Capítulo 2

Capa de Protocolo Base de Redes Blockchain

Requisitos

- Asumimos que hay un **registro de transacciones.**
- Veremos ciertos **requisitos a alcanzar** por el sistema en lo que se refiere al registro.
- Luego veremos que usar cadenas de bloques es una **solución**.
- **Requisitos**:
  - **Registro de transacciones:** capacidad de almacenar transacciones.
  - **Consistencia del estado del sistema:** Todos los participantes deben tener una visión unificada del estado actual del sistema.
  - **Descentralización**: queremos que el registro opere sin una autoridad central que controle el sistema.
  - **Inmutabilidad**: una vez que las transacciones se agregan al registro, no pueden ser modificados ni eliminados.
  - **Seguridad**: los datos del registro deben estar protegidos contra alteraciones y accesos no autorizados.
  - **Transparencia**: todos los participantes deben poder ver y verificar las transacciones y los datos en el registro.
  - **Consenso**: los nodos de la red deben acordar la validez de grupos de transacciones antes de agregarlas al registro.

Requisitos

Requisitos – cont:

- **Escalabilidad**: el registro debe ser capaz de manejar un número creciente de transacciones y nodos sin una disminución significativa en el rendimiento.
- **Rendimiento**: el tiempo de procesamiento de las transacciones y la actualización del registro debe ser eficiente.
- **Resiliencia**: el sistema debe ser robusto y capaz de recuperarse rápidamente frente a fallas o ataques.
- **Privacidad**: debe garantizarse la confidencialidad de ciertos datos y transacciones cuando sea necesario.

Solución: usar una **cadena de bloques** (blockchain).

- Es una estructura de datos descentralizada y cronológica que almacena información en forma de bloques.
- Cada **bloque** contiene un conjunto de transacciones.
- Se tiene una **red de nodos distribuidos** donde cada nodo tiene una copia completa de la blockchain.

Cadenas de bloques

Solución – cont:

- El **hash de un bloque**: Es un identificador único del bloque generado mediante un **algoritmo criptográfico**.
  - Funciona como una "huella digital" del bloque y cambia si se modifica cualquier dato del bloque.
  - Un hash hace extremadamente difícil alterar un bloque sin ser detectado.
- Los bloques de una cadena de bloques están enlazados mediante **hashes**.
  - Cada bloque contiene el hash del bloque anterior.
- Todos los participantes pueden ver las bloques de la blockchain.
- Se **usan mecanismos de consenso** para asegurar que los nodos acuerden la validez de los nuevos bloques.

Solución – cont:

- **Estructura de un bloque:**
  - **Encabezado del Bloque**: Contiene metadatos cruciales para la integridad y verificación.
  - **Cuerpo del bloque**: Almacena las transacciones realizadas.
  - **Hash del bloque** generado a partir de todos los datos contenidos en el bloque.
    - Este hash garantiza que cualquier cambio resultaría en un nuevo valor completamente diferente,
    - protegiendo así la integridad e inmutabilidad del registro blockchain

Cadenas de bloques

Solución – cont:

- **Encabezado del bloque:**
  - **Hash del bloque anterior.**
  - **Merkle Root**: es hash que resume todas las transacciones dentro del bloque.
  - **Nonce**: número aleatorio usado durante el proceso de minería para encontrar un hash válido.
  - **TimeStamp**: marca temporal indicando cuando se creó el bloque.
    - Proporciona orden cronológico de los bloques.

Cómo se logran los requisitos:

- **Distribución:** uso de varios nodos con copia de blockchain.
- **Inmutabilidad**: los bloques no pueden alterarse una vez agregados a la blockchain, cualquier cambio sería detectable porque alteraría el hash del bloque.
- **Transparencia**: todas las transacciones son visibles públicamente.
- **Consenso**: por medio de los mecanismos de consenso.

Cadenas de bloques

Cómo se logran los requisitos - cont:

- **Rendimiento**: la eficiencia en la creación de bloques y la validación de transacciones depende de la implementación específica de la blockchain y su algoritmo de consenso.
- **Privacidad**: se pueden implementar mecanismos para la privacidad como transacciones confidenciales.
- **Escalabilidad**: es un desafío por eso se desarrollaron soluciones de escalabilidad y otras tecnologías.
- **Seguridad**: se emplean algoritmos criptográficos para proteger los datos y las transacciones.
  - Además el uso de mecanismos de consenso como proof-of-work o proof-of-stake asegura que se necesita una cantidad significativa de recursos para comprometer la red.
  - La descentralización ayuda: un atacante tendría que comprometer mas del 50% de los nodos.
- **Consistencia**: mediante mecanismo de consenso todos los nodos acuerdan qué bloque es el siguiente en añadirse a la cadena.

Mecanismos de consenso

- **Principales mecanismos de consenso:**
  - **Proof of Work (PoW)**: Hay **nodos mineros** que compiten por resolver **problemas criptográficos** complejos.
    - El primero en resolverlo valida un bloque y recibe recompensas.
    - Hay un alto costo energético necesario para alterar bloques. Puede ser lento.
  - **Proof of Stake (PoS)**: Hay **nodos validadores** que son elegidos según su participación en la red.
    - Los nodos validadores verifican si las transacciones dentro de un bloque propuesto son válidas y si cumplen con las reglas de la red.

- **Proof of Stake – cont:**
  - Después de validar las transacciones, los nodos validadores **crean** nuevos bloques.
  - Cuando un nodo validador **propone** un nuevo bloque, otros nodos validadores revisan y validan ese bloque.
  - Luego solo los bloques válidos (aprobados por al menos 2/3 de los validadores) serán propagados por la red y añadidos a la blockchain.
  - Los validadores mantienen **copia** de la blockchain.
  - Los validadores reciben recompensas.
  - Tienen menor consumo energético que PoW. Puede concentrar poder entre grandes stakeholders.

Mecanismos de consenso

- **Principales mecanismos de consenso - cont:**
  - **Delegated Proof of Stake (DPoS)**: Los usuarios votan por **delegados** para validar bloques; estos delegados reciben recompensas por su trabajo.
    - Los delegados pueden validar y agregar nuevas transacciones a la blockchain.
    - Esto incluye validación de bloques y confirmación de transacciones.
    - DPoS es rápido y eficiente. Permite votaciones directas por parte del usuario final.
    - DPoS puede ser menos descentralizado si pocos delegados dominan las votaciones.

- **Byzantine Fault Tolerance (BFT):**
  - Un **líder** propone nuevos bloques mientras otros nodos **verifican su validez** antes del consenso generalizado.
  - Luego de la validación, se hace una votación por nodos participantes del consenso,
    - para determinar si aceptan o rechazan el bloque propuesto.
    - Para alcanzar el consenso se requiere que mas del 66% de los nodos honestos estén de acuerdo.

Mecanismos de consenso

- **Byzantine Fault Tolerance (BFT) - cont:**
  - Existen mecanismos para detectar nodos deshonestos e ignorarlos durante el proceso de consenso..
  - BFT garantiza alta velocidad y tolerancia a fallos bizantinos, incluso con presencia significativa de actores maliciosos.
  - BFT requiere confianza inicial en el líder o estructura jerárquica establecida dentro del sistema.

