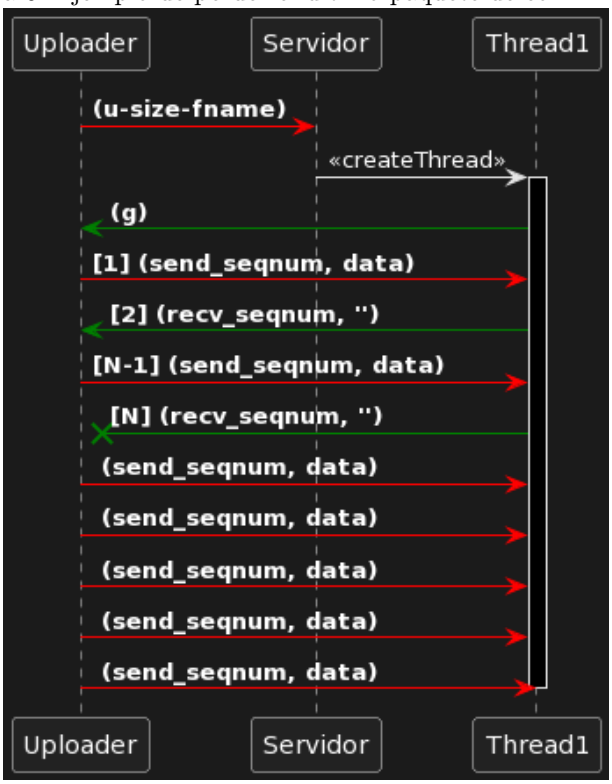
Si el mensaje de confirmación no llega en un tiempo dado, el paquete se vuelve a enviar. A su vez, el remitente tendrá en cuenta el número de secuencia esperado. Este valor se alternará entre 0 y 1 con cada paquete enviado, y servirá para verificar que el mensaje de confirmación corresponda efectivamente con el paquete enviado. Por su parte, el receptor enviará un mensaje de confirmación por cada paquete recibido, y, si posee el número de secuencia correcto, lo escribirá en el archivo.

Figura 5: Ejemplo de perdida de paquete de confirmación por parte del servidor a información subida y perdida de paquete con información subida por el cliente



Si bien la secuencia mostrada en la imagen anterior pertenece al caso de subida, para el caso de descarga es exactamente igual, cambiando de lugar al cliente y al servidor.

Figura 6: Ejemplo de perder el ultimo paquete de confirmación



Puede darse una caso mas que nos pareció importante abordar, y es el caso en que se pierda la ultima confirmación de un envió. Lo que podemos ver en la figura 6, es que el servidor sabe que recibió el ultimo paquete que necesitaba para completar el archivo que le estaban subiendo (aplica a la inversa para el caso de descargas), esto lo sabe porque al principio le pasaron el tamaño del archivo que le van a subir, entonces una vez recibido envía la confirmación y procede a cerrarse, para evitar el caso donde se pierda esta confirmación y el cliente, que también conoce que es su ultimo envió, quedase en un bucle de envió paquete y espera de confirmación, solo reintentara 5 veces mas y terminara.

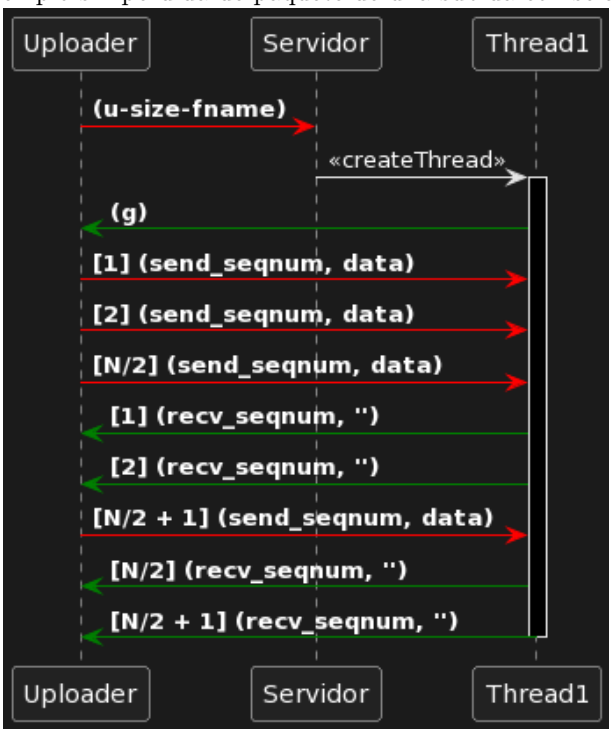
Selective repeat

Mediante este protocolo, la transferencia del archivo se realiza a través de paquetes de tamaño máximo de 1469. A diferencia del stop and wait, esta variante permite enviar múltiples paquetes en simultáneo, sin tener que esperar un mensaje de confirmación por cada uno. El número de paquetes enviados de esta manera se denomina ventana y se define por el siguiente cálculo:

```
cantidad_paquetes/2 - 1
```

A medida que se reciba el mensaje de confirmación del paquete con el menor número de secuencia, se enviará un paquete nuevo, resultando en un desplazamiento de la ventana. Por su parte, el receptor enviará los mensajes de confirmación correspondientes a los paquetes recibidos. Mientras sea el paquete que posea el número de secuencia esperado lo escribirá en el archivo. En caso contrario, lo guardará en un buffer, y esperará un paquete anterior, antes de escribirlo.

