**Capítulo 13**

**A Fusão e Além**

Neste capítulo, cobriremos “A Fusão”, o nome dado à atualização do Ethereum que substitui o Proof-of-Work (PoW) pelo consenso Proof-of-Stake (PoS) na blockchain do Ethereum. Também abordaremos o que o futuro reserva para o Ethereum e como, por meio de muitas inovações técnicas e atualizações, o Ethereum está alcançando seu objetivo final de se tornar um computador mundial escalável.

Abordaremos os seguintes tópicos neste capítulo:

* Introdução
* Ethereum após a Fusão
* A Fusão
* Sharding
* O futuro do Ethereum

**Introdução**

O objetivo do Ethereum é, em última análise, fazer a transição para uma versão escalável, com bom desempenho e segura do Ethereum que servirá como o computador mundial, que é a visão original do Ethereum. O conceito de um computador mundial foi introduzido com o Ethereum, no qual uma rede global e descentralizada de nós interconectados executa contratos ponto a ponto sem a limitação de desligamento, censura ou invasão. Essa visão começou em 2015 com a cadeia PoW do Ethereum e ganhou bastante tração; no entanto, desafios como escalabilidade, privacidade e desempenho dificultaram um pouco a adoção em massa. O chamado Ethereum 2.0 era esperado para resolver essas questões em busca de se tornar um computador mundial, porém, não há mais um Ethereum 2.0; mas a visão permanece e o primeiro marco importante da adoção do PoS em vez do PoW foi alcançado. Existe um roteiro rico e inovador à frente, o qual garantirá maior eficiência e melhorias para realizar o objetivo final de um computador mundial escalável. Esse trabalho está sendo feito em várias atualizações, que cobriremos neste capítulo.

**Ethereum após a Fusão**

Observe que a terminologia Ethereum 1.0 e Ethereum 2.0 não é mais usada. Em vez disso, as terminologias camada de execução e camada de consenso agora são usadas para descrever, respectivamente, Ethereum 1 e Ethereum 2.

As terminologias Eth 1 e Eth 2 não são mais utilizadas, e agora há apenas uma terminologia, ou seja, Ethereum, que é a versão atual e futura do Ethereum. Melhorias e recursos individuais como a Beacon Chain, a Fusão e o sharding agora são chamados de atualizações. A rede Ethereum será atualizada por meio de várias atualizações ao longo de alguns anos para alcançar seu objetivo final de um Ethereum escalável se tornando um Computador Mundial. Podemos ver isso como uma divisão lógica no protocolo onde a execução lida com o processamento das transações dentro de um bloco e mantém o estado global da cadeia criando e executando contratos inteligentes e transações de transferência. O consenso, por outro lado, escolhe quais blocos incluir na cadeia canônica.

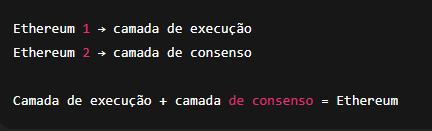
Clientes de execução são novas versões dos clientes Ethereum já em uso, como Geth, Nethermind e Besu. Esses clientes não possuem mais o componente PoW (ETHASH) e agora dependem exclusivamente de clientes de consenso para alcançar consenso.

Clientes de consenso são novos softwares de cliente desenvolvidos especificamente para o Ethereum baseado em PoS. Alguns exemplos incluem Lodestar, Lighthouse, Prysm e Teku. Esses clientes são responsáveis pela validação de blocos e consenso, garantindo que blocos válidos cheguem à cadeia canônica.

Os principais objetivos por trás das atualizações do Ethereum podem ser divididos nas seguintes dimensões:

* **Staking**: O Ethereum, desde seus primeiros dias, visava uma transição para PoS. Isso foi alcançado em setembro de 2022.
* **Sharding**: Um dos principais objetivos do Ethereum é o sharding, que melhorará a escalabilidade, eficiência e desempenho.
* **Eficiência energética**: O Ethereum tornou-se 99,95% eficiente devido à mudança para PoS.