Bitcoin

- **Características de Bitcoin:**
  - **Blockchain pública:** cualquiera puede unirse a la red, ejecutar un nodo y verificar transacciones.
  - **Permissionless:** No requiere permisos para participar como usuario, minero o nodo.
  - Bitcoin usa el mecanismo de consenso **Proof of Work**.
  - **Oferta limitada**: solo habrá **21 millones de bitcoins**,
    - lo que crea escasez y potencialmente valor a largo plazo.
  - Las transacciones suelen resolverse en minutos, aunque pueden tardar más dependiendo de la congestión de la red.
  - **Estructura de incentivos:** Los mineros reciben recompensas en BTC y tarifas de transacción para mantener la red segura y operativa.
  - **Tiempo de bloque:** Aproximadamente 10 minutos en promedio para minar un nuevo bloque.

- **Servicios provistos:**
  - **Transferencia de valor**: permite la enviar y recibir bitcoins de manera rápida y segura entre pares (P2P), sin intermediarios como bancos.
  - **Almacenamiento de valor**: debido a su oferta limitada y su seguridad, Bitcoin es considerado como una reserva de valor similar al oro.
  - **Pagos internacionales**: facilita pagos internacionales con menores costos y tiempos de procesamiento comparado con los métodos tradicionales
  - **Seguridad y verificación de transacciones**: usa un sistema público y transparente para evitar fraudes y dobles gastos.
  - **Bitcoin script:** Soporta contratos inteligentes básicos a través de su lenguaje de scripting, aunque con restricciones para mantener la seguridad.

Bitcoin

- Un bloque tiene un tamaño máximo que por ahora es de 1 MB,
  - lo que limita la cantidad de transacciones por bloque.
- Cada usuario tiene **un par de claves**:
  - **Clave privada** secreta.
    - Se genera un número aleatorio de 256 bits que es difícil de adivinar.
    - Generalmente las billeteras digitales generan esta clave privada.
  - **Clave pública** compartida en la red.
    - Se aplica una formula matemática a la clave privada para generar la clave pública correspondiente.
    - Generalmente se usa para ello el algoritmo ECDSA (Elliptic curve digital signature algorithm)

- **Problema**: Hay que garantizar que se cumplen los siguientes requisitos:
  - **Integridad**: Hay que garantizar que los datos (transacciones, bloques) no han sido alterados.
  - Es necesario generar **direcciones de Bitcoin** que son una forma de identificar las cuentas para poder enviar y recibir bitcoins.
  - Proteger la red contra ataques y asegurar la **confidencialidad de los datos.**
  - Hace falta un **mecanismo para validar** que los mineros han realizado una cantidad significativa de trabajo computacional antes de añadirse un nuevo bloque a la blockchain.

Bitcoin

- **Solución**: Usar una función de hash.
  - Una **función hash** es un algoritmo matemático que transforma datos de entrada de longitud variable en una cadena alfanumérica fija y única, conocida como código hash.
  - **Propiedades de una función de hash:**
    - **Determinismo**: una función de hash produce siempre el mismo resultado para una entrada dada.
    - **Eficiencia**: las funciones de hash deben ser rápidas para calcular.
    - **Resistencia a colisiones**: es prácticamente imposible encontrar dos entradas diferentes que produzcan el mismo código de hash.
- **SHA-256** (**Secure Hash Algorithm 256**) es un tipo específico de función hash criptográfica que se usa en Bitcoin y produce un código de hash de 256 bits (64 caracteres hexadecimales).
- Para **generar una dirección pública**
  - la clave pública pasa por dos funciones hash secuenciales:
    - Primero SHA-256.
    - Luego RIPEMD160.
  - Esto produce un hash que luego es **codificado en Base58Check** para crear una cadena alfanumérica.
- Cada bloque y transacción en la red Bitcoin **se identifica** mediante un hash SHA-256 único.

Bitcoin

- **Solución – cont:**
  - En Bitcoin SHA-256 se aplica a los datos de la transacción para crear un hash único que representa esa transacción específica.
  - Cada bloque contiene en su encabezado el hash SHA-256 del bloque anterior.
  - El encabezado de un bloque además contiene el Merkle Root:
    - Las transacciones en un bloque se organizan en pares y se aplica SHA-256 a cada par para crear un hash.
    - Este proceso se repite, combinando y volviendo a hashear los resultados, hasta que queda un único hash llamado el Merkle Root.

- **Minería de Bitcoin:** Así se usa SHA-256 en la minería de Bitcoin:
  1. Los mineros agrupan las transacciones pendientes en un **bloque candidato**.
  2. Se crea el **encabezado del bloque**.
     - El mismo contiene hash del bloque anterior, raíz de Merkle, marca de tiempo actual, nonce.
     - El **nonce** es un número de 32 bits que se modifica repetidamente durante el proceso de minería.
     - El nonce comienza generalmente desde 0
  3. Se aplica el **algoritmo SHA-256** dos veces al encabezado del bloque.
  4. Si el hash obtenido no cumple con los requisitos de dificultad establecidos por la red,
     - el nonce se incrementa aleatoriamente y se repite el proceso hasta encontrar un hash válido.

Bitcoin

- **Minería de Bitcoin – cont:**
  - Para ser **válido** el hash encontrado durante la minería debe cumplir:
    - El doble hash SHA-256 del bloque completo debe tener un número específico de ceros iniciales.
  - La cantidad de ceros se ajusta periódicamente (aproximadamente cada 2016 bloques – corresponde a dos semanas)
    - para mantener el tiempo de generación de bloques cercano a 10 minutos.
    - Cuantos más ceros se requieren, mayor será la dificultad y más trabajo computacional será necesario para encontrar un hash válido.
  - Una vez encontrado un hash válido, el minero **difunde el nuevo bloque** a la red.
  - Los nodos verifican el trabajo realizado y **añaden** el bloque a la blockchain.
  - Los otros nodos mineros que estaban tratando de crear sus bloques válidos pierden el trabajo realizado hasta ese momento.
  - El minero recibe una **recompensa** en bitcoins por su trabajo, así como las **tarifas de las transacciones** incluidas en el bloque.

Bitcoin

- **Minería de Bitcoin – cont:**
  - **Ejemplo**: Si en un periodo de dos semanas (2016 bloques) los bloques se encuentran en promedio cada 8 minutos en lugar de 10,
    - la red aumentará la dificultad para que los bloques tarden más en encontrarse y se acerquen nuevamente a los 10 minutos por bloque.
  - Notar que los nodos mineros que no ganaron la contienda abortan sus trabajos y tienen que comenzar de nuevo con el siguiente bloque.
    - Van a comenzar todos mas o menos al mismo tiempo.
- **Problema**: Si dos nodos mineros generan un bloque válido al mismo tiempo,
  - ¿Cómo evitar que se generen blockchains diferenrtes?

- **Solución**:
  - Diferentes nodos reciben uno u otro bloque y lo añaden a su copia local de la blockchain.
  - El siguiente minero en encontrar un bloque válido lo añadirá a la cadena que está usando.
  - La cadena que se extiende más rápido se convierte en la principal.
  - El bloque que no fue extendido se convierte en un bloque huerfano y es descartado.
  - Los nodos que tenían la cadena mas corta abandonan esa cadena y se pasan a la cadena mas larga.
  - El minero cuyo bloque se convierte en huérfano no recibe la recompensa por su trabajo.

Bitcoin

- **Problema**: Hay que garantizar que se cumple lo siguiente:
  - **Autenticidad**: Hay que verificar que una transacción ha sido enviada por la persona que afirma haberla enviado.
  - **Integridad**: Hay que asegurar que las transacciones no han sido alteradas desde que fueron creadas.
  - **No repudio**: imposibilitar a los creadores de las transacciones puedan negar haberlas creado.
  - **Seguridad**: proveer protección contra falsificaciones y otros tipos de ataque.
  - **Verificación descentralizada**: cualquier participante de la red debe poder verificar la validez de las transacciones.