Figura 7: Ejemplo sin perdida de paquete de una subida con selective repeat



Ahora también en caso de perdida de paquetes solo serán re-enviados los que no acusen confirmación de recibo todavía.

Figura 8: Ejemplo perdida de paquete confirmación de recibo

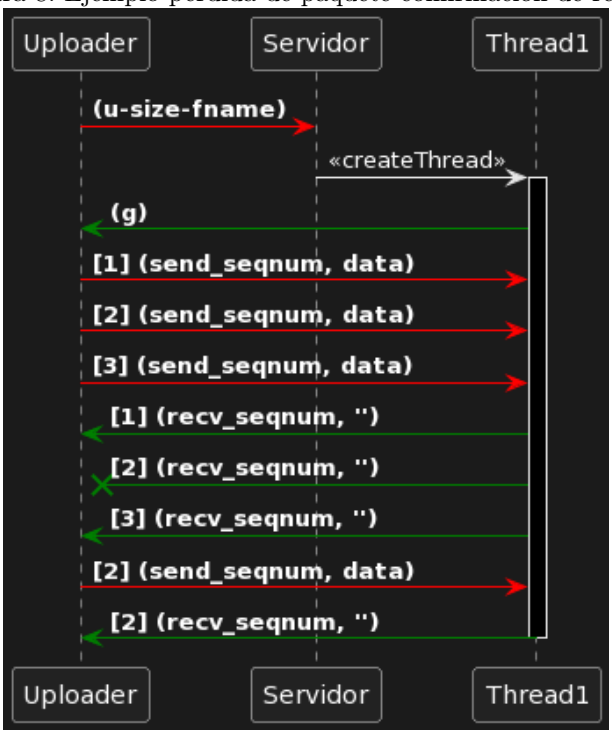
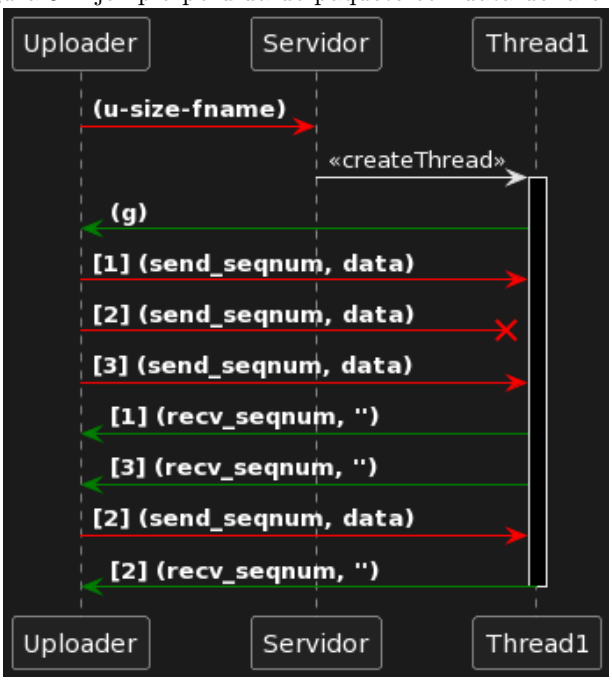


Figura 9: Ejemplo perdida de paquete con data del archivo



Concurrencia

Para que el servidor pueda atender múltiples consultas de clientes de manera concurrente se opto por crear un thread y socket UDP por cada cliente.

- El servidor espera por clientes, una vez alguien se conecta este genera un thread y socket nuevos, luego le delega la responsabilidad al thread de atender a ese cliente. Una vez terminada la tarea el thread termina.
- El socket se crea para poder distinguir de forma sencilla el origen del trafico que entra en el servidor.

Figura 10: Bloque de codigo donde esperamos por conexiones y lanzamos nuevos threads

```
def listen(self):
    self.active = True
    self.log.info('Esperando clientes')
    while self.active:
        self.socket.timeout(5)
        try:
            protocol_message, addr = self.socket.receive()
            self.log.info('Atendiendo cliente', addr)
            client_thread = threading.Thread(target=self.handle_client, args=(protocol_message, addr))
            client_thread.start()
            self.threads.append(client_thread)
        except socket.timeout:
            continue
    for t in self.threads:
        t.join()
```

Pruebas

Prueba de subida con un archivo de 30kB y sin perdida

- Stop And Wait

  Primero es necesario levantar el server en modo Stop And Wait, esto es posible agregando el flag –saw o -w.

