

Redes de Computadores – Laboratorio 2: Compartición de archivos con UDP

2 de junio de 2022

Continuando con lo que hemos desarrollado en el primer laboratorio, trabajaremos en esta ocasión implementando un programa que permita compartir archivos de una máquina a varias usando sockets UDP como base; algo parecido a lo que hacen las tradicionales páginas de descargas o los servidores FTP. Considerando que UDP no tiene las mismas garantías de su compañero TCP con respecto a fiabilidad y ordenamiento, además de no trabajar con conexiones (no existen handshakes de inicio ni de cierre en UDP), los principios con los que tendrán que trabajar serán ligeramente distintos.

1. Requerimientos

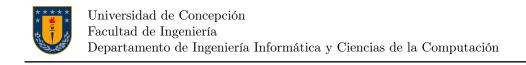
Los requerimientos a cumplir son los siguientes:

- El programa debe permitir compartir archivos de una máquina a otras.
- La máquina que hará de servidor permitirá el acceso a los archivos de una carpeta determinada por el usuario, cuyos archivos podrán ser descargados por cualquier otra máquina con un programa cliente.
- Al igual que para la primera parte, estos programas (servidor y cliente) pueden ser un mismo ejecutable o estar separados.
- El servidor deberá poder aceptar peticiones de varios clientes a la vez, no siendo necesario implementar concurrencia
- El programa debe utilizar sockets UDP manualmente, no pudiendo usar librerías que ya implementen transferencia de archivos.
- El programa cliente (receptor) deberá indicar el progreso de la transferencia del archivo, indicando también si no fue posible conectarse o si dejó de recibir respuestas del servidor.
- El protocolo debe permitir la transferencia de archivos binarios o de texto plano de tamaño arbitrario, así como también indicar el tamaño del archivo a descargar.

2. Evaluación

El laboratorio se evaluará (idealmente) de a pares bajo los siguientes requerimientos, mediante una escala de 5 puntos:

- Funcionalidad (2 puntos): El programa permitirá la transferencia de archivos (binarios y de texto plano) de servidor a cliente bajo los requerimientos dados, indicando cualquier error que pudiera ocurrir en el proceso y abortando si es necesario.
- Múltiples clientes (1 punto): El servidor puede aceptar peticiones de múltiples clientes, sin entremezclar las conexiones.





- Transferencias de tamaño arbitrario (0.5 puntos): El protocolo del programa permite la transferencia de archivos de cualquier tamaño.
- Información sobre la transferencia (0.5 puntos): El cliente mostrará por pantalla el progreso de la transferencia del archivo.
- **Demostración** (1 punto): Se demuestra el funcionamiento del programa, mostrando que los puntos anteriores se cumplen.

De forma adicional, cada uno de estos requisitos extras otorgarán puntos adicionales sobre la calificación anterior, hasta completar 5 puntos:

- Limpieza de código (0.5 puntos): El código del programa tiene una sintaxis clara, consistente y limpia.
 - Pista: Pueden utilizar programas de formato automático como clang-format (C/C++), autopep8 (Python) o rustfmt (Rust), dependiendo del lenguaje que quieran utilizar. Este clase de utilidades pueden ser integradas fácilmente en editores de código como Visual Studio Code o (Neo)vim.
- Compilación o testing automatizado (0.5 puntos): El programa utiliza un toolchain de compilación, que se encarga de ejecutar los pasos de compilación de forma automática o, bien, el proyecto cuenta con un script que permite probar el funcionamiento del programa de forma (semi)automática, en caso de que el lenguaje del proyecto sea interpretado (como Python).
 - Pista: Dependiendo del lenguaje, pueden utilizar herramientas como CMake, autotools o Cargo.
- Acceso seguro al sistema de archivos del servidor (0.5 puntos): El cliente no puede acceder a carpetas fuera de la usada para el servidor para compartir archivos.
 - **Ejemplo**: Si la carpeta que comparte el servidor es /home/jorge/Compartido, el cliente no debería poder descargar el archivo ../Documentos/claves.txt (/home/jorge/Documentos/claves.txt).
 - Pista: A este concepto se le conoce como chroot.
- Criptografía (1 punto): El programa permite utilizar algún algoritmo de cifrado (simétrico o asimétrico) para la transferencia de archivos.

Cada día de atraso incurrirá en una penalización de 0,1 puntos (una décima).

El código debe ser entregado como un archivo comprimido (de cualquier tipo) o un *link* a un repositorio de código, junto con un README con instrucciones de cómo compilarlo y ejecutarlo. También se requerirá un video corto de demostración (no mayor a 5 minutos) mostrando el funcionamiento del programa, para el cual deben entregar un *link* (consejo: para hacer más fácil la grabación, pueden ejecutar cliente y servidor en localhost; es igual de válido).