Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Minería de Datos

Arizmendi Ramírez Esiel Kevin
Coria Rios Marco Antonio

Proyecto Final: Network Cloud

○Objetivo

Crear un servicio de almacenamiento distribuido para 4 computadoras, escalable y tolerante a fallos. El sistema debe permitir la transferencia de archivos entre todas las máquinas, revisar el sistema de archivos (es decir, el contenido de los directorios permitidos) desde cualquier máquina, y ser transparente al usuario.

Además, las computadoras que fallen deben ser capaces de reintegrarse al sistema y continuar con sus tareas pendientes, actualizando así su sistema de archivos local.

○Propuesta

「*Network Cloud*」, una "nube" formada por las máquinas conectadas entre sí sin depender de un servidor de gestión.

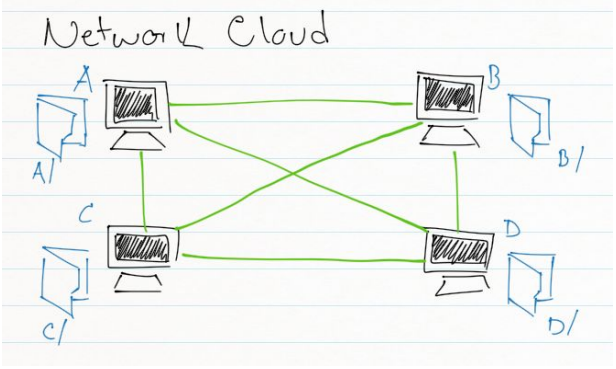
○Diseño

Modelo: Peer-to-Peer (P2P).
Interfaz: Gráfica.
Medio: Sockets.
Protocolo: TCP.

🔹 Nodos

Cada computadora conectada a la red que conforma la nube es un nodo. Todos los nodos están conectados lógicamente entre sí.
Un nodo tiene dos responsabilidades:

- Almacenar archivos y directorios, cual unidad de almacenamiento.
- Proveer al usuario de la interfaz de interacción para manipular los archivos locales y remotos.



Los nodos conforman una nube en sí, distribuyendo el almacenamiento entre ellos, y que se muestran al usuario como si fueran carpetas raíz. Por tanto, cada nodo representa también un usuario del sistema.

🔹 Almacenamiento y Respaldo

Los nodos son encargados de guardar los archivos de los usuarios, así como de proporcionarlos a otros nodos o eliminarlos cuando es solicitado.

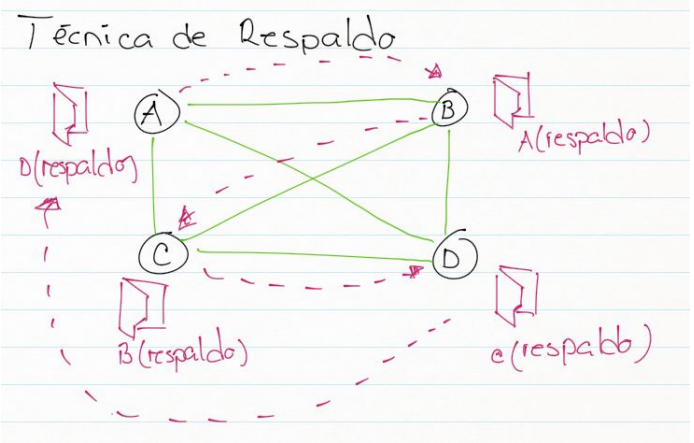
Para esto se hará uso del sistema de archivos nativo, haciendo uso de sus operaciones mediante las interfaces provistas por el lenguaje de programación utilizado. Por tanto, la información que se muestre y maneje de los archivos y directorios será la misma que el sistema operativo local esté registrando.

De esta forma también resulta sencillo "subir" o "cargar" archivos desde el almacenamiento local a los directorios de la nube para que puedan ser transferidos, pues solo sería necesario copiarlos o moverlos a una carpeta de la nube, usando las alternativas que tenga el sistema operativo local.

Esto también facilita la administración de los archivos, ya que se aprovecha el sistema de archivos nativo y no se tiene que realizar un rastreo y control adicional, ni crear estructuras de datos para ello.