Figura 11: Ejemplo de subida con archivo de 30kB con Stop And Wait del lado del servidor

```
vicky@vicky-VirtualBox:~/Escritorio/distribuidostp$ python3 start-server.py -H 127.0.0.1 -p 25000 -v --saw -s .
2023-04-24 09:20:17.271910 - None : Inicio server
2023-04-24 09:20:17.272476 - None : Esperando clientes
2023-04-24 09:20:17.272798 - None : Para finalizar q
2023-04-24 09:20:42.150698 - ('127.0.0.1', 43454) : Atendiendo cliente
2023-04-24 09:20:42.151094 - ('127.0.0.1', 43454) : Enviamos confirmacion
2023-04-24 09:20:42.151994 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.152021 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:20:42.152117 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.152130 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:20:42.152200 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.152212 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:20:42.152341 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.152355 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:20:42.152418 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.152430 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:20:42.152657 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.152764 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:20:42.152882 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.152896 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:20:42.152999 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.153014 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:20:42.153094 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.153108 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:20:42.153191 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.153205 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:20:42.153276 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.153290 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:20:42.153382 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.153396 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:20:42.153468 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.153481 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:20:42.153572 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.153586 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:20:42.153741 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.153757 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:20:42.153853 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.153868 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:20:42.153945 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.153958 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:20:42.154040 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.154054 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:20:42.154125 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.154138 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:20:42.154228 - ('127.0.0.1', 43454) : Recibimos paquete de 1472 bytes
2023-04-24 09:20:42.154242 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:20:42.154318 - ('127.0.0.1', 43454) : Recibimos paquete de 132 bytes
2023-04-24 09:20:42.154331 - ('127.0.0.1', 43454) : Numero de secuencia esperado b'0' y recibido b'0'
q
2023-04-24 09:20:47.157221 - None : Fin server
```

Figura 12: Ejemplo de subida con archivo de 30kB con Stop And Wait del lado del cliente

```
vicky@vicky-VirtualBox:~/Escritorio/distribuidostp$ python3 upload.py -v --saw -H 127.0.0.1 -p 25000 -s ./foto.jpg -n foto30kbsaw.jpg
2023-04-24 09:20:42.150383 - None : Enviamos operacion
2023-04-24 09:20:42.151711 - None : Esperamos confirmacion
2023-04-24 09:20:42.151798 - None : Intento numero: 1
2023-04-24 09:20:42.151871 - None : Recibimos confirmacion
2023-04-24 09:20:42.151895 - None : Enviamos archivo
2023-04-24 09:20:42.151941 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.152087 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.152174 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.152310 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.152394 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.152509 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.152845 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.152963 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.153061 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.153158 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.153246 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.153348 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.153437 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.153539 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.153692 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.153819 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.153913 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.154008 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.154094 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.154195 - ('127.0.0.1', 43127) : Enviamos paquete de 1472 bytes
2023-04-24 09:20:42.154286 - ('127.0.0.1', 43127) : Enviamos paquete de 132 bytes
2023-04-24 09:20:42.154455 - None : Ultimo paquete
```

- Selective Repeat

  Primero es necesario levantar el server en modo Selective Repeat, esto es posible agregando el flag –sr o -r.

Figura 13: Ejemplo de subida con archivo de 30kB con Selective Repeat del lado del servidor

```
vicky@vicky-VirtualBox:~/Escritorio/distribuidostp$ python3 start-server.py -H 127.0.0.1 -p 25000 -v --sr -s .
2023-04-24 09:11:23.559976 - None : Inicio server
2023-04-24 09:11:23.560386 - None : Esperando clientes
2023-04-24 09:11:23.560405 - None : Para finalizar q
2023-04-24 09:11:28.929493 - ('127.0.0.1', 60344) : Atendiendo cliente
2023-04-24 09:11:28.930340 - ('127.0.0.1', 60344) : Enviamos confirmacion
2023-04-24 09:11:28.932422 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.932478 - ('127.0.0.1', 60344) : Numero de secuencia esperado 0 y recibido 0
2023-04-24 09:11:28.932559 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.932570 - ('127.0.0.1', 60344) : Numero de secuencia esperado 1 y recibido 1
2023-04-24 09:11:28.932591 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.932600 - ('127.0.0.1', 60344) : Numero de secuencia esperado 2 y recibido 2
2023-04-24 09:11:28.932751 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.932782 - ('127.0.0.1', 60344) : Numero de secuencia esperado 3 y recibido 3
2023-04-24 09:11:28.932806 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.932815 - ('127.0.0.1', 60344) : Numero de secuencia esperado 4 y recibido 4
2023-04-24 09:11:28.932851 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.932862 - ('127.0.0.1', 60344) : Numero de secuencia esperado 5 y recibido 5
2023-04-24 09:11:28.932882 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.932891 - ('127.0.0.1', 60344) : Numero de secuencia esperado 6 y recibido 6
2023-04-24 09:11:28.932920 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.932930 - ('127.0.0.1', 60344) : Numero de secuencia esperado 7 y recibido 7
2023-04-24 09:11:28.932950 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.932958 - ('127.0.0.1', 60344) : Numero de secuencia esperado 8 y recibido 8
2023-04-24 09:11:28.932983 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.932992 - ('127.0.0.1', 60344) : Numero de secuencia esperado 9 y recibido 9
2023-04-24 09:11:28.933367 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.933384 - ('127.0.0.1', 60344) : Numero de secuencia esperado 10 y recibido 10
2023-04-24 09:11:28.933598 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.933614 - ('127.0.0.1', 60344) : Numero de secuencia esperado 11 y recibido 11
2023-04-24 09:11:28.933688 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.933702 - ('127.0.0.1', 60344) : Numero de secuencia esperado 12 y recibido 12
2023-04-24 09:11:28.933802 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.933817 - ('127.0.0.1', 60344) : Numero de secuencia esperado 13 y recibido 13
2023-04-24 09:11:28.933990 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.934006 - ('127.0.0.1', 60344) : Numero de secuencia esperado 14 y recibido 14
2023-04-24 09:11:28.934182 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.934197 - ('127.0.0.1', 60344) : Numero de secuencia esperado 15 y recibido 15
2023-04-24 09:11:28.934270 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.934283 - ('127.0.0.1', 60344) : Numero de secuencia esperado 16 y recibido 16
2023-04-24 09:11:28.934365 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.934378 - ('127.0.0.1', 60344) : Numero de secuencia esperado 17 y recibido 17
2023-04-24 09:11:28.934453 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.934466 - ('127.0.0.1', 60344) : Numero de secuencia esperado 18 y recibido 18
2023-04-24 09:11:28.934558 - ('127.0.0.1', 60344) : Recibimos paquete de 1472 bytes
2023-04-24 09:11:28.934572 - ('127.0.0.1', 60344) : Numero de secuencia esperado 19 y recibido 19
2023-04-24 09:11:28.934671 - ('127.0.0.1', 60344) : Recibimos paquete de 174 bytes
2023-04-24 09:11:28.934684 - ('127.0.0.1', 60344) : Numero de secuencia esperado 20 y recibido 20
2023-04-24 09:11:33.937061 - None : No obtuve paquete, finalizamos
q
2023-04-24 09:11:58.961948 - None : Fin server
```

