Trabajo Práctico Final 2024

Diseño e Implementación de cliente/servidor dedicado al compartimiento de archivos

Nombre	Apellido	Legajo	E-mail
Nicolas	Casella	62.311	**ncasella@itba.edu.ar**
Timoteo	Smart	62.844	**tsmart@itba.edu.ar**
Catalina	Müller	63.199	**camuller@itba.edu.ar**
Manuel	Quesada	63.580	**mquesada@itba.edu.ar**

Aclaración Importante:

El *tracker* tiene la posibilidad de traducir *localhost* en IPs públicas, con el fín de poder conectar clientes en distintas redes. Sin embargo, al testearlo falla en establecer conexiones. No se sabe si esto se debe a configuraciones de firewall/puertos o es una falla propia del código.

Si se inicia el servidor *tracker* con el flag *-l*, es posible desactivar esta funcionalidad, y trabajar completamente desde *localhost* con múltiples clientes.

1. Protocolos desarrollados

Para este trabajo práctico se desarrollaron dos protocolos de aplicación para compartir archivos binarios a través de internet: un *tracker* que almacena la información sobre los usuarios y los archivos que tienen en su repositorio, y una aplicación cliente que permite a un usuario obtener la copia a su propio repositorio de un archivo de otros usuarios que ya lo tengan descargado.

Aplicación Tracker

Mantiene una lista actualizada de los archivos compartidos por los usuarios, con información propia del archivo (nombre, MD5, tamaño), información de los usuarios que lo comparten (lo tienen en su repositorio), denominados *seeders*, e información de los usuarios que lo están descargando, denominados *leechers*. Si el *host* del *tracker* desea ser *seeder* o *leecher*, se utilizó una librería externa para obtener la IP pública mediante un servidor STUN[1].

Para poder descargar archivos, un usuario debe estar registrado y conectado. Puede realizar un *log-in*, indicando nombre de usuario y contraseña de la siguiente manera:

PLAIN *nombreDeUsuario*:*contraseña*

De no tener cuenta el usuario puede registrarse de la misma manera, indicando un nombre de usuario que no esté registrado. El servidor indicará si el nombre de usuario es válido, y de serlo pedirá confirmación al usuario (*y/n*), y si responde con '*y*' registrará al usuario. El usuario debe estar conectado para poder descargar archivos, y debe estar registrado para poder conectarse.

Si un usuario inicia más de una aplicación cliente con la misma IP y puertos distintos, podrá escuchar con esa IP en los distintos puertos, y el *tracker* agregará a la lista de usuarios múltiples entradas para esa IP, una por cada puerto.

[1] https://github.com/0xFireWolf/STUNExternalIP/

Un usuario registrado comparte los archivos que tenga almacenados en un directorio *repository*. El servidor registra los archivos de este directorio, agregándolos a la lista de archivos disponibles junto al *seeder* que los está compartiendo. Al ser UDP el protocolo de transporte entre el *tracker* y los usuarios, el servidor no puede saber cuando se desconecta un *seeder*, por lo chequea periódicamente el repositorio, renovando el archivo en la lista si lo vuelve a encontrar en el repositorio (lo vuelve a registrar), o eliminando el par *seeder-archivo* de la lista si no (los archivos son eliminados de la lista después de un tiempo fijo si no son registrados nuevamente).

Para poder ejecutar los comandos del protocolo debe ejecutarse tracker, y luego en otra terminal ejecutar el comando nc -u *ip* 15555, y en esa terminal ejecutar los comandos.