- **Solución**: Uso de **firmas digitales**.
  - Una firma digital se genera en función de los datos de una transacción.
    - La firma es única para cada transacción.
  - La clave privada del usuario es usada para firmar transacciones y demostrar la propiedad de los bitcoins.
  - La clave pública se usa para verificar la firma.
  - El remitente crea un **hash** de la transacción y lo **cifra** con su clave privada, produciendo la firma digital.
  - **Verificación de la firma:** Los nodos de la red utilizan la clave pública del remitente para **descifrar** la firma y comparar el hash resultante con el de la transacción.
    - Si coinciden, la firma es válida.

Bitcoin

- **Cumplimiento de requisitos por las firmas digitales:**
  - **Autenticidad:** Solo el propietario de una clave privada correspondiente a una clave pública puede generar una firma válida.
  - **Integridad:** si los datos de la transacción se modifican en cualquier forma luego de la firma, la firma se vuelve inválida.
    - Esto asegura que los datos no han sido alterados desde la creación de la firma.
  - **No repudio**: Una vez que el firmante ha creado la firma no puede negar haberlo hecho, ya que solo su clave privada podría haber creado esa firma específica.
  - **Seguridad**: Se usa un proceso llamado **ECDSA** que usa matemáticas avanzadas de curvas elípticas para crear la firma digital,
    - haciendo extremadamente difícil que alguien falsifique una firma sin conocer la clave privada.
  - **Verificación descentralizada**: Cualquier nodo de la red puede usar la clave pública del firmante para verificar la firma de la transacción.
- **Nota**: solo di las ideas gruesas principales de la solución de firmas digitales porque se puede hilar mucho más fino en esto y no es el propósito de la materia.

Bitcoin

- **Ejemplo de proceso de firma digital en Bitcoin:**
  1. Alice quiere enviar 1 Bitcoin a Bob. Alice crea una transacción
  2. Alice usa su clave privada para generar una firma digital de la transacción.
  3. Alice transmite la transacción firmada a la red Bitcoin.
  4. Los nodos de la red usan la clave pública de Alice para verificar la firma digital y asegurarse que la transacción no ha sido alterada y que Alice es la legítima propietaria de los fondos.

- **Tipos de nodos:**
  - **Nodos completos**: validan y transmiten transacciones y bloques. Mantienen copia completa de la blockchain.
  - **Nodos mineros**: crean nuevos bloques a través de la minería.
  - **Nodos ligeros**: verifican transacciones usando información resumida. Obtienen información necesaria de nodos completos. Consultan saldos, envían y reciben BTC (criptomoneda de Bitcoin)
  - **Supernodos**: actúan como hubs de alta capacidad, conectándose a muchos otros nodos y facilitando la distribución de datos.

Bitcoin

- **Escenario 1: envío y validación de una transacción**
  1. **Usuario** envía transacción a nodo completo
  2. **Nodo completo** verifica la validez de la transacción recibida (p.ej. que el remitente tiene suficientes fondos).
     - Luego difunde la transacción a otros nodos completos y supernodos.
  3. **Supernodo** retransmite transacción recibida de nodo completo a muchos otros nodos completos y ligeros a los que está conectado.
  4. **Nodo ligero** recibe transacción de supernodo, la verifica usando información resumida, si la verificación es exitosa, la acepta.
  5. **Nodo minero**: recibe la transacción del nodo completo o supernodo. Incluye la transacción en un bloque candidato y empieza a minar.
  6. **Nodo complete y supernodo**: continúan propagando la transacción a otros nodos en la red hasta que todos la hayan recibido.
- **Nota**: nodos mineros y completos transmiten una transacción a sus nodos vecinos.

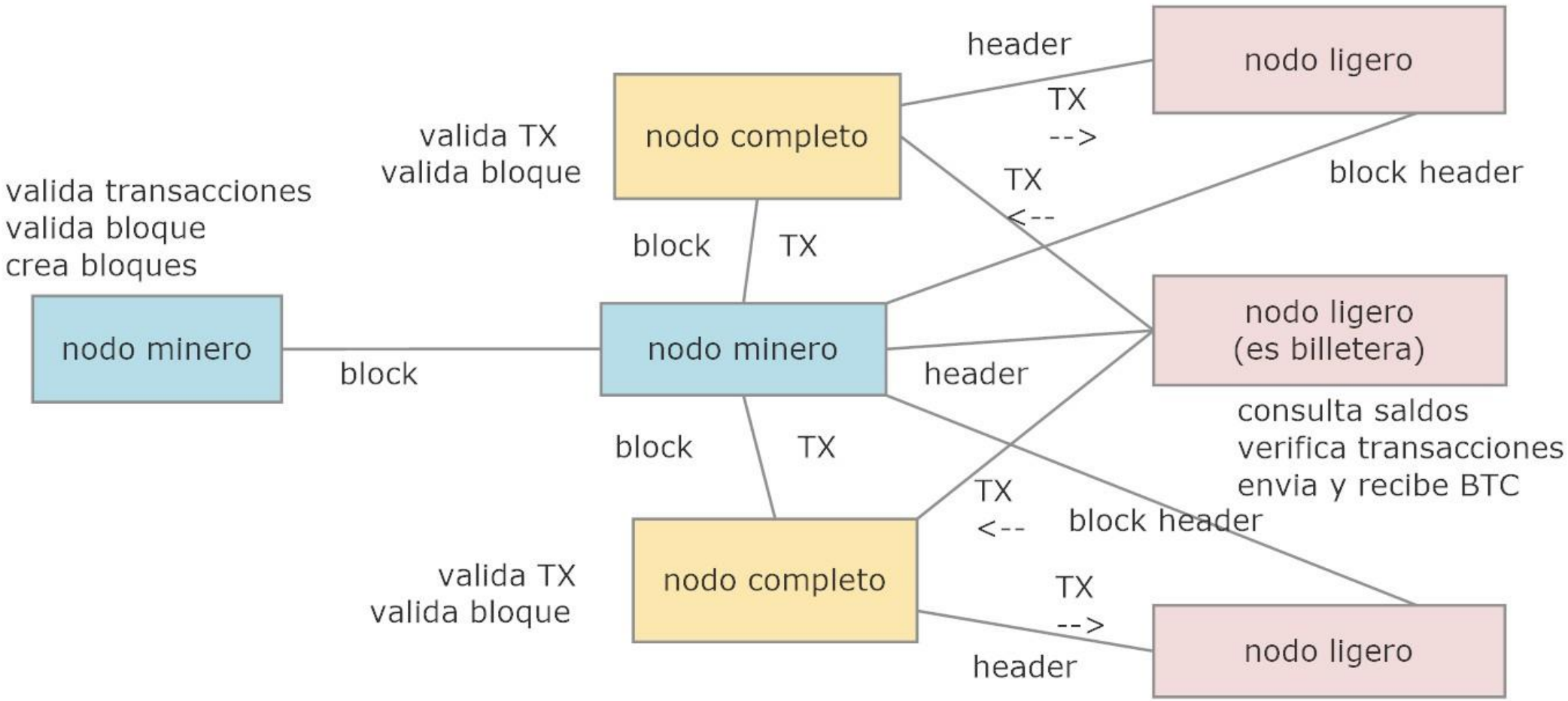
Bitcoin

- **Escenario 2: Minería y propagación de un nuevo bloque.**
  1. **Nodo de minería**: Resuelve problema criptográfico y crea un nuevo bloque con las transacciones recientes.
     - Difunde el nuevo bloque a un nodo completo.
  2. **Nodo completo**: verifica la validez del bloque recibido (o sea, que el bloque sigue las reglas de consenso y no contiene transacciones inválidas.)
     - Si el bloque es válido, lo agrega a su blockchain.
     - Difunde el bloque a otros nodos completes y supernodos.
  3. **Supernodo**: recibe y verifica bloque que recibió del nodo completo, y si todo está bien lo añade a su copia de la blockchain.
     - Retransmite el bloque a muchos otros nodos completos y ligeros a los que está conectado.
  4. **Nodo ligero**: recibe bloque del supernodo y verifica el bloque usando la cabecera del mismo y la cadena de bloques resumida.
     - Si es válido, almacena su información resumida.