Lembre-se de que agora a visão do Ethereum é a seguinte:



Um nó Ethereum completo após a Fusão é agora uma combinação de um cliente de execução e um cliente de consenso.

* **Segurança aumentada**: Com o advento da criptografia quântica e para mitigar as ameaças enfrentadas pela criptografia em um mundo pós-quântico, prevê-se que, no Ethereum, criptografia resistente a quântica (ou pelo menos a fácil inserção de elementos resistentes a quântica) será introduzida, de forma que a blockchain permaneça segura mesmo em um mundo pós-quântico. Além disso, devido ao mecanismo de proof-of-stake, comprometer a rede seria difícil.
* **Participação aumentada**: Também é esperado que uma maior participação de validadores contribua para uma segurança aprimorada e participação geral.
* **Desempenho aumentado**: Em vez de exigir hardware especializado ou CPUs e GPUs de alto desempenho para executar atividades de validação, espera-se que no Ethereum essas atividades possam ser executadas por computadores comuns de consumidores, com recursos normais. Com a introdução do PoS, espera-se um tempo de bloco mais rápido. Isso também resulta em desempenho aumentado.

A seguinte lista das principais versões do Ethereum mostra como o Ethereum evoluiu ao longo do tempo, juntamente com sua visão de eventualmente alcançar um computador mundial — uma blockchain escalável, descentralizada e segura:

* Julho de 2015: Frontier
* Março de 2016: Homestead
* Outubro de 2017: Byzantium
* Fevereiro de 2019: Constantinople
* Dezembro de 2019: Istanbul
* Final de 2020: Berlin
* Dezembro de 2020: Beacon Chain
* Agosto de 2021: Atualização London
* Setembro de 2022: A Fusão
* 2023 e além: Sharding
* Abril de 2023: Atualização Shanghai / atualização Shapella

Como indicado acima, o desenvolvimento do Ethereum está dividido em atualizações. As atualizações mais recentes relacionadas à nova visão do Ethereum são:

* **Beacon Chain**: Introduziu o staking no Ethereum e fornece uma fundação para futuras atualizações.
* **A Fusão**: A mudança do PoW para o PoS.
* **Sharding**: Consiste em múltiplas atualizações para melhorar a escalabilidade e a capacidade do Ethereum.

Vamos primeiro discutir a Beacon Chain com mais detalhes.

**A Beacon Chain**

A introdução da Beacon Chain marcou a integração do mecanismo PoS no ecossistema Ethereum. A Beacon Chain introduziu uma lógica de consenso e um protocolo de propagação de blocos, que serve para proteger a rede Ethereum.

A Beacon Chain é a cadeia central do sistema e gerencia o mecanismo de consenso PoS Gasper (explorado mais adiante no capítulo). Também é usada para armazenar e manter o registro de validadores. Validadores são nós que participam do mecanismo de consenso PoS. A Beacon Chain possui uma série de recursos, listados a seguir:

* **Produção de aleatoriedade de boa qualidade**, que será usada para selecionar propositores de blocos e comitês de atestação sem viés. Atualmente, um esquema baseado em RANDAO é proeminente; no entanto, outras opções como esquemas baseados em BLS e STARK foram consideradas no passado. RANDAO é um processo pseudoaleatório que seleciona propositores para cada slot em cada época e reorganiza os validadores nos comitês.
* **Fornecimento de atestação**, que essencialmente significa votos de disponibilidade para um bloco em uma cadeia de shard. Um número adequado de atestações para um bloco de shard resultará na criação de um *crosslink*, que fornece confirmação para blocos de shard na Beacon Chain.
* **Gerenciamento de validadores e de staking**.
* **Provisão de conjuntos de validadores (comitês)** para votar em blocos propostos.
* **Aplicação do mecanismo de consenso e dos mecanismos de recompensa e penalidade**.
* **Servir como cadeia central do sistema onde os shards podem escrever seus estados, para que transações entre shards possam ser habilitadas**.

Para participar da Beacon Chain, é necessário um cliente (ou nó) da Beacon Chain.