El protocolo de comunicación para el *tracker* tiene la siguiente forma:

- **PLAIN *nombreDeUsuario:contraseña***
  Inicia sesión del usuario, o registra al nuevo usuario si no existe.
  Responde un mensaje iniciado por OK en caso de éxito y ERR en caso de error.
- **LIST <files|peers *md5*>**
  Si recibe el parámetro files, lista los archivos disponibles para descargar, junto con su tamaño en bytes y el hash md5 del archivo.
  Si recibe el parámetro peers, se le debe pasar también el hash md5 de un archivo y lista los usuarios que tienen completo ese archivo para descargar.
  En ambos casos primero se enviará un mensaje indicando la cantidad de peers o archivos a ser recibidos.
  En el caso de files se enviará un último mensaje diciendo “No more files”.
- **REGISTER *nombreDeArchivo bytes md5***
  Registra un archivo, agregándole a la lista de archivos junto con la IP del usuario que lo registró como *seeder*.
- **CHECK *usuario seederIP md5***
  Indica si el usuario de ip *seederIP* en el puerto *puerto* tiene disponible para descargar el archivo de hash *md5*.
  Responde un mensaje iniciado por OK en caso de éxito y ERR en caso de error.
- **CHANGEPASSWORD *contraseñaVieja:contraseñaNueva***
  Cambia *contraseñaVieja* por *contraseñaNueva*, si el usuario está logueado y la contraseña actual coincide con *contraseñaVieja*.
- **SIZE *md5***
  Indica el tamaño en bytes del archivo de hash *md5*
- **SETPORT *puerto***
  modifica el puerto donde escucha el seeder de la IP actual por *puerto*
- **NAME *md5***
  Indica el nombre del archivo con hash *md5*
- **QUIT**
  Cierra la sesión del usuario

Argumentos posibles (flags)

-h Imprime los argumentos posibles y termina

-P <port> Puerto entrante para conexiones

Aplicación Cliente

Transfiere archivos de forma confiable entre usuarios utilizando TCP como protocolo de transporte. Los archivos se transfieren sin encriptar, solicitándolo de a partes.

Para la descarga de un archivo, se utilizan distintos *threads* para solicitar a como mucho 5 clientes que tengan el archivo, distintas partes del mismo, para optimizar la descarga. Un *file manager* indica a cada thread qué *chunk* del archivo es necesario y debe conseguir siguiente. De esta forma se asegura que dos *threads* no busquen el mismo pedazo del archivo, y si un thread, después de consultar a los *seeders* disponibles, no consigue el chunk indicado, se lo indica al file manager y este vuelve a intentar pedir ese pedazo, como mucho un total de 4 veces. Si 4 veces no se pudo obtener esa parte del archivo, se retorna un error en la descarga del archivo.

Es posible iniciar dos aplicaciones cliente con la misma IP que escuchen en distintos puertos.

Cuando un seeder recibe el intento de conexión de un leecher, antes de aceptarlo consulta con el tracker si ese leecher está registrado, y de no estarlo rechaza la conexión.

Los comandos para la aplicación son los siguientes:

- **login *nombreDeUsuario contraseña***
  Llama a *PLAIN* de la aplicación del *tracker* para iniciar sesión o registrar a un usuario.
- **files**
  Llama a *LIST files* de la aplicación del *tracker* para mostrar los archivos disponibles para descargar.
- **download *md5***
  Si hay *seeders* que tienen registrados el archivo de hash *md5*, lo descarga a la carpeta *repository* del *leecher*.
- **pause**
  Pausa la descarga si hay algún archivo descargandose al momento de llamar al comando.
- **resume**
  Reanuda la descarga pausada.
- **cancel**

Cancela la descarga del archivo descargandose al momento de llamar al comando.

- **dstatus**
  Imprime el estado de la descarga actual: qué porcentaje ya se descargó de la misma.
- **seeders**
  Imprime la lista de IPs y puertos de los seeders de los cuales se está descargando un archivo al momento de llamar al comando.
- **leechers**
  Imprime la lista de leechers que están descargando algún archivo del usuario, y el hash md5 del archivo.

El protocolo de comunicación entre dos clientes es el siguiente:

- **<hash>:<byteFrom>:<byteTo>:<user>:** Debe devolver el desde el *byteFrom* hasta *byteTo* del archivo que corresponda a *hash*. Únicamente en caso de que *user* sea un usuario válido y registrado.
- Los clientes no necesitan informarse entre ellos cuando una conexión se rechaza, corta o se acepta. En caso de recibir un pedido no válido, se debe cortar la conexión con el fin de evitar enviar bytes inválidos.