Bitcoin

- **Escenario 3**: **respuesta a un ataque de doble gasto**
  1. **Usuario malicioso**: envía dos transacciones conflictivas (doble gasto) a diferentes nodos completos.
  2. **Nodo completo 1**: recibe la primera transacción y la verifica. Difunde esa transacción a otros nodos completos y supernodos.
  3. **Nodo completo 2**: recibe la segunda transacción conflictiva y la verifica. Detecta el conflicto con la primera transacción; rechaza la segunda transacción y no la difunde.
  3. **Supernodo**: recibe la primera transacción válida del nodo completo 1 y le difunde a otros nodos.
  4. **Nodo de minería**: recibe la primera transacción válida y la incluye en un bloque candidato. Si la segunda transacción llega después de la primera y detecta el conflicto y la rechaza.
- **Notas**: asume que usuario malicioso demora bastante entre la primera transacción y la segunda. (Como para que la primera se propague antes.)
  - Si esto no fuera cierto, podría pasar (con muy baja probabilidad) que dos mineros generan bloques en simultaneo con las dos transacciones diferentes.
  - En este caso el escenario se complica; pero hay solución y ya la explicamos antes.

Bitcoin


header
nodo ligero
TX
-->
valida TX
valida bloque
nodo completo
TX
<--
block header
valida transacciones
valida bloque
crea bloques
block
TX
nodo minero
nodo minero
block
header
nodo ligero
(es billetera)
consulta saldos
verifica transacciones
envia y recibe BTC
block
TX
TX
<--
block header
valida TX
valida bloque
nodo completo
TX
-->
header
nodo ligero

Ethereum

- **Propósito de Ethereum:**
  - La red Ethereum fue diseñada para **descentralizar la web** y permitir la creación de **aplicaciones descentralizadas** (DApps) y **contratos inteligentes**.
  - Su objetivo principal es eliminar intermediarios y proporcionar una plataforma transparente, segura y autónoma para transacciones y acuerdos digitales.
  - Ethereum busca ser la base de la **Web 3.0,** ofreciendo una infraestructura para una internet más abierta y descentralizada.

- **Características salientes:**
  1. **Contratos inteligentes**
     - Códigos autoejecutables que se activan cuando se cumplen condiciones predefinidas.
     - Eliminan la necesidad de confianza entre partes y reducen costos de intermediación
  2. **Descentralización**
     - Opera mediante una red global de nodos que validan y registran transacciones sin una autoridad central
  3. **Token nativo (Ether - ETH)**
     - Utilizado como "combustible" para pagar transacciones y ejecutar contratos inteligentes
  4. **Máquina Virtual Ethereum (EVM)**
     - Proporciona un entorno seguro para ejecutar contratos inteligentes.
  5. **Inmutabilidad y transparencia**
     - Todas las transacciones y contratos son registrados en la blockchain, lo que los hace verificables y resistentes a alteraciones

Ethereum

- **Servicios provistos:**
  1. **Desarrollo de DApps**
     - Permite a los desarrolladores crear aplicaciones descentralizadas en áreas como finanzas, juegos y gestión de datos
  2. **Tokens personalizados**
     - Facilita la creación de tokens ERC-20 y ERC-721 para representar activos digitales o físicos
  3. **Finanzas descentralizadas (DeFi)**
     - Plataforma para servicios financieros como préstamos, intercambios y seguros sin intermediarios
  4. **Mercados descentralizados**
     - Permite la creación de plataformas de comercio peer-to-peer para bienes digitales y físicos
  5. **Gestión de identidad y datos**
     - Ofrece soluciones para almacenar y compartir datos de forma segura y descentralizada

- **El staking en Ethereum** es el proceso mediante el cual los usuarios bloquean una cantidad específica de Ether (ETH) para participar en la validación de transacciones y la creación de nuevos bloques en la red.
  - Este mecanismo forma parte del modelo de consenso de **Prueba de Participación** (**Proof of Stake, PoS**), que Ethereum 2.0 adoptó como alternativa a la Prueba de Trabajo

Ethereum

- En Ethereum hay una **máquina virtual de Ethereum** (**EVM**) cuyo **estado** es aceptado por todos los participantes de la red de Ethereum.
  - Cada nodo de la red de Ethereum mantiene una **copia del estado** de la EVM.
- Cualquier participante puede enviar una **solicitud** para que esta EVM realice un **cálculo arbitrario**.
  - Cuando se transmite una solicitud de este tipo, otros participantes de la red verifican, validan y llevan a cabo (ejecutan) el cálculo.
  - Esta ejecución provoca un cambio de estado en la EVM, que se registra y propaga por toda la red.
  - Las solicitudes de cálculo se llaman **solicitudes de transacción**.
- El registro de todas las transacciones y el estado actual de la EVM se almacenan en la blockchain.
- Todas las transacciones son **firmadas** y ejecutadas con los **permisos adecuados**.
  - Nadie debería poder enviar activos digitales desde la cuenta de una persona P, excepto la propia persona P.
- Cualquier participante que envíe una solicitud de transacción debe ofrecer una cantidad de ETH a la red como **recompensa**.
  - La red **quemará una parte** de esta recompensa y otorgará el resto a quien finalmente realice el trabajo de verificar la transacción, ejecutarla, registrarla en la blockchain y transmitirla a la red.
  - La cantidad de ETH pagada corresponde a los recursos necesarios para hacer el cálculo.
    - Así, nadie puede bloquear la red con cálculos infinitos.

Ethereum

- Los desarrolladores de aplicaciones cargan programas en el estado de la EVM, y los usuarios realizan solicitudes para ejecutar estos fragmentos de código con parámetros variados.
  - A estos programas cargados en la red y ejecutados por ella se los llama **contratos inteligentes**.
  - Un contrato inteligente es como un script que cuando se usa con ciertos parámetros, realiza acciones o cálculos si se cumplen determinadas **condiciones**.
  - **Ejemplo**: un contrato inteligente de un vendedor puede crear y asignar la propiedad de un active digital si quien lo invoca envía ETH a un destinatario específico.
- Cualquier desarrollador puede crear un contrato inteligente y **hacerlo público en la red**, utilizando la blockchain como capa de datos, a cambio de una **tarifa pagada** a la red.
- Cualquier usuario puede luego **invocar el contrato inteligente** para ejecutar su código, también **pagando una tarifa** a la red.
- Así, mediante los contratos inteligentes, los desarrolladores **pueden crear y desplegar aplicaciones y servicios** complejos orientados a los usuarios, como mercados, instrumentos financieros, juegos, etc.
- Una vez que un contrato inteligente es desplegado en la red, no se lo puede cambiar.
  - Ser muy cuidadoso en al diseñar y testear un Contrato inteligente.