❖ Respaldo

Para asegurar la disponibilidad de los archivos y poder realizar operaciones aún cuando algunos nodos no estén disponibles, cada nodo tendrá una copia del almacenamiento de su "vecino anterior".



Cada vez que un nodo sea actualizado, el "vecino siguiente" tendrá que actualizar la copia de respaldo de sus archivos. Los archivos que se generen como copias en el nodo de respaldo serán ajustados para que las fechas de su última modificación coincidan. De esta forma, al actualizar el respaldo solo se enviarán aquellos que hayan resultado con modificaciones desde el último respaldo.

Los archivos de respaldo **no** serán visibles al usuario, sino que el sistema se encargará de recurrir a ellos en caso de que el nodo dueño no se encuentre disponible

Una comunicación directa e independiente de las demás operaciones se dará entre el nodo dueño y el nodo que contiene su respaldo. El nodo dueño se encargará de actualizar su respaldo siempre que esté disponible.

El proceso de respaldo es un proceso independiente de las demás secciones del sistema, por lo que se activa por sí mismo. (ver Hilos/Gestor del Respaldo). El proceso se realizaría a través de un recorrido de profundidad-primero en postorden (adaptado a un árbol no binario); es decir, se visitarán todos los hijos de un directorio antes de visitar el directorio en sí.

En este proceso pueden ocurrir tres situaciones con cada archivo, si el directorio a respaldar también existe en el nodo de respaldo:

- El archivo es **nuevo**. Es decir, no existe en el directorio de respaldo.
- El archivo fue **modificado**. Es decir, las fechas de modificación no coinciden.
- El archivo fue **eliminado**. Es decir, se encuentra en el respaldo pero no en el nodo actual.

Sin embargo, si el directorio actual no existe en el respaldo, quiere decir que todo su contenido tampoco está respaldado. Por ende, se crea el directorio, y: cada archivo en él, se considera **nuevo**, y por cada directorio en él se repite este proceso.

Por último, si el directorio actual sólo existe en el nodo de respaldo, se procede a eliminarlo completamente, junto con todo su contenido.

❖ Archivos del Sistema

Los archivos que necesita el sistema para funcionar se deberán guardar en una localización aparte de la ruta principal, y estar ocultas para el usuario.

Los Archivos del Sistema comprenden:

- Colas de Operaciones: Divididas en Maestra y Subordinadas (ver Comunicación/Colas de Operaciones).

🔹 Comunicación

Los nodos del sistema se comunicarán a través del uso de Sockets TCP para

garantizar la entrega de toda la información en la comunicación.
Sin embargo, cada nodo actúa en sí mismo como un “servidor” y como un “cliente”. Ofrece sus servicios, es decir, las operaciones que le pueden solicitar, como un “servidor”; y por otro lado, solicita operaciones a los demás nodos como “cliente”. Por esto, existe en cada nodo un socket de tipo “servidor”, que crea sockets cuando solicitan la conexión, y una serie de sockets que se conectarán a los sockets “servidor” de los demás nodos.

Los sockets que cree el socket tipo “servidor” serán únicamente para recibir solicitudes, pero ninguna clase de datos saldrá de ellos. Por otro lado, el resto de sockets únicamente enviarán datos y mensajes (ver Mensajes). Las operaciones de cada uno se pueden realizar en paralelo con las de los demás, por lo que un hilo especial gestiona este proceso, teniendo únicamente un solo socket.

Lo anterior hace que se vuelva necesario utilizar mecanismos de protección para evitar incoherencias en las operaciones. Dos transferencias (copia) de salida se pueden gestionar en paralelo sin ningún problema, incluso si es del mismo archivo. Sin embargo, el guardado (escritura, transferencia de entrada) y la eliminación pueden causar problemas si se realizan de forma paralela.

❖ Colas de operaciones
Dado que los nodos son independientes, se pueden gestionar diferentes operaciones de forma paralela. Para esto, cada socket se encuentra en un *Hilo de Gestión*. Además, cada nodo tiene, por nodo en la red (incluido a sí mismo) una *Cola de Operaciones*.