Argumentos posibles (flags)

-h	Imprime los argumentos posibles y termina
-L <port>	Dirección del puerto pasivo para atender conexiones
-t <addr>	Dirección IP del tracker a conectarse
-T <port>	Dirección del puerto del tracker a conectarse
-N <limit>	Límite de conexiones entrantes (leechers) a aceptar

2. Problemas encontrados durante el diseño y la implementación

El primer problema encontrado fue el diseño de la arquitectura cliente/tracker y como estos dos deben comunicar los datos de los usuarios entre sí. Inicialmente se consideró la aplicación A y el tracker como entidades diferentes que, a su vez, cumplían propósitos similares entre la aplicación A y la aplicación B.

En la aplicación cliente hubieron problemas relacionados a *race conditions* durante el desarrollo, aunque estos pudieron ser solucionados eventualmente aún así consumieron una gran cantidad de tiempo. Por otro lado, surgieron problemas en cuanto a la comunicación con el *tracker* desde el cliente, mayormente debido al uso de UDP como protocolo de transporte y su falta de uso de sesiones.

Otro problema que surgió fue al momento de intentar descargar archivos de mayor tamaño. Para poder hacerlo se modificó el File Manager para que el buffer que reservaba en memoria tuviera un tamaño máximo, y si el archivo era más grande que eso se descargara de a secciones. Con esta modificación empezó a ocurrir que se copiaban bytes de más o se pisaban bytes al escribir el archivo, y llevó bastante tiempo poder solucionarlo. A la vez, esta modificación nos permitió aumentar el tamaño de los *chunks* enviados, lo que reveló que no estaba propiamente administrado el caso donde los mensajes se fraccionaran en múltiples paquetes distintos.

Durante el último día de desarrollo, se decidió implementar una funcionalidad que permitía traducir *localhost* en IPs públicas, sin embargo ocurrieron problemas a la hora de crear conexiones desde distintas redes. Esto ya había sido testeado en los días previos, pero alguna modificación lo cambió.

Por último, al priorizar la velocidad de implementación de *features* y funcionalidad del código (con el fin de cumplir con la fecha de entrega); la calidad y claridad del código se vió afectada negativamente. Dejando una pequeña deuda técnica.

3. Limitaciones de la aplicación

- La aplicación no permite que un usuario descargue más de un archivo simultáneamente.
- Un archivo solo puede ser descargado de no más de 5 seeders distintos a la vez.
- A pesar de que una aplicación cliente no pueda asignar múltiples puertos de escucha, se pueden utilizar múltiples aplicaciones cliente desde un mismo IP + usuario.
- Los archivos solo pueden ser procesados por el cliente si el nombre no incluye espacios.

4. Posibles extensiones

- Actualmente, si se desea descargar un archivo y ya hay uno en la carpeta de descargas con el mismo nombre, este último se reemplaza por el nuevo. Se podría implementar que el nuevo archivo se genere agregando (i) al nombre, i siendo un índice secuencial empezando en uno, aumentando por cada archivo con el mismo nombre que se descarga(*filename*, *filename*(1), *filename*(2)...).
- Permitir a un mismo usuario descargar más de un archivo simultáneamente.
- Una implementación futura podría refactorizar parte del código, mejorando el estilo y adaptabilidad del mismo.

5. Conclusión

Se consiguieron los objetivos de desarrollar tanto el *tracker* como la aplicación cliente en el tiempo pedido. El trabajo permitió al grupo de aprender y entender más a fondo tanto los protocolos de comunicación/transmisión (como pueden ser TCP, UDP, etc.) como el uso, administración y desarrollo de servidores y conexiones p2p. Las extensiones de tiempo dadas por la cátedra fueron de gran ayuda a la hora de mejorar el funcionamiento y claridad del código.