Ethereum

- Una **aplicación descentralizada (dApp)** combina un contrato inteligente con un front end conteniendo una interfaz de usuario.
- Una dApp tiene su **código de backend** funcionando en la red descentralizada de Ethereum que es de igual a igual.
  - Ninguna persona tiene el control de Etherum.
  - Las dApp se ejecutan en la EVM.
- Una dApp puede tener **código frontend** e **interfaces de usuario** escritas en cualquier lenguaje (igual que una app) para realizar **llamadas a su backend**.
- Si el contrato tiene un error, no afectará el funcionamiento normal de la red de Ethereum.

Cuentas

- Una **cuenta de Ethereum** es una entidad con un **saldo de ether** (ETH) que puede **enviar transacciones** en Ethereum.
- Las cuentas pueden ser controladas por usuarios o implementadas como contratos inteligentes.
- Ethereum tiene dos **tipos de cuentas**:
  - **Cuenta de propiedad externa (EOA)**: controlada por cualquier persona con las claves privadas.
  - **Cuenta de contrato**: un contrato inteligente implementado en la red, controlado por código.
- Ambos tipos de cuentas tienen la capacidad de:
  - Recibir, mantener y enviar ETH y tokens.
  - Interactuar con contratos inteligentes implementados.

Ethereum

Cuentas – cont:

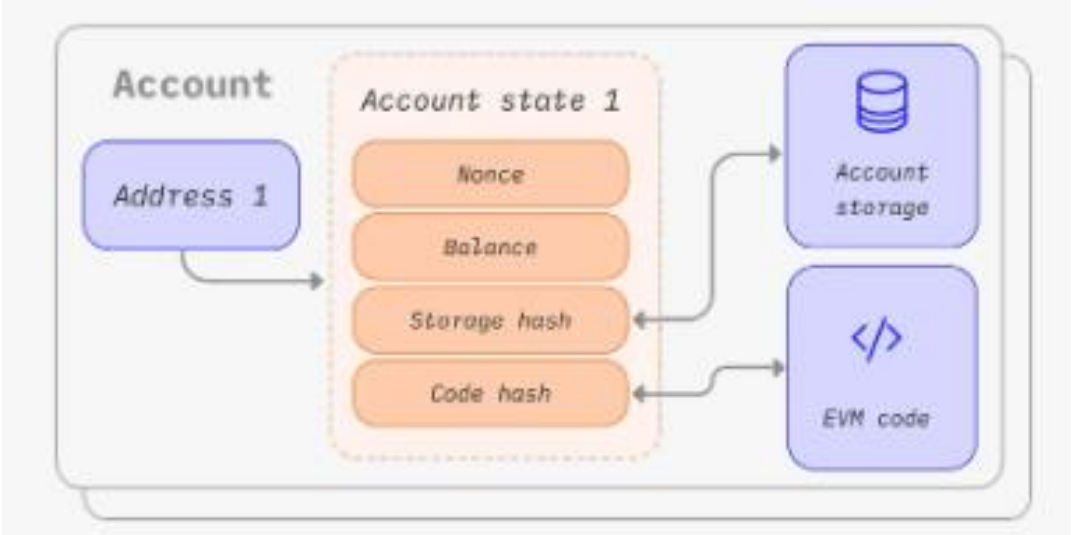
- Para una **cuenta de propiedad externa**:
  - Crear una cuenta no tiene costo.
  - Puede **iniciar transacciones**.
  - Las transacciones entre cuentas de propiedad externa solo pueden ser **transferencias de ETH o tokens**.
  - Se compone de un par criptográfico de claves: **claves públicas** y **privadas** que controlan las actividades de la cuenta.

- Para una **cuenta de contrato**:
  - Crear un contrato **tiene un costo**, ya que utiliza **almacenamiento en la red**.
  - Solo puede enviar transacciones en respuesta a recibir una transacción.
  - Las transacciones desde una cuenta externa a una cuenta de contrato pueden activar código que puede ejecutar muchas acciones diferentes, como transferir tokens o incluso crear un nuevo contrato.
  - Las cuentas de contrato **no tienen claves privadas.** En cambio, son controladas por la lógica del código del contrato inteligente.

Ethereum

Cuentas – cont:

- Una cuenta tiene un **Nonce**; el mismo es un contador y significa:
  - Para cuenta de propiedad externa: cantidad de transacciones enviadas desde ella.
  - **Para una cuenta de contrato**: cantidad de contratos creados por ella.
- Una cuenta tiene un **balance**, que es la cantidad de wei que tiene; hay 1e+18 wei por cada ETH.
- Las cuentas de contrato tienen **fragmentos de código** programados que pueden realizar diferentes operaciones.
  - Un fragmento de código se ejecuta si la cuenta de contrato recibe una **llamada de mensaje.**
  - Los fragmentos de código están contenidos en una **base de datos de estado** bajo sus hashes correspondientes (llamados **codehash**).
- En una cuenta de propiedad externa el campo codehash es el hash de una cadena vacía.
- Además hay un hash asociado al almacenamiento de una cuenta (llamado **storehash**).
- Solo una transacción con un nonce específico puede ejecutarse para cada cuenta, lo que protege contra ataques de repetición, donde transacciones firmadas se transmiten y ejecutan repetidamente.



Ethereum

Cuentas – cont:

- Una cuenta está compuesta por un par de **claves criptográficas**: pública y privada.
  - Estas claves ayudan a demostrar que una transacción fue **firmada** realmente por el remitente y previenen falsificaciones.
- Tu clave privada es la que utilizas para firmar transacciones, lo que te otorga la **custodia de los fondos** asociados con tu cuenta.
  - En realidad, no posees criptomonedas directamente, posees claves privadas; **los fondos siempre están en el registro de Ethereum**.
- Esto impide que actores malintencionados transmitan transacciones falsas, ya que siempre puedes verificar el remitente de una transacción.
- Si Alice quiere enviar ether desde su propia cuenta a la cuenta de Bob, Alice necesita crear una solicitud de transacción y enviarla a la red para su verificación.
- El uso de la criptografía de clave pública en Ethereum asegura que Alice pueda demostrar que ella fue quien inició originalmente la solicitud de transacción.
- Sin mecanismos criptográficos, un adversario malicioso como Eve podría simplemente transmitir públicamente una solicitud que diga algo como "enviar 5 ETH desde la cuenta de Alice a la cuenta de Eve", y nadie podría verificar que esta solicitud no proviene de Alice.

Ethereum

Cuentas – cont:

- Cuando deseas crear una cuenta, la mayoría de las bibliotecas generarán para ti una **clave privada aleatoria**.
  - Una clave privada está compuesta por **64 caracteres hexadecimales** y puede ser encriptada con una contraseña.
  - **Ejemplo**:ffffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd036415f
- La **clave pública** se genera a partir de la clave privada utilizando el algoritmo Elliptic Curve Digital Signature Algorithm.
- Obtienes una **dirección pública para tu cuenta** tomando los últimos 20 bytes del hash Keccak-256 de la clave pública y agregando "0x" al comienzo.
  - Una cuenta de propiedad externa tiene una dirección de 42 caracteres - segmento de 20 bytes, que son 40 caracteres hexadecimales más el prefijo "0x".
  - Ejemplo:0x5e97870f263700f46aa00d967821199b9bc5a120
- Es vital mantener las claves privadas seguraqs y como el nombre lo sugiere, privadas.