Una cola de operaciones es un archivo de texto que indica la secuencia de operaciones que un nodo debe realizar. Existen dos tipos de colas de operaciones:

- Cola de operaciones subordinada: Se encuentra asociada a un socket del nodo (y por lo tanto, a una dirección IP y puerto específico). Indica las operaciones que tiene que indicarle a otro nodo que realice.
- Cola de operaciones maestra: Se encuentra asociada al nodo en sí, y es el que se encarga de gestionar las operaciones que son indicadas desde el usuario o desde otros nodos. La cola maestra reúne las operaciones y las distribuye entre las colas subordinadas correspondientes, o las realiza localmente, según sea requerido.

Las colas de operaciones se manejan como archivos para que las operaciones a realizar persistan en el tiempo. De esta forma se mantienen *pendientes* operaciones en caso de que el nodo destino se encuentre desconectado. También sirve para recordar la operaciones sin realizar al conectar de nuevo un nodo que se encontraba desconectado.

Nuevamente, mecanismos de protección evitarán que operaciones, por tener largo tiempo de espera, puedan generar errores al volverse incoherentes.

❖ Conexión Dueño-Respaldo
Independientemente de las conexiones entre los nodos, una conexión adicional se dará entre un nodo y el nodo que contiene el respaldo de sus archivos, por cada nodo en el sistema. Esta conexión, al ser independiente de las demás, no contiene una cola de operaciones, y no es gestionada por la cola maestra.

Esta conexión permite que un nodo esté en constante comunicación con su nodo de respaldo, con el fin de mantenerlo actualizado. No obstante, a pesar de que se diga independiente, algunas de las operaciones realizadas en este medio

requerirán bloquear otras, como la eliminación o la recepción de archivos.

La conexión se llevará a cabo a través del modelo Cliente-Servidor, solo que en este caso, el dueño de los datos respaldados actúa como cliente, y el que se encarga de almacenarlos actúa como Servidor.

🔹 Operaciones (al usuario)

Las siguientes son las operaciones que son mostradas al usuario a través de la interfaz. Estas operaciones pueden representar varios pasos para el sistema.

❖ Transferir archivo(s)
Realiza una copia de los archivos seleccionados (o dentro de un directorio) del nodo fuente en el nodo destino. Los archivos no se eliminan del nodo fuente.
Si los archivos ya existían en el destino, los datos de éstos se reemplazan, y se actualiza su fecha de modificación.

Tanto fuente como destino pueden ser cualquier ruta de cualquier nodo.

❖ Eliminar archivo(s)
Elimina del nodo indicado los archivos y/o directorios que fueron seleccionados. Los archivos eliminados no son rescatables.

❖ Crear directorio
Crea una nueva carpeta (directorio) en la ruta del nodo indicado.

❖ Listar Contenido de Directorio
Muestra los archivos y directorios que se encuentran en el directorio especificado.

🔹 Operaciones (del sistema)

Las siguientes son las operaciones que se cargan en las colas de operaciones y que son interpretadas por los nodos. Las operaciones al usuario son combinaciones de éstas.
(✱ Recordar que todas las operaciones se cargan primero en una cola maestra, que se encarga de gestionarlas.)

❖ Transferencia de archivos
Ubicación: Cola subordinada.
Envía el mensaje a otro nodo de que debe transferir (copiar) los datos de uno o varios archivos a otro nodo.
(✱Esto no implica que los nodos comunicados sean la fuente o el destino del archivo a transferir.)
Parámetros (4): Nodo Transmisor/Nodo Receptor/Ruta en el Nodo Transmisor del archivo a enviar/Tipo de Información a Transferir
Nota: El *Tipo de Información a Transferir* corresponde a si se está solicitando los datos de un archivo o el contenido de un directorio. De esta forma se solicita el contenido de un directorio remoto. El *Tipo de Información* se propaga a la Operación "Envío de archivos" (ver ↴)

❖ Envío de archivos
Ubicación: Cola subordinada.
El nodo que cargue esta operación es la fuente del archivo. El nodo receptor del mensaje es el nodo destino.
Envía el mensaje a otro nodo para que se prepare y reciba uno o varios archivos.
Parámetros (3): Nodo Receptor/Ruta local del archivo a enviar/Tipo de Información a Enviar

❖ Eliminación de archivos
Ubicación: Cola subordinada (remoto); Cola maestra (local).
Si la operación se indica local, entonces se carga en la cola maestra para eliminar el archivo.