Figura 14: Ejemplo de subida con archivo de 30kB con Selective Repeat del lado del cliente

```
vicky@vicky-VirtualBox:~/Escritorio/distribuidostp$ python3 upload.py -v --sr -H 127.0.0.1 -p 25000 -s ./foto.jpg -n foto30kbsr.jpg
2023-04-24 09:11:28.928974 - None : Enviamos operacion
2023-04-24 09:11:28.929370 - None : Esperamos confirmacion
2023-04-24 09:11:28.929415 - None : Intento numero: 1
2023-04-24 09:11:28.930859 - None : Recibimos confirmacion
2023-04-24 09:11:28.930881 - None : Enviamos archivo
2023-04-24 09:11:28.930944 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.930979 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.931003 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.931020 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.931037 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.931055 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.931071 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.931087 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.931105 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.931132 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.933313 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.933538 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.933654 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.933766 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.933950 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.934116 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.934160 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.934250 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.934345 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.934433 - ('127.0.0.1', 33942) : Enviamos paquete de 1469 bytes
2023-04-24 09:11:28.934538 - ('127.0.0.1', 33942) : Enviamos paquete de 171 bytes
2023-04-24 09:11:28.934715 - None : Ultimo paquete transmitido
```

- Comprobación de archivos de subida exitosos:

Para comprobar que los archivos que subimos se hayan subido bien utilizamos el comando md5sum para comprobar que el hash de cada archivo es igual al de los demas.

Figura 15: Ejemplo de comprobación de archivos con md5sum

```
vicky@vicky-VirtualBox:~/Escritorio/distribuidostp$ md5sum foto.jpg foto30kbsaw.jpg foto30kbsr.jpg
ae70a005ba82c8dce3d46ac5b444146d  foto.jpg
ae70a005ba82c8dce3d46ac5b444146d  foto30kbsaw.jpg
ae70a005ba82c8dce3d46ac5b444146d  foto30kbsr.jpg
```

Prueba de bajada con un archivo de 30kB y sin perdida

- Stop And Wait

  Primero es necesario levantar el server en modo Stop And Wait, esto es posible agregando el flag –saw o -w.

Figura 16: Ejemplo de bajada de archivo de 30kB con Stop And Wait del lado del server

```
vicky@vicky-VirtualBox:~/Escritorio/distribuidostp$ python3 start-server.py -H 127.0.0.1 -p 25000 -v --saw -s .
2023-04-24 09:44:59.938779 - None : Inicio server
2023-04-24 09:44:59.939473 - None : Esperando clientes
2023-04-24 09:44:59.939759 - None : Para finalizar q
2023-04-24 09:45:05.125293 - ('127.0.0.1', 57795) : Atendiendo cliente
2023-04-24 09:45:05.135695 - ('127.0.0.1', 57795) : Enviamos longitud
2023-04-24 09:45:05.135884 - None : Esperamos confirmacion
2023-04-24 09:45:05.136165 - None : Recibimos confirmacion
2023-04-24 09:45:05.136296 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.136639 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.136736 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.136813 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.137005 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.137220 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.137454 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.137760 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.138034 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.138274 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.138545 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.138763 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.138999 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.139186 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.139399 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.139610 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.139839 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.140064 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.140268 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.140519 - ('127.0.0.1', 57795) : Enviamos paquete de 1472 bytes
2023-04-24 09:45:05.140812 - ('127.0.0.1', 57795) : Enviamos paquete de 132 bytes
2023-04-24 09:45:05.140919 - None : Ultimo paquete
q
2023-04-24 09:45:30.155652 - None : Fin server
```