6. Referencias

Se utilizaron secciones de código dadas por la cátedra de Protocolos de Comunicación 1C-2024. Principalmente:

selector.c
selector.h
args.c (modificado)
args.h (modificado)

Además, se modificó el código relacionado a la apertura de puertos UDP y TCP.

A pesar de que el resto del código fue escrito y desarrollado por los integrantes del grupo, se le hicieron consultas a ChatGPT para algunas secciones de código.

7. Pruebas de testeo

En el archivo README.md se pueden encontrar ejemplos de testeo para probar las distintas funcionalidades del proyecto:

REGISTRAR UN USUARIO
1. iniciar tracker con ./tracker
2. iniciar client con ./client
3. dentro de la consola de comandos de client, correr:
   - register testuser testpassword
   - files
   - logout

INGRESAR SESIÓN
1. iniciar tracker con ./tracker
2. iniciar client con ./client

3. dentro de la consola de comandos de client, correr:
   - login testuser testpassword
   - files
   - logout

DESCARGAR UN ARCHIVO LOCAL
1. iniciar tracker con ./tracker
2. iniciar client con ./client
3. dentro de la consola de comandos de client, correr:
   - login testuser testpassword
   - files
   - download <hash>
   - logout

DESCARGAR UN ARCHIVO DESDE OTRO CLIENTE
1. iniciar tracker con ./tracker
2. iniciar client con ./client
3. iniciar una segunda sesión con ./client -L 2526, puede ser cualquier otro puerto, mientras que no sea el default
4. dentro de la consola de comandos de client, correr:
   - login testuser testpassword
   - files
   - download <hash>
   - logout

DESCARGAR UN ARCHIVO DESDE OTRO CLIENTE
1. iniciar tracker con ./tracker
2. iniciar client con ./client
3. iniciar una segunda sesión con ./client -L 2526, puede ser cualquier otro puerto, mientras que no sea el default
4. dentro de la consola de comandos de client, correr:
   - login testuser testpassword
5. dentro de la segunda consola, correr:
   - register testuser2 testpassword2
   - files
   - download <hash>
   -logout

VER LOS SEEDERS CONECTADOS
1. iniciar tracker con ./tracker
2. iniciar client con ./client
3. iniciar una segunda sesión con ./client -L 2526, puede ser cualquier otro puerto, mientras que no sea el default

4. dentro de la consola de comandos de client, correr:
   - login testuser testpassword
5. dentro de la segunda consola, correr:
   - register testuser2 testpassword2
   - files
   - download <hash>
   - seeders

VER LOS LEECHERS CONECTADOS
1. iniciar tracker con ./tracker
2. iniciar client con ./client
3. iniciar una segunda sesión con ./client -L 2526, puede ser cualquier otro puerto, mientras que no sea el default
4. dentro de la consola de comandos de client, correr:
   - login testuser testpassword
5. dentro de la segunda consola, correr:
   - register testuser2 testpassword2
   - files
   - download <hash>
6. volver a la primer consola, correr:
   - leechers

VER EL ESTADO DE LA DESCARGA ACTUAL
1. iniciar tracker con ./tracker
2. iniciar client con ./client
3. iniciar una segunda sesión con ./client -L 2526, puede ser cualquier otro puerto, mientras que no sea el default
4. dentro de la consola de comandos de client, correr:
   - login testuser testpassword
5. dentro de la segunda consola, correr:
   - register testuser2 testpassword2
   - files
   - download <hash>
   - dstatus

PAUSAR, REANUDAR Y CANCELAR DESCARGA
1. iniciar tracker con ./tracker
2. iniciar client con ./client
3. iniciar una segunda sesión con ./client -L 2526, puede ser cualquier otro puerto, mientras que no sea el default
4. dentro de la consola de comandos de client, correr:
   - login testuser testpassword
5. dentro de la segunda consola, correr:

- register testuser2 testpassword2
- files
- download <hash>
- pause
- dstatus
- resume
- cancel
- logout