Por el contrario, si la operación es remota, se carga en la cola subordinada correspondiente. Envía el mensaje para eliminar el archivo correspondiente. (Finalmente la operación se cargará en la cola maestra del nodo indicado).

Parámetros (2): Nodo/Ruta (local o remota) del archivo o directorio a eliminar

❖ Creación de Directorios
Ubicación: Cola subordinada (remoto), Cola maestra (local).
De manera similar a la eliminación. Si la operación es local, se carga a la cola maestra para crear el directorio indicado.
Si la operación es remota, se carga en una cola subordinada para enviar la indicación al nodo correspondiente. (Eventualmente, terminará en la cola maestra del nodo indicado)
Parámetros (2): Nodo/Ruta (local o remota) del directorio a crear

❖ Listado de Contenido de Directorio
Ubicación: Cola Maestra
Obtiene la lista de archivos y directorios dentro del directorio solicitado en la ruta del parámetro.

Nota: Esta operación es para uso únicamente **local**. Esto se debe a la diferencia del proceso para mostrar el contenido de un directorio local al de un directorio remoto.
Para listar el contenido de un directorio local basta con utilizar la interfaz que provee el lenguaje de programación.
Sin embargo, para listar el contenido de un directorio remoto es necesario hacer una solicitud al nodo dueño del directorio, la cual deberá ser contestada. Entonces, el listado del contenido de un directorio remoto se hace mediante una solicitud de Transferencia especial (ver Transferencia de archivo/Nota).

Parámetros (1): Ruta del directorio a mostrar

La ventaja de las operaciones es que son transferibles entre nodos. Cuando una operación debe ser realizada por otro nodo que no es el que posee la operación, entonces transfiere esta información al nodo correspondiente. Con la información recibida, el nodo receptor estructurará una nueva operación, sea del mismo tipo o distinta, la cual ya representa una operación local para él.

Ninguna operación realiza más de un salto para ser completada. Una operación puede ser *local* o *remota*, y en el caso de esta segunda, sólo se deberá realizar una transferencia de la operación para que se vuelva *local*, a excepción de la operación para listar el contenido de un directorio, la cual no puede ser *remota*. La transferencia también es un caso especial, ya que ésta no puede ser *local*.

🔹 Hilos
Para llevar a cabo la gestión del nodo, determinadas tareas se realizarán simultáneamente a través de hilos.

❖ Interfaz del usuario: Se encargará de gestionar la comunicación con el usuario (recibir indicaciones) y mostrar los resultados (exitosos o fallidos).
Las acciones que reciba por parte del usuario serán cargadas en el archivo de la cola de operaciones maestra.

❖ Cola maestra: Éste se encargará de administrar las operaciones cargadas en el archivo de la cola maestra, y, en dado caso, distribuir las operaciones en los archivos de las colas subordinadas.

❖ Envío de mensajes: Conectado únicamente con otro nodo; uno por cada nodo conectado a la red. Se encargará de enviar los mensajes a través del socket correspondiente, para realizar operaciones remotas.

❖ Recepción de mensajes: Conectado únicamente con otro nodo; uno por cada nodo en la red. Recibe los mensajes desde nodos remotos y carga las operaciones indicadas en el archivo de la cola maestra.

❖ Punto de Conexión: Hilo que se encarga de actuar como “servidor” para recibir las

conexiones de los demás nodos y formar los hilos para Recepción de Mensajes.

❖ Gestor del Respaldo: Un hilo independiente de los demás mencionados. Se encarga de administrar la comunicación Dueño-Respaldo, para mantener actualizado el respaldo de los archivos locales. Al funcionar de forma independiente, se ejecuta de forma asíncrona a las operaciones del sistema.
El Gestor del Respaldo se comporta a manera de un *demonio*, realizando sus operaciones cada cierto tiempo, bloqueando otras que sean necesarias para evitar inconsistencias. Sólo él conoce la vía de comunicación (el socket) Dueño-Respaldo, y por tanto también es quien actualiza el respaldo tras una reconexión de éste.
Además, cada vez que realiza un respaldo, independientemente si se pudo completar o no, vuelca la información de las tablas de archivos a la memoria secundaria.