Figura 17: Ejemplo de bajada de archivo de 30kB con Stop And Wait del lado del cliente

```
vicky@vicky-VirtualBox:~/Escritorio/distribuidostp$ python3 download.py -v --saw -H 127.0.0.1 -p 25000 -d ./fotobajadasaw.jpg -n foto30kbs
aw.jpg
2023-04-24 09:45:05.124886 - None : Enviamos operacion
2023-04-24 09:45:05.125299 - None : Esperamos longitud del archivo
2023-04-24 09:45:05.125412 - None : Intento numero: 1
2023-04-24 09:45:05.136006 - None : Recibimos longitud
2023-04-24 09:45:05.136119 - ('127.0.0.1', 48798) : Enviamos confirmacion
2023-04-24 09:45:05.136522 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.136573 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:45:05.136687 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.136700 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:45:05.136773 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.136784 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:45:05.136905 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.136953 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:45:05.137087 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.137131 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:45:05.137354 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.137396 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:45:05.137535 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.137589 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:45:05.137969 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.137987 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:45:05.138153 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.138228 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:45:05.138393 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.138468 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:45:05.138626 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.138669 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:45:05.138871 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.138914 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:45:05.139081 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.139124 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:45:05.139286 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.139329 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:45:05.139484 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.139527 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:45:05.139716 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.139789 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:45:05.139943 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.140020 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:45:05.140182 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.140196 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:45:05.140349 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.140392 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'0' y recibido b'0'
2023-04-24 09:45:05.140676 - ('127.0.0.1', 48798) : Recibimos paquete de 1472 bytes
2023-04-24 09:45:05.140735 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'1' y recibido b'1'
2023-04-24 09:45:05.141012 - ('127.0.0.1', 48798) : Recibimos paquete de 132 bytes
2023-04-24 09:45:05.141062 - ('127.0.0.1', 48798) : Numero de secuencia esperado b'0' y recibido b'0'
```

- Selective Repeat

  Primero es necesario levantar el server en modo Selective Repeat, esto es posible agregando el flag –sr o -r.

Figura 18: Ejemplo de bajada de archivo de 30kB con Selective Repeat del lado del server

```
vicky@vicky-VirtualBox:~/Escritorio/distribuidostp$ python3 start-server.py -H 127.0.0.1 -p 25000 -v --sr -s .
2023-04-24 09:54:17.493615 - None : Inicio server
2023-04-24 09:54:17.494580 - None : Esperando clientes
2023-04-24 09:54:17.495717 - None : Para finalizar q
2023-04-24 09:54:22.545437 - ('127.0.0.1', 60313) : Atendiendo cliente
2023-04-24 09:54:22.546174 - ('127.0.0.1', 60313) : Enviamos longitud
2023-04-24 09:54:22.546380 - None : Esperamos confirmacion
2023-04-24 09:54:22.547235 - None : Recibimos confirmacion
2023-04-24 09:54:22.547374 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.547468 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.547518 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.547557 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.547587 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.547614 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.547635 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.547652 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.547679 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.547711 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.549775 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.550072 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.550153 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.550511 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.550560 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.550725 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.550915 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.551037 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.551270 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.551370 - ('127.0.0.1', 60313) : Enviamos paquete de 1469 bytes
2023-04-24 09:54:22.551401 - ('127.0.0.1', 60313) : Enviamos paquete de 171 bytes
2023-04-24 09:54:22.553000 - None : Ultimo paquete transmitido
q
2023-04-24 09:54:47.574781 - None : Fin server
```

Figura 19: Ejemplo de bajada de archivo de 30kB con Selective Repeat del lado del cliente

```
vicky@vicky-VirtualBox:~/Escritorio/distribuidostp$ python3 download.py -v --sr -H 127.0.0.1 -p 25000 -d ./fotobajadasr.jpg -n foto30kbsr.
jpg
2023-04-24 09:54:22.544133 - None : Enviamos operacion
2023-04-24 09:54:22.546489 - None : Esperamos longitud del archivo
2023-04-24 09:54:22.546520 - None : Intento numero: 1
2023-04-24 09:54:22.546547 - None : Recibimos longitud
2023-04-24 09:54:22.547118 - ('127.0.0.1', 34694) : Enviamos confirmacion
2023-04-24 09:54:22.549554 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.549633 - ('127.0.0.1', 34694) : Numero de secuencia esperado 0 y recibido 0
2023-04-24 09:54:22.549740 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.549767 - ('127.0.0.1', 34694) : Numero de secuencia esperado 1 y recibido 1
2023-04-24 09:54:22.549799 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.550063 - ('127.0.0.1', 34694) : Numero de secuencia esperado 2 y recibido 2
2023-04-24 09:54:22.550322 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.550370 - ('127.0.0.1', 34694) : Numero de secuencia esperado 3 y recibido 3
2023-04-24 09:54:22.550449 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.550490 - ('127.0.0.1', 34694) : Numero de secuencia esperado 4 y recibido 4
2023-04-24 09:54:22.550578 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.550590 - ('127.0.0.1', 34694) : Numero de secuencia esperado 5 y recibido 5
2023-04-24 09:54:22.550708 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.550731 - ('127.0.0.1', 34694) : Numero de secuencia esperado 6 y recibido 6
2023-04-24 09:54:22.550925 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.550939 - ('127.0.0.1', 34694) : Numero de secuencia esperado 7 y recibido 7
2023-04-24 09:54:22.551041 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.551149 - ('127.0.0.1', 34694) : Numero de secuencia esperado 8 y recibido 8
2023-04-24 09:54:22.551266 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.551307 - ('127.0.0.1', 34694) : Numero de secuencia esperado 9 y recibido 9
2023-04-24 09:54:22.551353 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.551365 - ('127.0.0.1', 34694) : Numero de secuencia esperado 10 y recibido 10
2023-04-24 09:54:22.551401 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.551411 - ('127.0.0.1', 34694) : Numero de secuencia esperado 11 y recibido 11
2023-04-24 09:54:22.551507 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.551527 - ('127.0.0.1', 34694) : Numero de secuencia esperado 12 y recibido 12
2023-04-24 09:54:22.551608 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.551630 - ('127.0.0.1', 34694) : Numero de secuencia esperado 13 y recibido 13
2023-04-24 09:54:22.551696 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.551719 - ('127.0.0.1', 34694) : Numero de secuencia esperado 14 y recibido 14
2023-04-24 09:54:22.551799 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.551821 - ('127.0.0.1', 34694) : Numero de secuencia esperado 15 y recibido 15
2023-04-24 09:54:22.551922 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.551997 - ('127.0.0.1', 34694) : Numero de secuencia esperado 16 y recibido 16
2023-04-24 09:54:22.552077 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.552099 - ('127.0.0.1', 34694) : Numero de secuencia esperado 17 y recibido 17
2023-04-24 09:54:22.552198 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.552395 - ('127.0.0.1', 34694) : Numero de secuencia esperado 18 y recibido 18
2023-04-24 09:54:22.552522 - ('127.0.0.1', 34694) : Recibimos paquete de 1472 bytes
2023-04-24 09:54:22.552600 - ('127.0.0.1', 34694) : Numero de secuencia esperado 19 y recibido 19
2023-04-24 09:54:22.552782 - ('127.0.0.1', 34694) : Recibimos paquete de 174 bytes
2023-04-24 09:54:22.552844 - ('127.0.0.1', 34694) : Numero de secuencia esperado 20 y recibido 20
2023-04-24 09:54:27.556286 - None : No obtuve paquete, finalizamos
```

