Enunciado del proyecto de prácticas: *nanoFiles*

Redes de Comunicaciones - Curso 2024/25

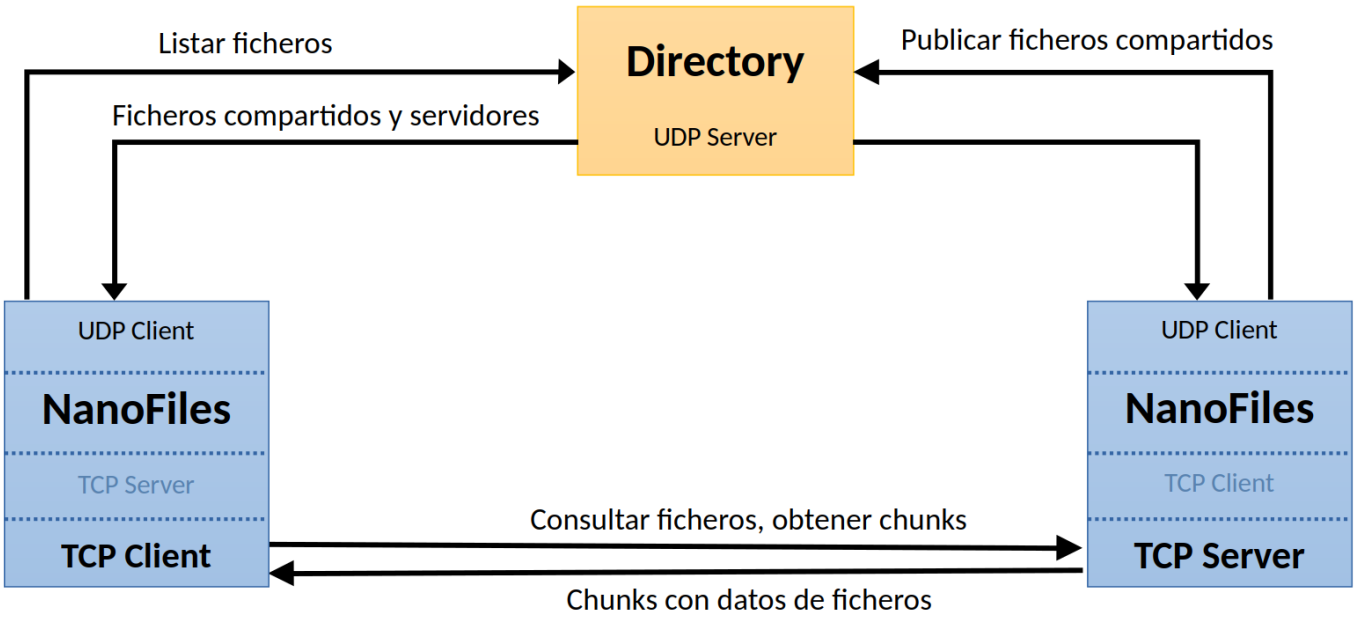
Visión global y objetivos

En el desarrollo de esta práctica, el alumnado aprenderá a diseñar y programar protocolos de comunicación en Java. La práctica consiste en realizar un sistema de compartición y transferencia de ficheros al que denominaremos *nanoFiles*, y que se describe a continuación.

El sistema *nanoFiles* está formado por un servidor de directorio (programa *Directory*) y un conjunto de *peers* o pares (programa *NanoFiles*), que se comunican entre sí de la siguiente forma:

- Por un lado, la comunicación entre cada *peer* de *NanoFiles* y el servidor de directorio se rige por el modelo *cliente-servidor*. Un *peer* actúa como cliente del directorio para consultar los ficheros que pueden ser descargados de otros *peers*, publicar los ficheros que quiere compartir con el resto de pares y obtener los servidores que comparten un determinado fichero.
- Por otro lado, el modelo de comunicación entre pares de *NanoFiles* es *peer-to-peer* (P2P).
    - Cuando un *peer* actúa como cliente de otro *peer* servidor, el cliente puede consultar los ficheros disponibles en el servidor y descargar aquellos fragmentos (*chunks*) de un fichero determinado que solicite.
    - De manera complementaria, un *peer* puede convertirse a petición del usuario en servidor de ficheros, de forma que escuche en un puerto determinado en espera de que otros *peers* se conecten para solicitarle fragmentos de los ficheros que está compartiendo.