❖ Esclavo del Respaldo: Así como el gestor, el *esclavo* es el hilo que administra el respaldo de otro nodo guardado localmente. Se mantiene a la espera de instrucciones del Gestor del Respaldo del nodo remoto, y actuará solo cuando éste se lo indique.
Esto último quiere decir que, tras una reconexión, el nodo respaldo solicita al dueño la actualización, sino es el nodo dueño quien está a la espera de la reconexión para actualizar el respaldo.

🔹 Protección de operaciones

Para garantizar que las operaciones se realizarán adecuadamente y no generarán fallos se siguen las siguientes reglas:

1. Un nodo puede enviar archivos a dos o más nodos simultáneamente (Incluye la descarga de archivos).
2. Un nodo que se encuentra enviando archivos no puede ejecutar operaciones de eliminación.
3. A su vez, un nodo eliminando archivos no puede enviar o realizar descargas hasta completar la eliminación.
4. Un nodo sólo puede llevar a cabo una operación de eliminación a la vez.
    a. Sin embargo, se pueden eliminar varios archivos o directorios en esa operación.
5. Un nodo sólo puede recibir un archivo a la vez (Incluye la subida de archivos).
6. Un nodo puede recibir y enviar simultáneamente (Incluye la subida y la descarga de archivos, pero no la combinación de estas dos simultáneamente).

Lo anterior considera al “envío” de archivos locales tanto a otras ubicaciones como al respaldo a través del Gestor del Respaldo.

❖ Semáforos
La sincronización de los hilos de un nodo se llevará a cabo a través de semáforos. Éstos controlarán el acceso a los archivos de la colas de operaciones y la ejecución de operaciones en sí.

- Semáforo de cola de operaciones: Por cada archivo de cola de operaciones se tendrá un semáforo *mutex*; es decir, solo un hilo puede acceder al archivo a la vez.
- Semáforo(s) de eliminación: Sincronizan el envío de archivos hacia nodos remotos con la eliminación de archivos locales.
  Dado que se pueden enviar varios archivos simultáneamente pero solo se puede ejecutar una eliminación a la vez, y ambas operaciones se restringen mutuamente, este problema se resolverá mediante la técnica de *Lectores y Escritores*, donde los *Lectores* representan los envíos de archivos y los *Escritores* representan las operaciones de eliminación.
  La prioridad se dará a los *Escritores*.

- Semáforo de recepción: *Mutex*. Sincroniza a los hilos para recibir únicamente un archivo a la vez. También sincroniza al Gestor del Respaldo. El orden de acción será tal cual haya sido solicitado el semáforo.

Los semáforos gestionan las colas de operaciones y las operaciones en sí con el objetivo de evitar incoherencia en los datos del sistema. Sin embargo, no gestionan problemas como el intento de envío de un archivo que ya fue eliminado (el semáforo evitará que estas dos operaciones se realicen a la vez). Estos problemas se resolverán por medio de manejo de excepciones.

A su vez se utiliza el semáforo de gestión de recepción y eliminación para evitar modificar los archivos mientras el Gestor del Respaldo esté llevando a cabo sus operaciones. Esto, debido a que el Gestor requerirá de obtener los datos de todos los archivos que requieran actualizarse en el respaldo, y por tanto no se les puede modificar o eliminar en el proceso.

🔹 Estructuras de Datos

❖ Operaciones

Las operaciones son simples, pero para facilitar su manejo se incluyen como una estructura de datos, compuesta de:

- Tipo: Según sea el tipo de operación (ver Mensajes).
- Parámetros: Una cadena de caracteres que determina la información necesaria para completar la operación. (ver Operaciones del Sistema)

🔹 Configuración y Escalabilidad

En este documento se presenta el caso concreto con el sistema utilizando 4 nodos. Sin embargo, es posible tener desde 2 de ellos, o querer incrementar el número de nodos para aumentar el almacenamiento y la cantidad de usuarios.

La configuración del sistema se divide en configuraciones individuales para cada nodo, de tal forma, que a cada nodo se le tiene que indicar las ubicaciones en la red de los demás nodos a los que va a conectarse, quién va a ser su respaldo, y a quién respaldará él.
Esta información se especificará en un archivo de texto, utilizando el formato JSON, para indicar todas las propiedades que debe tener un nodo.