- Comprobación de archivos de bajada exitosos:

  Para comprobar que los archivos que subimos se hayan subido bien utilizamos el comando md5sum para comprobar que el hash de cada archivo es igual al de los demás.

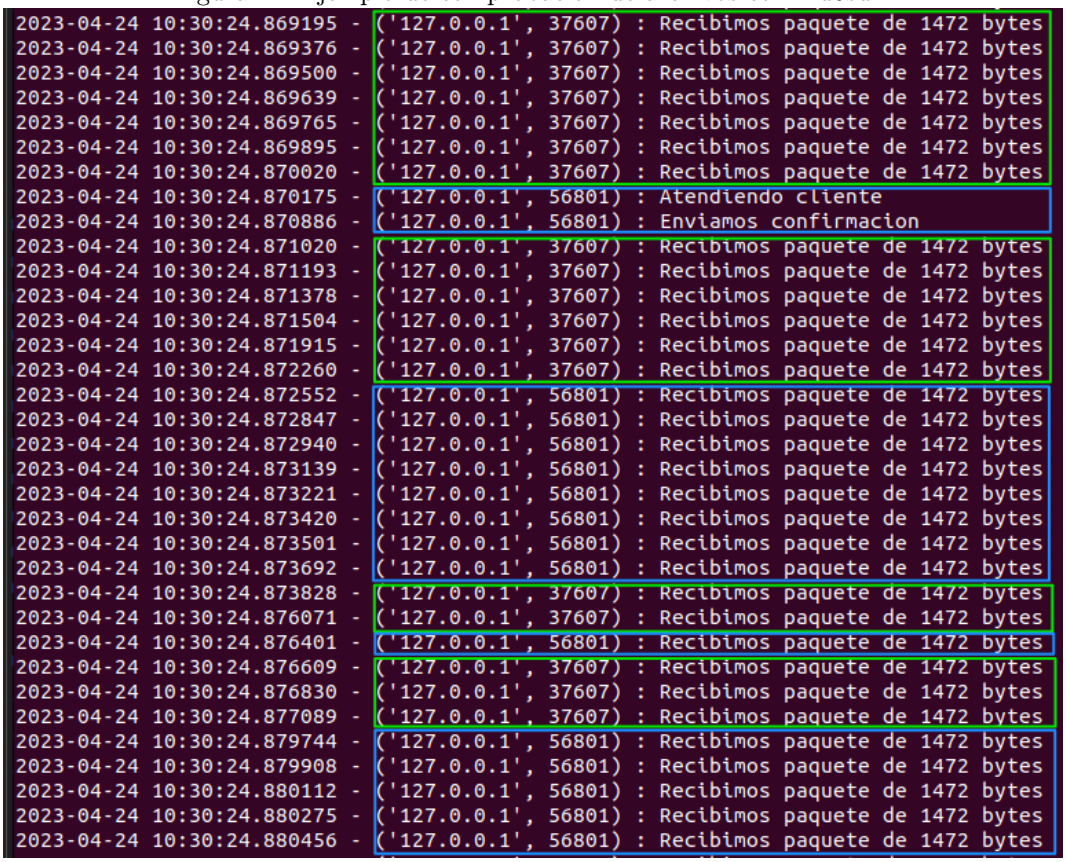
Figura 20: Ejemplo de subida de archivos concurrentemente

```
vicky@vicky-VirtualBox:~/Escritorio/distribuidostp$ md5sum foto30kbsaw.jpg foto30kbsr.jpg fotobajadasaw.jpg fotobajadasr.jpg
ae70a005ba82c8dce3d46ac5b444146d  foto30kbsaw.jpg
ae70a005ba82c8dce3d46ac5b444146d  foto30kbsr.jpg
ae70a005ba82c8dce3d46ac5b444146d  fotobajadasaw.jpg
ae70a005ba82c8dce3d46ac5b444146d  fotobajadasr.jpg
```