Ethereum

Cuentas – cont:

- La clave privada se usa para **firmar mensajes y transacciones**, lo cual produce una **firma**.
- Otros pueden tomar la firma para **derivar tu clave pública**, proveyendo el autor del mensaje.
- Las cuentas de contrato también tienen una **dirección** de 42 caracteres hexadecimales.
- **Ejemplo**:`0x06012c8cf97bead5deae237070f9587f8e7a266d`
- La **dirección del contrato** se proporciona cuando un contrato se implementa en la blockchain de Ethereum.
- La dirección proviene de la dirección del creador y del número de transacciones enviadas desde esa dirección (el "nonce").
- En Ethereum existe otro tipo de claves llamadas claves 'BLS',
  - que se utilizan para identificar validadores.
  - Estas claves pueden agregarse de manera eficiente para reducir el ancho de banda necesario para que la red alcance el consenso.

Ethereum

Cuentas – cont:

- Una cuenta no es una billetera.
- Una **billetera** es una interfaz o aplicación que te permite interactuar con tu cuenta de Ethereum, ya sea una cuenta de propiedad externa o una cuenta de contrato.

Transacciones:

- Las **transacciones** son instrucciones firmadas criptográficamente provenientes de cuentas.
- Una cuenta iniciará una transacción para **actualizar el estado** de la red Ethereum.
- La transacción más sencilla es transferir ETH de una cuenta a otra.
- Una transacción de Ethereum se refiere a una acción iniciada por una cuenta de propiedad externa.
- Las transacciones que modifican el estado de la EVM necesitan ser **transmitidas a toda la red**.
- Cualquier nodo puede enviar una solicitud para que se ejecute una transacción en la EVM;
  - una vez hecho esto, un **validador** ejecutará la transacción y propagará el cambio de estado resultante al resto de la red.

Ethereum

Transacciones – cont:

- Una transacción enviada por una cuenta de propiedad externa incluye la siguiente información:
  - **from**: la dirección del remitente, quien firmará la transacción.
  - **to**: la dirección del destinatario (si es una cuenta de propiedad externa, la transacción transferirá valor; si es una cuenta de contrato, la transacción ejecutará el código del contrato).
  - **signature**: el identificador del remitente. Este se genera cuando la clave privada del remitente firma la transacción y confirma que este ha autorizado dicha transacción.
  - **nonce**: un contador que incrementa secuencialmente y que indica el número de transacciones enviadas desde la cuenta.
  - **value**: cantidad de ETH a transferir del remitente al destinatario (denominado en wei).
  - **input data**: campo opcional para incluir datos arbitrarios..
  - **gasLimit**: la cantidad máxima de unidades de gas que puede consumir la transacción. La EVM especifica las unidades de gas requeridas para cada paso computacional.
  - **maxPriorityFeePerGas**: el precio máximo del gas consumido que será incluido como propina al validador.

Ethereum

Transacciones – cont:

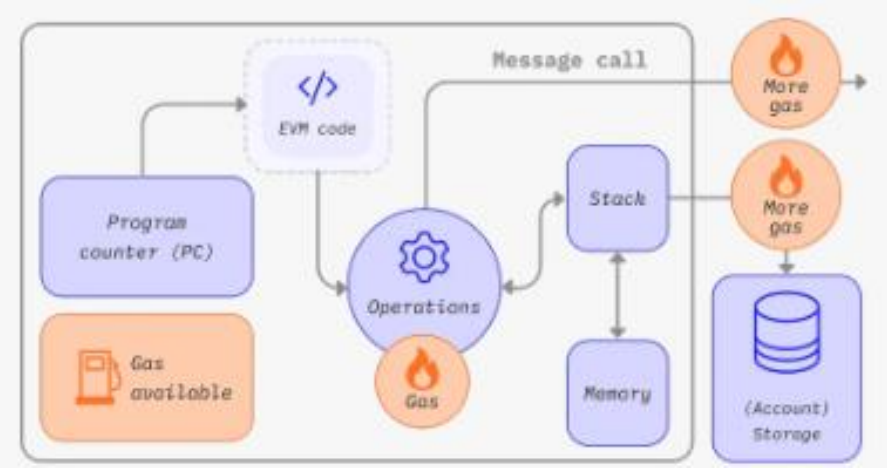
- **Tipos de transacciones:**
  - **Transacciones regulares**: una transacción de una cuenta a otra.
  - **Transacciones de implementación de contratos**: una transacción sin una dirección "to", donde el campo de datos se utiliza para el código del contrato.
  - **Ejecución de un contrato**: una transacción que interactúa con un contrato inteligente ya implementado. En este caso, la dirección "to" es la dirección del contrato inteligente.

- **Ciclo de vida de una transacción**: Una vez que la transacción ha sido enviada, ocurre lo siguiente:
  1. Se genera criptográficamente un hash de la transacción.
  2. La transacción se **transmite a la red** y se agrega a un grupo de transacciones que incluye todas las demás transacciones pendientes de la red.
  3. Un validador debe elegir tu transacción e incluirla en un bloque para verificarla y considerarla "exitosa".
  4. A medida que pasa el tiempo, el bloque que contiene tu transacción será actualizado a "justificado" y luego a "finalizado

Ethereum

Máquina virtual de Ethereum (EVM):

- La **Máquina Virtual de Ethereum** (EVM) es un **entorno virtual descentralizado** que ejecuta código de manera consistente y segura en todos los nodos de Ethereum.
- Los nodos ejecutan la EVM para realizar contratos inteligentes, utilizando "gas" para medir el esfuerzo computacional requerido para las operaciones, lo que asegura una asignación eficiente de recursos y la seguridad de la red.



- El **estado de Ethereum** es una gran estructura de datos que contiene no solo todas las cuentas y balances, sino también un **estado de máquina**,
  - el cual puede cambiar de un bloque a otro según un conjunto de reglas predefinidas y puede ejecutar código de máquina arbitrario.
- Las reglas específicas para cambiar el estado de bloque a bloque son definidas por la EVM.
- En el contexto de Ethereum, el estado es una enorme estructura de datos llamada **Merkle Patricia Trie modificada**, que mantiene todas las cuentas enlazadas mediante hashes y reducibles a un único hash raíz almacenado en la blockchain.

Ethereum

Máquina virtual de Ethereum (EVM) - cont:

- La EVM se ejecuta como una **máquina de pila** con una profundidad de 1024 elementos.
  - Cada elemento es una palabra de 256 bits, lo que se eligió por su facilidad de uso con criptografía de 256 bits.
- Durante la ejecución, la EVM mantiene una **memoria transitoria** (como un arreglo de bytes direccionado por palabras), que no persiste entre transacciones.
- El **código de bytecode de contratos inteligentes compilados** se ejecuta como una serie de opcodes de la EVM,
  - que realizan operaciones estándar de pila como XOR, AND, ADD, SUB, etc.
  - La EVM también implementa una serie de operaciones de pila específicas de blockchain, como ADDRESS, BALANCE, BLOCKHASH, etc.

Ethereum

- **Prueba de participación:** es una forma de demostrar que los validadores han puesto algo de valor en la red que puede ser destruido si actúan deshonestamente.
  - Los validadores **apuestan capital** en forma de ETH mediante un contrato inteligente en Ethereum.
  - El validador verifica que los nuevos bloques propagados por la red son válidos y ocasionalmente crean y propagan bloques por si mismos.
  - Para **participar como validador** un usuario deposita 32 ETH en el **contrato de depósito** y ejecuta tres piezas de software por separado:
    - Un cliente de ejecución, un cliente de consenso, un cliente de validación.
  - Al depositar sus ETH, el usuario entra en **cola de activación** que limita la tasa de nuevos validadores que se unen a la red.
  - Una vez activados, los validadores reciben nuevos bloques de los pares en la red.
  - Se verifica el bloque y el validador **envía un voto (llamado atestación)** a favor de ese bloque a través de la red.