Los parámetros configurables son los siguientes:

- Nombre o identificador del nodo.
- Ruta del directorio para la raíz del sistema. Se puede utilizar cualquier directorio; el sistema nunca accederá a un directorio superior, todos los archivos necesarios se almacenarán a partir del directorio especificado.
- Puerto del servidor local para recibir conexiones de los demás nodos.
- Nodos remotos: Lista de información de los demás nodos que conforman el sistema.
  Por cada nodo se debe tener:
  - Nombre o identificador
  - Dirección IP.
  - Puerto de conexión remoto.
- Nodo de respaldo: Información del nodo que se encargará de respaldar la información almacenada en este nodo.
  - Dirección IP.
  - Puerto de conexión remoto.
  - Intervalo de actualización (en segundos)
  - Modo de operación al establecer conexión: Tres modos, “Sobreescribir y actualizar respaldo”, “Recuperar respaldo”, “No realizar operaciones al establecer conexión”.
- Nodo respaldado: Información del nodo dueño de la información que se respalda en este nodo.
  - Puerto de conexión local.

Nótese que la configuración se debe realizar de forma manual, y no se gestiona automática por el sistema. Esto se debe a que no se ocupa ningún punto centralizado, y por lo tanto cada nodo es independiente en sí. Antes de levantar un sistema, es importante realizar un esquema de la configuración para evitar problemas de conexión.

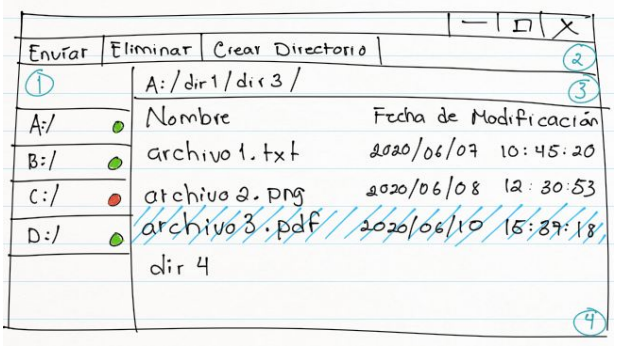
La configuración se establece por medio del archivo y una vez arrancado el sistema en un nodo no hay forma de cambiarla. Por ello, si se requiere hacer un cambio, es necesario apagar el sistema en todos los nodos conectados, y cambiar su configuración para hacer coherente el cambio.

De esta forma se consigue la escalabilidad. Aunque el sistema no se actualiza de forma dinámica para recibir el nuevo nodo, a través de los parámetros de configuración es posible reconfigurar un sistema ya existente para aceptar un nuevo nodo.
Nótese que, este nuevo nodo tendrá que respaldar a otro, y a su vez será respaldado. Estos cambios sí son actualizados automáticamente, y no se requiere mover los respaldos de forma manual. Cuando el sistema total se restablezca, las operaciones habituales (como verificar el respaldo) eliminarán las inconsistencias que genere el nuevo nodo, y crearán la nueva estructura requerida.

De esta forma, no es lo mismo hablar de una configuración de N nodos con K nodos ausentes (desconectados) que una configuración de N-K nodos, ya que el sistema sigue considerando a los K nodos ausentes para las operaciones que se deban realizar.

- **Interfaz de Usuario**

La interfaz para comunicar el sistema con el usuario será una interfaz gráfica. El diseño de la vista principal es el siguiente:



① Panel de Unidades de Almacenamiento.
O también panel de nodos, es la sección donde se muestran cada uno de los nodos que se conectó (o se intenta conectar) el sistema. Se muestra también el estado de conexión, que se obtiene a través del hilo de recepción. (ver Hilos/Recepción de Mensajes)

② Panel de Acciones
Botones que permiten interactuar con los archivos y los directorios. Corresponden con las Operaciones al Usuario: Enviar (Transferir una copia de un nodo a otro), Eliminar, y Crear un Directorio en la ruta actual.

