Escuela de Ciencias de la Computación
Laboratorio 04 2021-II
CC4P1 Programación Concurrente y Distribuida

A día de hoy gran parte de los sistemas son centralizados y poco transparentes. Sin embargo, ya hace casi una década se ha comprobado los beneficios que se puede obtener al usar un sistema distribuido, transparente y robusto como lo es la blockchain. Tanto es así que hay muchas empresas bancarias que han invertido para su desarrollo, teniéndose muchas propuestas como un sistema monetario distribuido, elecciones electorales electrónicas, banco de patentes, bancos genéticos, contratos inteligentes, etc.

Concurrencia de Recursos Distribuidos

Simular una cadena de bloques para transacciones bancarias aplicando el concepto de prueba de trabajo (PoW) y merkle tree en un cluster de “n” nodos.

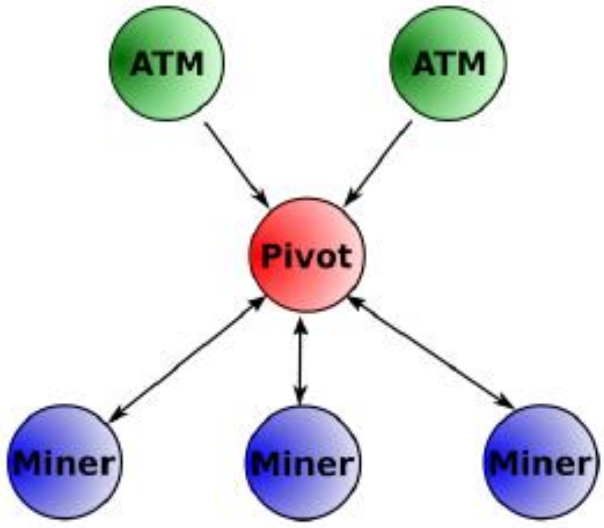
Implementar las funciones de conexión entre entes, además de un pool de transacciones aplicando merkle trees para evitar exceso de generación de bloques.

Implementar la generación de bloques y validación como aplicación de la prueba de trabajo (PoW) de manera paralela para reducir el tiempo de ejecución.

Almacenar transacciones, bloques y merkle trees para su posterior revisión y/o auditorías.

Para la implementación de la blockchain se propone trabajar con tres entes que simulan su comportamiento.

Para ello, se cuenta con un servidor llamado Pivot, unos nodos mineros llamados Miner y unos nodos clientes llamados ATM



El modo de operación se dará de la siguiente forma:
1. Los nodos ATM mandarán transacciones al servidor Pivot (ejm, enviar monedas de un nodo a otro).
2. Si la transacción es válida, el servidor Pivot la almacenará en un pool de transacciones.

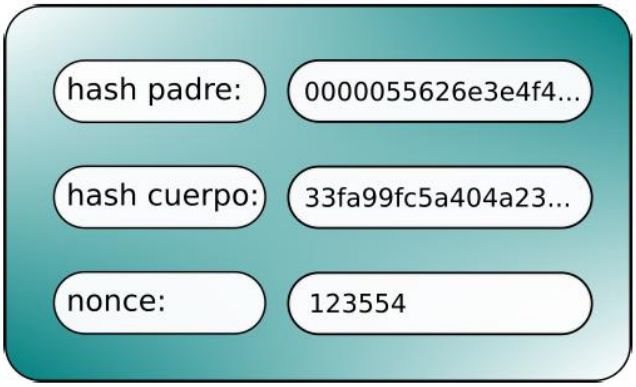
3. Cuando se tenga un monto límite de transacciones se mandará en bloque a los mineros Miner para que puedan conseguir un bloque válido.

Para el caso de las transacciones se usará el formato como se muestra a continuación:

“Cliente$_1$” > “Cliente$_2$” “Monto”

Donde Cliente1 hace referencia al cliente que desea enviar una transferencia, el Cliente2 hace referencia al cliente destino y Monto hace referencia a la cantidad que se desea transferir entre los clientes.

En cada transacción se valida si el Cliente$_1$ cuenta con el saldo suficiente para poder emitir el Monto.
Para la validación de un bloque se tomaron 3 parámetros



La primera en relación al hash del padre, la segunda al hash de las transacciones y por último el nonce, que es el término que se obtiene por medio de la prueba de trabajo (PoW).

Para la obtención del hash de las transacciones se usará la estructura de Merkle, siendo el hash de la raíz la que se usará como hash resultante.

Éstos tres parámetros se concatenaron para obtener un hash (haciendo uso de sha1). Ya que el único parámetro que varía es el nonce por medio de la prueba de trabajo se espera en algún momento obtener un hash válido.

Ya que se tenía como propósito académico la emulación se realizará con una dificultad baja, el número de ceros será una **constante** que se pueda variar, en este caso de 5 (con ceros=5 iniciales), a continuación, se muestra un hash válido con dificultad 5.

00000286e3e4f4fba5d0448333fa99fc5a404a73

Cada nodo al recibir las transacciones iniciará con un nonce aleatorio e irán incrementando este valor hasta obtener un hash válido. Se inicia con un nonce aleatório con el propósito de diferenciar el tiempo de cálculo de cada minero. Una vez calculado, el nodo minero comunica al servidor que encontró una solución y éste avisará a los otros nodos Miners, para que se detengan y dejen de calcular. Luego, se envía el nonce para que puedan comprobar y confirmar que el hash obtenido por el nodo ganador es el correcto.
Finalmente se archiva en un fichero los resultados, para así tener un historial de los bloques generados.

También se generará un fichero con la estructura de Merkle de cada bloque para verificar que todo funcionaba correctamente.

Se le pide lo siguiente:

- Se pide escribir un código en LP1 y LP2, exponer y redactar un informe de mínimo 3 hojas.
- El Pivot = LP1, los Mineros = LP2, donde LP1 <> LP2
- Los Mineros deben de minar de manera concurrente paralela, calcular el número de hilos adecuado.
- Al iniciar el servidor Pivot (Servidor) este recibirá las peticiones de Clientes (ATM) y luego distribuirá las peticiones a los diferentes nodos Miners.
- Tomar como base las explicaciones y el código de clase.
- Exponer y Ejecutar en su clusters y en LP1 y LP2 para poder realizar la comparación y resultados.

Presentación:

- Subir en un Comprimido
- Comprimido consta
    - Clases en extensión de LP1 y LP2
    - PDF Informe
- Evaluación del sistema
    - Evaluar el desempeño con el mayor número de nodos en red