Ethereum

Nodos:

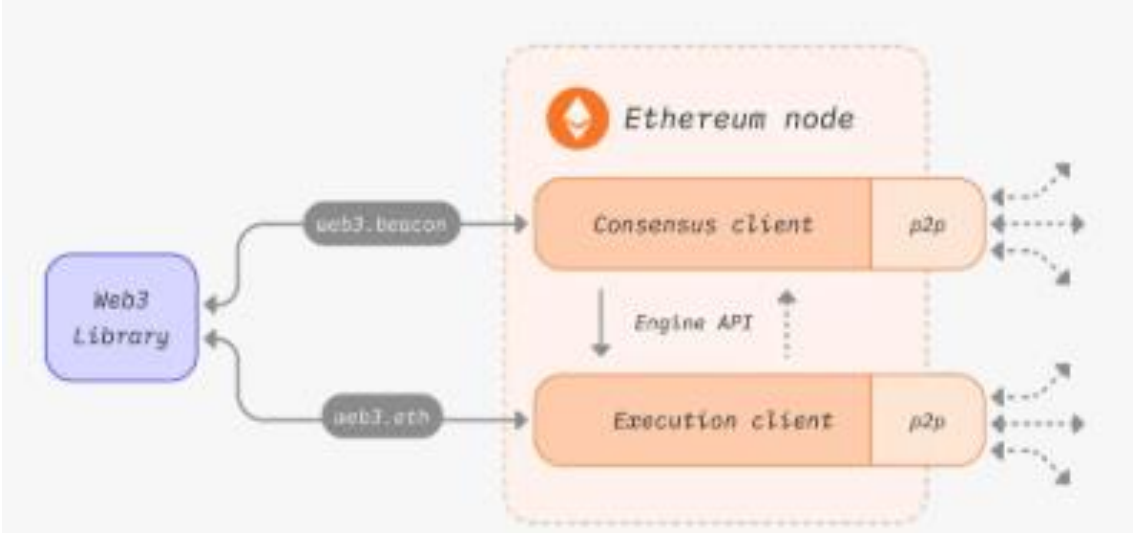
- Un "**nodo**" es cualquier instancia del software cliente de Ethereum que está conectado a otras computadoras que también ejecutan el software de Ethereum, formando una red.
- Un **cliente** es una implementación de Ethereum que verifica los datos según las reglas del protocolo y mantiene la seguridad de la red.
- Un nodo debe ejecutar dos clientes: un cliente de consenso y un cliente de ejecución.

- **Cliente de ejecución** (también conocido como Motor de Ejecución): escucha las nuevas transacciones transmitidas en la red, las ejecuta en la EVM y mantiene el estado más reciente y la base de datos de todos los datos actuales de Ethereum.
- **Cliente de consenso** (también conocido como **Nodo Beacon**): implementa el algoritmo de consenso basado en prueba de participación, que permite a la red alcanzar un acuerdo basado en los datos validados por el cliente de ejecución.

Ethereum

Nodos – cont:

- También existe una tercera pieza de software, conocida como "**validador**", que se puede agregar al cliente de consenso, permitiendo que un nodo participe en la seguridad de la red.
- Estos clientes trabajan juntos para realizar un seguimiento de la cabeza de la cadena de Ethereum y permitir que los usuarios interactúen con la red Ethereum.



Ethereum

- **Tipos de nodos:**
  - **Nodos completos**.
    - Participan en validación de bloques y estados.
    - Mantienen una copia actualizada de la blockchain.
    - Proveen acceso a datos de la blockchain a nodos ligeros.
    - Almacena el estado global.
  - **Validadores**:
    - Los validadores crean bloques.
    - Un validador es elegido aleatoriamente en cada ranura para que proponga un bloque.
    - Distribuyen bloques creados a otros nodos de la red.
    - Además actualizan el estado global.
    - Son recompensados con ETH por producir bloques.
  - **Nodos ligeros**: proporcionan una forma menos demandante de interactuar con la red Ethereum, ideal para dispositivos con recursos limitados.
    - No participan en el consenso.
    - No requieren de staking.
    - Descargan solo los encabezados de los bloques.
    - Verifican la validez de bloques
    - Solicitan datos específicos a nodos completos.
  - **Nodos de archivo**: mantienen copia completa del historial de la blockchain, incluyendo estados antiguos y datos detallados.
    - Almacenan datos históricos para consultas y auditorias.
    - Facilitan el acceso a la información antigua por aplicaciones.
    - Además almacenan el estado global actual.

Ethereum

- **Prueba de participación – cont:**
  - El tiempo se divide en **ranuras** de 12 segundos y **epochs** que corresponden a 32 ranuras.
  - En cada ranura, se **selecciona aleatoriamente** a un validador para que sea el proponente de un bloque.
  - Este validador es responsable de crear un nuevo bloque y enviarlo a otros nodos en la red.
  - Además, en cada slot, se elige aleatoriamente a un **comité de validadores** cuyas votaciones se utilizan para determinar la validez del bloque propuesto.
  - Dividir el conjunto de validadores en comités es importante para mantener la carga de la red manejable.
  - Los comités organizan a los validadores de manera que cada validador activo realiza atestaciones **en cada epoch,** pero no en cada slot.
- Suponiendo que todos los validadores están en línea y funcionando, va a haber un bloque en cada ranura, significando que el tiempo de bloque es 12 segundos.

Ethereum

- **Prueba de participación - ejecución de transacciones:**
  - El usuario **crea y firma una transacción** con su clave privada (Esto generalmente lo gestiona una billetera o una biblioteca).
  - El usuario define la **cantidad de gas** que está dispuesto a pagar como **propina** a un validador para incentivarlo a incluir la transacción en un bloque.
    - las propinas se pagan al validador, mientras que la tarifa base se quema.
  - La transacción se envía a un cliente de ejecución de Ethereum que verifica su validez.
    - Esto significa asegurarse de que el remitente tenga suficiente ETH para cumplir con la transacción y que la haya firmado con la clave correcta.
  - Si la transacción es válida, el cliente de ejecución la agrega a su **lista de transacciones pendientes (mempool local)** y también la transmite a otros nodos a través de la red.
  - Cuando otros nodos escuchan sobre la transacción, la agregan a su lista de transacciones pendientes.

Ethereum

- **Prueba de participación - ejecución de transacciones – cont:**
  - Uno de los nodos validador en la red es el **proponente de bloques** para el slot actual, habiendo sido seleccionado previamente de forma pseudoaleatoria mediante el algoritmo RANDAO.
  - Este nodo es responsable de construir y transmitir el siguiente bloque que se añadirá a la cadena de bloques de Ethereum y de actualizar el estado global.
  - El nodo se compone de tres partes: un cliente de ejecución, un cliente de consenso y un cliente de validación.
- El **cliente de ejecución** agrupa las transacciones del mempool local en un "execution payload" y las ejecuta localmente para generar un **cambio de estado.**
- Esta información se pasa al **cliente de consenso**, donde el "execution payload" se envuelve como parte de un **bloque faro**
  - que también contiene información sobre recompensas, penalizaciones, slashing, atestaciones, etc., que permite que la red acuerde la secuencia de bloques en la cabeza de la cadena.