③ Panel de la Ruta Actual
Muestra la ruta de directorios para llegar hasta el que se muestra en pantalla. También mostrará la opción de “Subir de Directorio”, para regresar al contenido del directorio padre.

④ Panel Principal.
Aquí se muestran tanto los archivos y directorios que contiene un directorio, como ciertos estados del sistema.

Al desplegar el contenido de un directorio, permite seleccionar y activar las acciones correspondientes. Muestra el Nombre y la última Fecha de Modificación de los archivos y directorios.

Los estados del sistema que puede mostrar son: el momento en el que el sistema se está iniciando y; el momento en el que se está

esperando la información de un directorio remoto.
Esto es así para dar información al usuario, ya que son procesos que pueden tardar algo de tiempo en realizarse, y no mostrar cambios inmediatamente.
En el caso del arranque del sistema, todas las acciones quedan bloqueadas.

○ Implementación

Paradigmas: Orientado a Objetos, MVC
Lenguaje de Programación: Java
Versión: 8

Para facilitar la portabilidad del sistema, volviéndolo independiente del sistema operativo, se decidió implementarlo utilizando Java, el cual provee nativamente bibliotecas para el manejo tanto de sockets para la comunicación entre computadoras, como de hilos para los procesos concurrentes.

Java no tiene bibliotecas nativas para el manejo de archivos JSON, sin embargo, sí existen bibliotecas externas que se pueden agregar para utilizarlas dentro del programa, por lo que se utilizará la biblioteca *org.json* para este propósito.

La implementación se realizó siguiendo el diseño de clases (ver *NetworkCloud-UMLClassDiagram.pdf*) el cual representa principalmente el funcionamiento interno del sistema (es decir, se detalla únicamente el *Modelo*).

🔹 Modelo

Como se menciona en la sección del Diseño, el Modelo se divide en seis tipos de hilos (que heredan de la clase *Thread*), que se encargan de realizar las funciones principales del sistema, siendo el más importante de todos ellos el núcleo, definido en la clase *CloudCore*.

El núcleo del sistema se encargará de configurar todas las demás secciones del sistema, revisar las Operaciones en la cola de operaciones maestra, delegarlas o realizarlas según sea, y de notificar al Controlador para realizar cambios en la Vista. Aquí se contienen, además, los semáforos de eliminación y de recepción, indicados en la sección de Diseño/Protección de Operaciones.

Por otro lado, *RemoteSender* corresponde al hilo de Envío de Mensajes; *RemoteReceiver* al hilo de Recepción de Mensajes; *ConnectionPoint* al Punto de Conexión, *BackupAdmin* al Gestor del Respaldo; y *BackupSlave* al Esclavo del Respaldo.

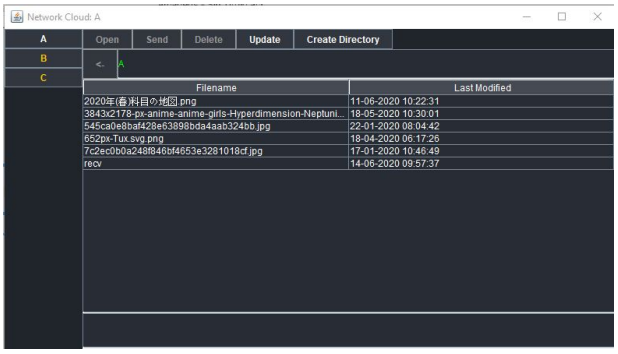
Las demás clases no heredan de *Thread*, y simplemente representan las estructuras de datos que utiliza el sistema. Se utiliza una representación adicional, que no es propia de UML, para representar el flujo hacia las colas de operaciones, así como para representar la comunicación entre sockets.

Una flecha punteada, con una o dos puntas blancas (sin símbolo del otro lado en caso de solo una) representa flujo de datos de una clase a un archivo. Por otro lado, una flecha punteada con una sola punta blanca y una línea en el extremo contrario se refiere al flujo de datos de un socket.

Aunque están expresadas en el diagrama, las flechas de comunicación por sockets no hacen referencia a una interacción de ambas clases en el mismo proceso. Dicha flecha representa la interacción de ambas clases en procesos diferentes, iniciados a partir del mismo código fuente.