La siguiente figura muestra un esquema general orientativo del conjunto de componentes de *nanoFiles*:



El objetivo global de esta práctica es que el alumnado sea capaz de diseñar los protocolos necesarios para intercambiar los mensajes involucrados así como de implementar el software de todos los elementos que constituyen el sistema *nanoFiles*.

Especificación de la práctica

Una vez que el alumnado haya implementado la funcionalidad exigida, un *peer* podrá actuar simultáneamente como cliente y como servidor de ficheros (modelo de comunicación P2P). En el sistema *nanoFiles*, todos los pares deben ser capaces de comprobar si existe un servidor de directorio que utilice un protocolo de comunicación *compatible* con el del cliente. De esta forma, deberá ser posible distinguir al servidor de directorio del propio grupo, de otros servidores de directorio implementados por otros grupos de prácticas que se estén ejecutando en ese instante en el laboratorio.

La función principal del servidor de directorio es mantener el listado de los ficheros que están siendo compartidos por los *peers* del sistema *nanoFiles*, así como la identidad de los servidores que comparten cada uno de dichos ficheros. De esta forma, los *peers* podrán consultar a través del directorio el listado de ficheros disponible, y obtener a través del directorio las direcciones de los servidores que comparten un determinado fichero que se desea descargar. Una vez que un *peer* haya obtenido del directorio las direcciones de los servidores del fichero que desea descargar, se conectará directamente a dichos servidores para realizar la descarga del fichero, de forma que sea posible obtener fragmentos de los datos del fichero de diferentes servidores.

Para hacer posible que otros *peers* puedan consultar y descargar ficheros en *nanoFiles*, un par debe poder lanzar un servidor de ficheros. Una vez hecho esto, el *peer* aceptará conexiones de otros peers para solicitarle descargar alguno de los ficheros que comparte. Para que el resto de *peers* puedan conocer qué ficheros hay disponibles para descargar en un instante dado, cada nuevo par que se convierta en servidor de ficheros deberá comunicarlo al directorio, enviando el listado de ficheros que comparte, con el nombre, tamaño y hash de cada uno. De esta forma, cualquier otro par podrá consultar al directorio qué ficheros están compartiendo el resto de *peers* servidores, realizar búsquedas y descargas.

En *NanoFiles*, los pares cliente que desean descargar ficheros de otros pares obtienen la información necesaria para conectarse a los pares servidor a través del directorio. Para llevar a cabo una descarga, se obtendrán trozos del fichero de **todos los servidores que compartan el fichero** que el usuario desea descargar.

Diseño de los protocolos necesarios

Específicamente, el alumnado deberá llevar a cabo el diseño de dos protocolos de comunicación de nivel de aplicación del sistema *NanoFiles*:

- Para la comunicación entre cada *peer* y el directorio, el alumnado diseñará un protocolo confiable semi-dúplex basado en parada y espera, que operará sobre un protocolo de nivel de transporte no confiable como UDP.
- Para la comunicación entre dos *peers*, el alumnado diseñará un protocolo asumiendo que el protocolo de nivel de transporte es confiable (como TCP).

En particular, los estudiantes deberán realizar:

- Respecto a los autómatas, se pide especificar, por separado, los siguientes autómatas:
  - Autómata para el servidor de directorio.
  - Autómata para un *peer* cuando actúa como cliente del directorio.
  - Autómata para un *peer* cuando actúa como cliente de otro *peer* servidor de ficheros.
  - Autómata para un *peer* cuando actúa como servidor de ficheros otro *peer* cliente.
- Respecto a las especificaciones de los mensajes a intercambiar, se deben utilizar los siguientes lenguajes:
  - **Mensajes textuales** (*Field:Value*) para el protocolo de comunicación con el directorio.
  - **Mensajes binarios** para la comunicación entre pares.

Se deja a elección del alumnado las decisiones relativas a los diferentes tipos de mensajes necesarios así como a la codificación de los distintos campos.