Prueba de subida con dos clientes de forma concurrente

Para realizar esta prueba se ejecuto al servidor con el modo Stop And Wait y dos clientes (uno atrás del otro) con el mismo modo. En la Figura 21 se puede observar como el servidor venia atendiendo a un cliente y pasa al siguiente, intercalando la recepción de paquetes de ambos clientes.

Figura 21: Ejemplo de comprobación de archivos con md5sum



Prueba de archivo inexistente en subida y en bajada

Al intentar subir o descargar un archivo que no existe se mostraran los siguientes errores:

Figura 22: Ejemplo de intentar subir archivo que el cliente no posee

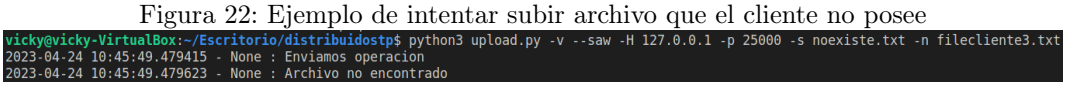


Figura 23: Ejemplo de intentar bajar archivo que el servidor no posee

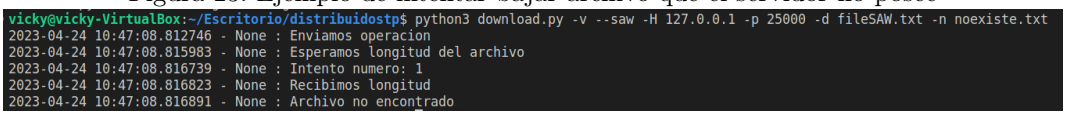
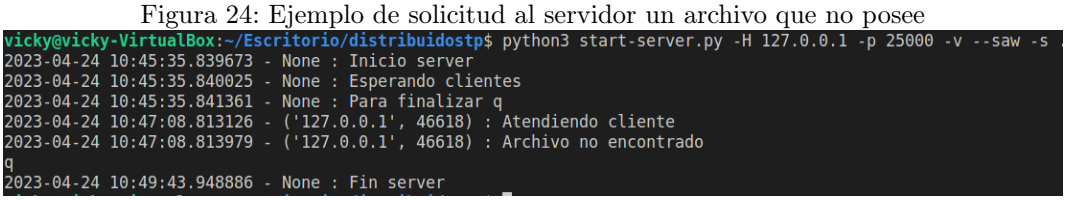


Figura 24: Ejemplo de solicitud al servidor un archivo que no posee



Tiempos con perdida

Subiendo un archivo de 30kB con Stop And Wait:

Perdida	Duracion
0 %	8.009 ms
20 %	2.989 seg
40 %	34.986 seg

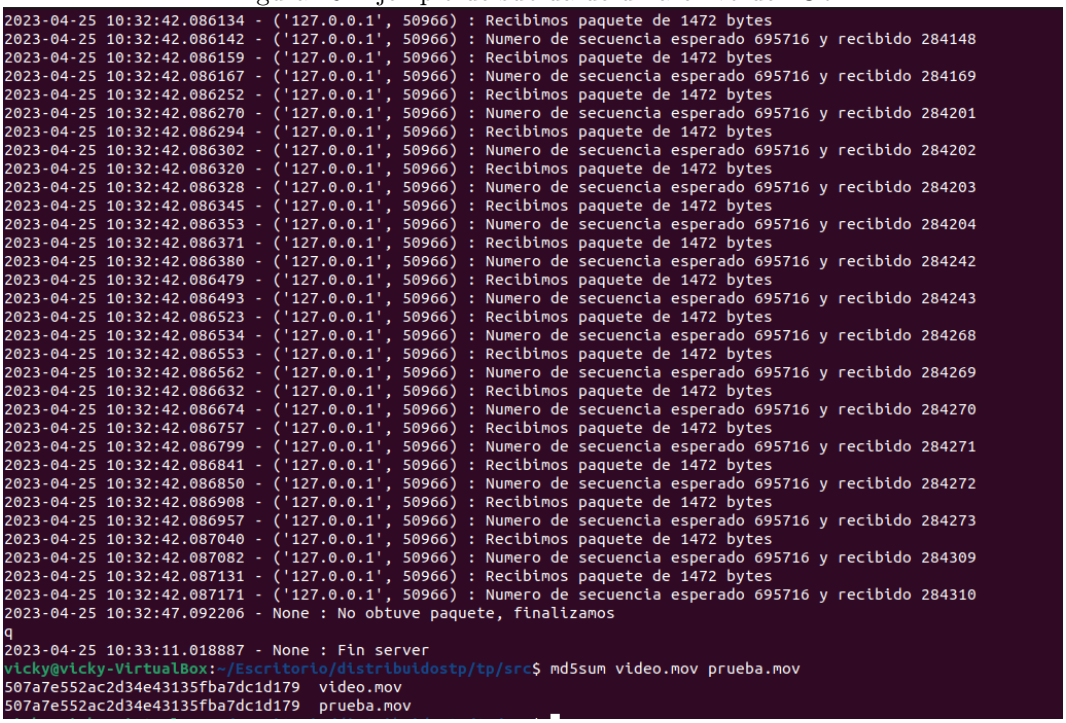
Subiendo un archivo de 30kB con Selective Repeat:

Perdida	Duración
0 %	5.676 ms
20 %	1.866 seg
40 %	3.654 seg

Subiendo archivos grandes

Se probo la subida de una grabación de pantalla de 1Gb con Selective Repeat sin perdida. El tiempo aproximado que llevo esta subida fue de 30m.

Figura 25: Ejemplo de subida de un archivo de 1Gb



Preguntas

1) La arquitectura Cliente-Servidor cuenta con dos tipos de componentes, el cliente y el servidor. El componente servidor se caracteriza por almacenar y manipular datos, escuchando peticiones de clientes. Por el otro lado, el componente cliente se encarga de realizar peticiones al servidor, esto es, enviar mensajes con el fin de recuperar o publicar algún dato hacia o desde el servidor. Dado que el servidor está constantemente activo, la comunicación es iniciada por parte del cliente, cuando este requiera algo del mismo. Esta arquitectura permite a un número de clientes conectarse con un mismo servidor.

2) El protocolo de la capa de aplicación tiene como función la de recuperar e interpretar el mensaje entregado por la capa de transporte, y define los campos y acciones para generar un mensaje válido.

3)El protocolo de aplicación desarrollado para este trabajo posee dos variantes, stop and wait y selective repeat. Ambas incluyen un establecimiento de conexión entre las dos partes previo al envío y recibo de mensajes. Esta conexión se realiza de la siguiente manera:

- El cliente intenta iniciar una conexión, mandando un mensaje al servidor. Este mensaje contiene el nombre y el tamaño del archivo que se quiere cargar o descargar. Si no recibe respuesta del servidor, vuelve a enviar el mensaje. El cliente repite este proceso hasta recibir una respuesta o mandar 5 veces el mensaje.

- Si se trata de una operación de carga (upload), el servidor manda un mensaje de confirmación al cliente que quiere iniciar la comunicación, en el que se muestra una "G".

- Si se trata de una operación de descarga (download), el servidor envía un mensaje con el tamaño del archivo a descargar, y el cliente, en respuesta, envía un mensaje de confirmación ("G") para iniciar la descarga.

Una vez establecida la conexión, dependiendo de la operación el cliente o el servidor inicia la transferencia del archivo. El archivo se envía en paquetes de 1469 bytes para selective repite, y de 1471 para stop and wait. Para ambas variantes (stop and wait y selective repeat) se utiliza una lógica similar a la que se usa en el protocolo TCP para los mecanismos con el mismo nombre. Esto incluye un timer y mensajes de confirmación de recepción (acks) para cada paquete (ver sección implementación).

4)El protocolo de transporte TCP asegura que los mensajes lleguen a destino, en el orden en el que fueron enviados y sin posibilidad de que estén corrompidos. Además, el mensaje de TCP posee un campo checksum que sirve como un sistema débil de manejo de errores. Todo esto requiere que ambas partes establezcan una conexión anterior al envío de mensajes, que se realiza a través del treeway-handshake. Este protocolo incluye también un sistema de control de flujo y de tráfico, con el que regula la velocidad de envío de mensajes si, por ejemplo, la red está congestionada. Por otro lado, el protocolo UDP no asegura la recepción correcta de mensajes, y confía en el best effort del protocolo de red IP. También posee el campo checksum, pero no requiere establecer una conexión previa, ni posee sistema de control alguno. La consecuencia de esto último es una comunicación posiblemente más rápida, por lo que este protocolo se utiliza en aplicaciones en lo que es relevante mantener una velocidad mínima, sin importar la pérdida de algunos paquetes.

Dificultades encontrades

Para la realización del trabajo practico encontramos las siguientes dificultades:

- El armado de la arquitectura inicial a partir de la cual pudiésemos escalar el resto de los desarrollos.
- El tiempo, 2 fines de semana considerando que todos los miembros del equipo trabajan y estudian deja lugar a poco tiempo de re-implementar cosas en el caso que fuese necesario.
- Cierre de conexiones, el proceso de diagramar salidas para que tanto servidor o cliente no queden en estados inválidos permanentes.

Conclusiones

Se llegaron a las siguientes conclusiones:

- Pudimos profundizar en el conocimiento de entrega de paquetes fiable.Se logro implementar un protocolo que incluye Stop and Wait y Selective Repeat, en forma encapsulada haciéndolo transparente para el resto del protocolo.
- Pudimos observar las diferencias de performance entre Stop and Wait y Selective Repeat, donde por ejemplo, para el caso de una imagen de 500kb con enlaces con un 20 por ciento de perdida de paquetes en promedio Stop and Wait estaba tardando 14 segundos mientras que Selective Repeat 3.89 segundos.
- Se podría encontrar un mejor valor para el timeout que nos permita mejorar los tiempos.