🔹 Vista

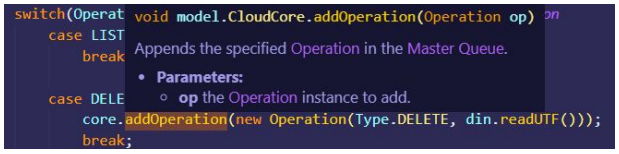
Finalmente, la interfaz se implementó utilizando también la biblioteca nativa de Java para interfaces gráficas de usuario *Swing*, dividiéndola en componentes según se especificó en el Diseño. No se detallan, sin embargo, las clases que la conforman, debido a la gran cantidad de componentes que se necesitan.



Se añadieron algunas opciones al Panel de Acciones, para mejorar la interacción con el usuario. La opción “Open” se activa con los directorios, y sirve para acceder a las carpetas visibles. La opción “Update” sirve para volver a solicitar el contenido del directorio actual y actualizar la información que se tiene.

🔹 Documentación

Adicionalmente, se utilizarán los comentarios de documentación de Java para que la información sobre cada clase, atributo y función sea visible en cualquier parte del código. Esto, por supuesto, con la ayuda de un entorno de desarrollo integrado o un editor de texto especializado para desarrollo de software con soporte para Java.



Para más detalles sobre las clases, sus constructores y métodos (parámetros, retornos, excepciones), y sus atributos es necesario revisar directamente el código.

○Limitaciones

El diseño del sistema jugó un factor clave en el desarrollo del mismo, debido a que requirió una gran parte del tiempo para completarse. Se trató de definir cada detalle con el fin de facilitar la programación, pero al considerar la tolerancia a fallos y el rendimiento, y con el fin de no propagar errores de diseño a la implementación, el tiempo requerido fue más del estimado.

Por ello, la implementación total del diseño no fue completada, pero sí se consiguió una versión estable y funcional, por lo que se considera el producto final de este trabajo como un prototipo del sistema.

🔹 De las funcionalidades

Aunque en este documento se describe la capacidad de enviar múltiples archivos en una sola operación (no simultáneamente), en la implementación final no contiene tal funcionalidad. En el producto final, solo es posible enviar un archivo a la vez. Esto mismo aplica a enviar directorios, pues significa enviar su contenido, que puede estar compuesto por varios archivos.

Aunado a ello, tampoco es posible transferir el archivo a una ruta específica de cualquier nodo remoto. Los archivos se transfieren a un directorio predefinido por el sistema, y el cual también es visible desde el propio sistema.

Las funcionalidades planeadas en el diseño no son imposibles de programar, y la estructura final del sistema permite su implementación. No obstante, debido a los tiempos establecidos para su desarrollo, se ha tenido que optar por estas funcionalidades más sencillas, con el fin de tenerlas a tiempo y libres de errores.

🔹 Del respaldo

En este documento también se describe el diseño y la estrategia para llevar a cabo el respaldo de la información de cada nodo, pero tampoco se alcanzó a implementar, por lo que los archivos en cada nodo no se respaldan en ningún lado.

Al igual que con las funcionalidades reducidas, el sistema se encuentra preparado para realizar una implementación de la estrategia de respaldo.

○Conclusiones

Desarrollar un sistema distribuido completamente y tolerante a fallos es una tarea que requiere mucho tiempo en su diseño. Definir un sistema en su diseño como “tolerante a fallos” implica que todos los fallos que puede provocar están contemplados de antemano, se conocen, y se debe diseñar una estrategia para darles solución o soporte, que además sea transparente al usuario. Además, en un sistema distribuído los fallos pueden ser mucho más graves, al punto de una o varias de las partes que conforman al sistema se vuelvan inutilizables.

No considerar alguno de estos fallos y permitir que se propague a las siguientes fases del desarrollo de software significa aún mayor consumo de tiempo del que significaba resolverlo en la fase de diseño.

Adicionalmente, las estrategias para resolver los fallos deben considerar que no existe una unidad centralizada, y que deben actuar con las partes del sistema que quedan disponibles. Esto se vuelve aún más complicado de diseñar, ya que la cantidad de escenarios posibles puede dispararse, y, dependiendo del fallo o del sistema, pueden no existir formas tan generales para agruparlos.