Implementación del software

En paralelo a la fase de diseño, los estudiantes tendrán que implementar los elementos software necesarios para el proyecto de prácticas. Para facilitar esta tarea, se proporciona al alumnado un material de partida que consta de los siguientes elementos:

- Esqueleto de código del servidor de directorio
- Implementación sin terminar de la parte cliente del programa *NanoFiles*, incluyendo:
  - Interfaz de usuario, basada en un sencillo intérprete de comandos (ya implementado).
  - Lógica del programa cliente (parcialmente implementado).
- Implementación esbozada de la parte servidor del programa *NanoFiles*, incluyendo:
  - Servidor capaz de ejecutarse tanto en primer como en segundo plano.
  - Clase que modela los hilos que pueden ser creados por el servidor para atender a los clientes conectados.
- Clases auxiliares:
  - Para la representación de mensajes (sólo esbozadas).
  - Para la obtener la base de datos de ficheros compartidos (ya implementada)

Así pues, el trabajo del alumnado consistirá en completar tanto el programa NanoFiles (código que ejecuta cada uno de los *peers*) como el programa Directory (servidor de directorio) para conseguir que la práctica sea

funcional. A continuación, se describe la lógica de cada uno de las dos aplicaciones Java cuya programación debe realizarse como parte de las prácticas de esta asignatura.

Lógica del programa *Directory*

El programa *Directory* quedará a la espera, una vez lanzado, de los mensajes que vaya recibiendo de los *peers*. El servidor de directorio utilizará el **puerto 6868/udp** para recibir mensajes entrantes. El directorio irá actualizando sus estructuras de datos a partir de las peticiones recibidas de los *peers* y, en su caso, irá respondiendo con la información solicitada en la consulta.

Con la finalidad de que el alumnado pueda depurar el correcto funcionamiento de los clientes que le solicitan servicio, **el directorio debería imprimir por consola un breve resumen de los mensajes recibidos y enviados**, así como de aquellos mensajes que se han perdido.

El programa *Directory* ya tiene implementado el mecanismo por el cual se le puede indicar cuál es la probabilidad de que se pierda un paquete en la comunicación con el directorio. Esto se consigue mediante la opción `-loss <probability>` y resultará útil para verificar el correcto funcionamiento de la implementación del protocolo entre el directorio y los pares, ya que dicho protocolo de nivel de aplicación debe garantizar la entrega confiable a pesar de que se produzcan pérdidas en el nivel de transporte (UDP). El valor de probabilidad oscila entre 0 (no se pierde ningún datagrama) y 1 (se pierden todos). Por defecto, la probabilidad de pérdida es 0.

Lógica del programa *NanoFiles*

El programa *NanoFiles* interactuará con el usuario mediante una línea de comandos. Al ejecutarlo, se le puede proporcionar opcionalmente un argumento, que es la ruta al directorio donde se ubican los ficheros que el *peer* comparte en el sistema *nanoFiles*. Por defecto, el programa *NanoFiles* buscará en un directorio llamado `nf-shared` a partir del directorio actual, el cual se creará al ejecutar el programa en caso de que no exista. En el escenario habitual en el que el programa se ejecuta desde el IDE Eclipse, el directorio de ficheros compartidos `nf-shared` se creará en el directorio raíz del proyecto Java (p.ej., `nanoFilesP2Palumnos`), al mismo nivel que los directorios `bin` y `src`.

Una vez en ejecución, el programa cliente acepta un conjunto de órdenes, si bien la funcionalidad asociada a la mayor parte de dichas órdenes no está programada y por tanto no hay ningún efecto al teclearlas. Dicha funcionalidad debe ser añadida por el alumnado, aunque la implementación de algunas de las órdenes que ya acepta el *shell* del programa *NanoFiles* no forma parte de funcionalidad mínima exigida sino que constituyen posibles mejoras. Tanto la funcionalidad mínima exigida como las posibles mejoras serán descritas más adelante en este documento.