Ethereum

- **Prueba de participación - ejecución de transacciones – cont :**
  - Otros nodos reciben el nuevo bloque faro en la red de difusión de la capa de consenso.
    - Lo pasan a su cliente de ejecución, donde las transacciones se vuelven a ejecutar localmente para garantizar que el cambio de estado propuesto sea válido.
  - El **cliente de validación** luego atesta que el bloque es válido y es el siguiente bloque lógico según su vista de la cadena
    - El bloque se añade a la base de datos local en cada nodo que lo atesta.
  - Los **checkpoints** ocurren al inicio de cada epoch y existen para tener en cuenta el hecho de que solo un subconjunto de validadores activos realiza atestaciones en cada slot, pero todos los validadores activos realizan atestaciones durante cada epoch.
  - El primer bloque de cada epoch es un checkpoint. Los validadores votan por pares de checkpoints que consideran válidos.
  - Si un par de checkpoints recibe votos que representan al menos dos tercios del total de ETH apostados, los checkpoints son actualizados.
  - El más reciente de los dos (objetivo o target) se convierte en "**justificado**".
  - El más antiguo de los dos ya está justificado porque era el "objetivo" en el epoch anterior. Ahora se actualiza a "**finalizado**".

Ethereum

- **Campos en un bloque al nivel más alto:**
  - **Slot**: la ranura a la cual pertenece el bloque.
  - **Proposer_index**: el ID del validador proponiendo el bloque.
  - **Parent_root**: el hash del bloque previo.
  - **State_root**: el hash de la raíz del objeto de estado.
  - **Body**: contiene numerosos campos.
- **Algunos campos del cuerpo (body):**
  - **Randao_reveal**: valor usado para elegir el siguiente nodo que va a proponer bloques.
  - **Attestations**: lista de atestaciones a favor del bloque corriente.
  - **Execution_payload**: transacciones pasadas por el cliente de ejecución.
- Hay muchos más campos, que no menciono.

Ethereum

Estado Global

- El estado global de Ethereum usa una estructura de datos llamada **árbol de Merkle-Patricia.**
  - Es una estructura de **pares clave-valor**, donde las **claves** representan direcciones (de cuentas y contratos), y los **valores** representan datos asociados (saldos, variables de contrato, etc.)
- El campo **raíz de estado (state root)** es un **hash** de la estructura del árbol de Merkle-Patricia.
  - El mismo se almacena en el encabezado del bloque.

- El árbol de Merkle-Patricia se compone de tres **tipos de nodos:**
  - **Nodos hoja**: contienen el valor asociado a una clave específica.
    - Las claves son hashes que identifican cuentas o datos de contratos.
  - **Nodos de ramificación**: Contienen hasta 16 enlaces (uno por cada posible carácter hexadecimal en las claves).
    - Sirven para enrutar claves hacia sus valores asociados.
  - **Nodos de extensión**: optimizan el árbol almacenando secuencias consecutivas de caracteres en una sola ruta, en lugar de usar múltiples nodos.

Ethereum

- La clave como una dirección específica **guía la ruta a través del árbol**,
  - pasando por los nodos de ramificación hasta llegar a las hojas que contienen los valores correspondientes para esa clave.
- Un **nodo hoja** puede contener informaciones como:
  - el saldo de la cuenta, el nonce (contador de transacciones), datos específicos de almacenamiento de un contrato.
- El **cálculo del State Root** en el árbol de Merkle-Patricia sigue una metodología de **hashing jerárquico**, donde los hashes se aplican progresivamente desde las hojas hasta llegar a la raíz del árbol.
  - Para cada hoja se calcula un hash de su contenido (incluyendo la clave y el valor).
  - Los hashes de los nodos hijo se combinan para formar los hashes de los nodos superiores.
  - Este proceso se repite hasta llegar al nodo raíz del arbol, que genera el state root.
- **Actualizar el estado global** requiere recalcular los nodos directamente afectados por las transacciones del nuevo bloque.
- Luego los hashes de los nodos modificados se propagan hacia arriba en el árbol, para obtener el state root.

Ethereum

Calculo eficiente del state root

- En el árbol de Merkle-Patricia, cada **nodo tiene su propio hash** que representa su contenido y la relación con sus hijos.
  - Los hashes de los nodos que no fueron modificados permanecen igual, y pueden ser reutilizados al calcular el nuevo **state root**.
  - Los cambios en los nodos afectados se propagan hacia arriba en el árbol, recalculando los hashes de los nodos superiores hasta llegar a la raíz.
  - Los nodos que no están en la ruta de las modificaciones no necesitan ser recalculados.

- El **estado de un contrato inteligente** contiene:
  - **Variables de estado** definidas en el contrato inteligente.
  - También puede haber **saldo en ETH asociado al contrato**.
- **Actualizaciones del estado de un contrato**:
  - El estado de un contrato cambia cuando se ejecuta una transacción que **interactúa con el contrato**.
  - **Las interacciones pueden incluir:**
    - llamadas a funciones del contrato que modifican las variables de estado.
    - **Transferencias de ETH** al contrato.
    - **Ejecución de operaciones lógicas** definidas en el contrato.

Ethereum

Construcción del estado global:

- Cada nodo completo almacena el **estado global** actualizado en sus discos que incluye: saldos de las cuentas, estado de los contratos inteligentes.
  - Calculan el nuevo estado global procesando las transacciones del bloque y actualizan el estado global de acuerdo.
- Los **nodos validadores** ejecutan todas las transacciones en el bloque propuesto y **actualizan el estado global**. Esto incluye:
  - Actualizar los saldos de las cuentas.
  - Actualizar las variables de los contratos inteligentes.
- Para **consultar el estado global** sin ejecutar un nodo propio,
  - se pueden consultar servicios de terceros que operan como nodos completos, como **Infura** o **Alchemy**.
- Recordar que en la red de Ethereum solo se difunden bloques y no estados globales completos.

Información de estado global en la blockchain

- En Ethereum la blockchain almacena tanto las transacciones como información relacionada al estado global. Interesa guardar solo:
  - Valores nuevos de los saldos modificados y de variables de contratos inteligentes.
  - **Raiz de Merkle del estado**: o sea un hash del estado global actualizado.
- En Ethereum cada bloque contiene un hash conocido como la **raíz del estado** (state root).
  - Este hash es una representación compacta del estado global.
- Cuando un bloque se propaga por la red, los nodos que lo reciben ejecutan las mismas transacciones del bloque y verifican que el estado que resulta coincide con el hash del estado contenido en el bloque.

Ethereum

Aclaraciones sobre los nodos archivo

- Los **nodos de archivo** pueden proporcionar datos como:
  - El estado de una cuenta (saldo, contratos, etc.) en cualquier bloque específico.
  - Los valores de las variables de un contrato inteligente en un bloque pasado.
- Muchos usuarios y desarrolladores dependen de servicios de terceros como **Infura** y **Alchemy** que operan nodos de archivo para acceder a la información histórica de estado.

- Los nodos de archivo almacenan el árbol de Merkle-Patricia, pero **no necesitan duplicar el árbol entero para cada estado histórico.**
  - En lugar de eso, reutilizan las partes del árbol que no han cambiado entre estados.
  - En vez de almacenar el estado completo de cada bloque, los nodos de archivo guardan las **diferencias** **(deltas)** entre los estados.
  - Solo registran los cambios que ocurrieron como resultado de las transacciones en un bloque específico.

Ethereum

- Igual que los nodos completos, los nodos de archivo **procesan cada bloque** recibido, ejecutando las transacciones incluidas en él y calculando el nuevo estado global.
- A diferencia de los nodos completos, los nodos de archivo no descartan los estados anteriores.
- Al calcular el nuevo estado global, guardan también los **cambios incrementales (deltas)** que se producen en el estado para poder reconstruir cualquier estado histórico.

Ethereum

- Este apunte sobre Ethereum fue sacado del sitio oficial de Ethereum y es un resumen del mismo.
- Si quieren más detalles (son numerosos), ir a:

https://ethereum.org/en/developers/docs/