Las órdenes cuya funcionalidad ya está implementada, y que **no suponen ninguna comunicación entre procesos**, son:

- `help`: Muestra ayuda sobre las órdenes soportadas.
- `myfiles`: Muestra los ficheros de la carpeta compartida por este *peer*, indicando su nombre, tamaño y *hash*. El directorio por defecto es `nf-shared` y por lo general se ubicará dentro del directorio del proyecto Eclipse.

Por otro lado, las órdenes que reconoce el *shell* de *NanoFiles* **cuya funcionalidad está pendiente de implementar** son:

- `ping`: Comprueba que el servidor de directorio está activo y usa un protocolo compatible con el del *peer*.
- `filelist`: Muestra los ficheros que están siendo compartidos por otros *peers*, que han sido publicados en el directorio.
- `serve`: Lanza un servidor de ficheros que escucha conexiones en el **puerto 10000/tcp**. Al lanzar un servidor de ficheros, se publica automáticamente al directorio los metadatos de los ficheros que este *peer* tiene en su carpeta compartida.
- `download <filename_substring> <local_filename>`: Descarga el fichero identificado por el primer parámetro (`<filename_substring>`) de todos los servidores de ficheros que lo tengan disponible, y lo guarda con el nombre indicado como segundo parámetro (`<local_filename>`).
- `upload <filename_substring> <remote_server>`: Sube (copia) un fichero local a otro *peer* remoto que esté sirviendo ficheros. El primer parámetro (`<filename_substring>`) indica el fichero local a enviar, mediante una subcadena del nombre del fichero que lo identifique de manera no ambigua. El segundo parámetro (`<remote_server>`) es la dirección del servidor destino de la subida.
- `quit`: Sale del programa.

La secuencia correcta en la cual se pueden teclear dichas órdenes, o las situaciones en las que algunas de ellas no están permitidas, dependerá del autómata que el alumnado haya diseñado. Por tanto, será responsabilidad de los mismos controlar dicho orden.

Con la finalidad de que el alumnado pueda depurar el correcto funcionamiento de sus servidores de ficheros, se sugiere que se imprima un resumen de los mensajes recibidos y enviados.

Funcionalidad obligatoria a implementar

Ninguno de los programas que se entregan al alumnado está concluido: la fase de implementación de las prácticas de la asignatura consiste en terminar su funcionalidad. Para ello el alumnado tendrá que llevar a cabo, entre otras, las siguientes tareas:

- Implementar el formato de mensajes diseñado para que los segmentos UDP intercambiados con el servidor de directorio contengan mensajes adecuadamente formateados.
- Implementar el formato de mensajes diseñado para que los segmentos TCP intercambiados entre un *peer* cliente y un *peer* servidor contengan mensajes adecuadamente formateados.

Con respecto a los requisitos mínimos para que un proyecto de prácticas se considere satisfactorio, será necesario implementar al menos la siguiente funcionalidad:

1. Comando `ping`: Contactar con el servidor de directorio para comprobar que está a la escucha y que utiliza un protocolo compatible con el del *peer*.

2. Comando `filelist`: Mostrar la lista de ficheros publicados al directorio, los cuales están disponibles para su descarga desde otros *peers* servidores de ficheros. El listado debe indicar para cada fichero, su nombre, tamaño y *hash*.

3. Comando `serve`: Lanzar un servidor de ficheros que escuche conexiones en el **puerto 10000/tcp**. Este servidor se deberá ejecutar en un hilo del programa distinto al principal, para que sea posible continuar utilizando el programa a través del *shell*, e interaccionando como cliente tanto con el directorio como con otros *peers*. Además, el servidor de ficheros lanzado en segundo plano debe poder atender a múltiples clientes conectados simultáneamente.

4. Comando `download`: Descargar un fichero identificado por una subcadena de su nombre, de entre los ficheros disponibles en un *peer* servidor de ficheros. El servidor deberá informar en caso de que la subcadena proporcionada no concuerde con ningún fichero compartido, o bien sea ambigua. La descarga debe obtener fragmentos del fichero (*chunks*) de **todos los servidores** que lo tengan disponible, si bien será suficiente con que dicha descarga de *chunks* sea secuencial: en cada instante sólo se descargan *chunks* desde uno de los servidores que lo comparte.

La implementación de esta **funcionalidad básica** permite alcanzar una **calificación máxima de 7 puntos**.

Funcionalidad opcional a implementar (mejoras)

Para obtener una mayor calificación en la práctica se propone la siguiente funcionalidad adicional que podría implementarse, cuya puntuación se detalla en la tabla posterior. Junto a las mejoras propuestas, se valorarán también de forma positiva otras mejoras que el alumnado quiera plantear al profesorado.

1. **Comando `upload`**: Subir uno de los ficheros compartidos por un *peer* a otro servidor, quien aceptará el fichero entrante siempre que no lo tenga ya entre sus ficheros disponibles.
2. **Comando `serve` con puerto efímero**: Ampliar comando `serve` para que el servidor de ficheros pueda utilizar cualquier puerto disponible (*ephemeral port*) para escuchar, en lugar del 10000/tcp.
3. **Comando `filelist` ampliado**: Mostrar junto a cada fichero disponible, la lista de servidores que lo están compartiendo, además del nombre, tamaño y *hash* del fichero.
4. **Comando `quit` actualiza ficheros y servidores**: Mantener permanentemente actualizada la información sobre ficheros compartidos (comando `filelist`). Si un *peer* que está siriviendo ficheros ejecuta la orden `quit`, deberá comunicarlo al directorio para dar de baja la lista ficheros que comparte.
5. **Comando `download` paralelo**: Ampliar comando `download` para descargar simultáneamente los fragmentos del fichero obtenidos de diferentes servidores de manera simultánea.

La siguiente tabla resume las mejoras propuestas e indica la puntuación adicional que podría obtenerse con la implementación de cada una de ellas.

Mejora	Puntuación máxima
`upload`	1,5 puntos
`serve` puerto efímero	0,5 punto(s)
`filelist` ampliado con servidores	0,5 punto(s)
`quit` actualiza ficheros y servidores	0,5 puntos
`download` paralelo	2 puntos

Detalles de la entrega

El trabajo que los estudiantes deberán desarrollar es el siguiente:

- Programa cliente/servidor *NanoFiles* y programa servidor de directorio *Directory*.
- Documentación de la práctica.

Instrucciones detalladas:

- Las prácticas deben ser realizadas obligatoriamente por **grupos de dos personas**.
- Los grupos deberán subir al Aula Virtual, a la tarea denominada "Práctica de *nanoFiles*" un archivo comprimido .ZIP que contenga lo siguiente:
    - El **código fuente** en Java del proyecto, listo para ser importado y ejecutado en Eclipse.
    - Los programas `Directory` y `NanoFiles` en formato **JAR ejecutable**. Los ficheros se deben llamar **obligatoriamente** `NanoFiles.jar` y `Directory.jar`, y deben funcionar con **Java 21**.
    - La documentación del proyecto. Debe estar en formato de documento PDF e incluir al menos los siguientes apartados:
        - Introducción.
        - Protocolos diseñados.
            - Directorio:
                - Formato de los mensajes y ejemplos de los mismos.
                - Autómatas cliente y servidor
            - Peer-to-peer:
                - Formato de los mensajes y ejemplos de los mismos.
                - Autómatas cliente y servidor
        - Mejoras implementadas y breve descripción sobre su programación.
        - Capturas de pantalla que muestren mediante Wireshark un intercambio de mensajes con el directorio.
        - Opcional: Enlace a grabación de pantalla mostrando los programas en funcionamiento.
        - Conclusiones.
- **El proyecto entregado debe funcionar en los laboratorios de la Facultad**, en el sistema operativo **Ubuntu**, y en un escenario en el que tanto los *peers* como el directorio se ejecuten en **hosts distintos**.

La fecha tope de entrega será el **día 2 de mayo de 2025 a las 23:55**, a través de la tarea del Aula Virtual creada a tal efecto.

Las entrevistas se desarrollarán preferentemente durante la siguiente semana, en el horario y lugar habitual de las clases prácticas, aunque, de ser necesario se añadirán turnos adicionales. Los estudiantes se incribirán para las entrevistas a través de la herramienta *Apúntate*, que cada profesor publicará con suficiente antelación.