**Nós da Beacon**

O nó beacon é o elo principal na Beacon Chain que forma o núcleo central da blockchain do Ethereum. As responsabilidades do nó beacon incluem a sincronização da Beacon Chain com outros pares. Ele realiza a atestação de blocos e executa um servidor RPC para fornecer informações críticas aos validadores sobre atribuições e atestação de blocos. Além disso, também lida com a transição de estado e atua como uma fonte de aleatoriedade para o processo de atribuição de validadores.

O nó beacon também atua como um ouvinte para eventos de depósito na cadeia Ethereum, além de criar e gerenciar conjuntos de validadores. Um nó beacon mantém um relógio sincronizado com outros nós na Beacon Chain para garantir a aplicação das regras de penalidade (*slashing*). Em um nó beacon, diversos serviços com base nas responsabilidades mencionadas anteriormente são implementados. Isso inclui o serviço de blockchain, serviço de sincronização, serviço de operações, serviço Ethereum, servidor RPC público e o serviço de rede P2P.

Para participar da Beacon Chain, é necessário um cliente da Beacon Chain, ou cliente de consenso. Com base na especificação do Ethereum, esse cliente é desenvolvido por várias equipes diferentes. Uma lista não exaustiva está disponível aqui:  
<https://ethereum.org/en/developers/docs/nodes-and-clients/#consensus-clients>

Observe que usamos os termos *nó* e *cliente* de forma intercambiável. No entanto, há uma diferença sutil entre os termos: quando dizemos cliente, estamos nos referindo ao software que executa as funções de consenso ou execução, enquanto um nó pode ser visto como uma combinação do software cliente e o hardware (computador) em que ele está sendo executado.  
Geralmente, no entanto, cliente e nó são usados como sinônimos.

**Cliente de Consenso**

A principal responsabilidade do cliente de consenso é manter a sincronização com a rede Ethereum implementando a lógica necessária. Esse processo envolve o recebimento de blocos de outros nós e o uso do algoritmo de escolha de bifurcação (*fork choice*) para garantir que o nó sempre siga a cadeia com mais atestações. Além disso, o cliente de consenso também participa de sua rede P2P para compartilhamento de blocos e atestações.

No entanto, o cliente de consenso não propõe nem atesta blocos. Em vez disso, essa tarefa é reservada aos validadores, um componente opcional adicionado ao cliente de consenso caso o usuário aposte pelo menos 32 ETH.

Sem um validador, o cliente de consenso apenas acompanha a cabeça da Beacon Chain, garantindo que o nó permaneça sincronizado.

O cliente de consenso realiza principalmente a propagação de blocos e atestações sobre sua rede P2P, acompanha a cadeia, executa o algoritmo de escolha de bifurcação e gerencia o estado da Beacon.

**Cliente de Execução**

O cliente de execução atua como ponto de entrada para os usuários do Ethereum na rede da blockchain do Ethereum. O cliente de execução lida com transações, gerencia o estado e contém a Máquina Virtual Ethereum (EVM), o estado e o *pool* de transações. O cliente de execução gera *payloads* de execução, que consistem em transações, tentativas de estado atualizadas e dados semelhantes. Esses *payloads* de execução são incluídos em cada bloco pelos clientes de consenso. Além disso, o cliente de execução, usando a EVM, reexecuta transações dos blocos que recebe para garantir a validade das transações. Ademais, o cliente de execução oferece aos usuários uma interface para interagir com o Ethereum por meio de chamadas RPC, permitindo que eles consultem a blockchain do Ethereum, enviem transações e implantem contratos inteligentes. As chamadas RPC são tipicamente gerenciadas por uma biblioteca como *web3.js* ou por meio de uma carteira de criptomoedas. Clientes de execução permitem interação com o Ethereum via API JSON-RPC e propagam transações na rede P2P, executam transações, verificam mudanças de estado, gerenciam tentativas de estado e tentativas de recibos, criam o *payload* de execução e o enviam para o cliente de consenso.

Na próxima seção, apresentamos os nós validadores, que atuam como propositores de blocos e atestadores na rede como parte do mecanismo PoS no Ethereum.

**Cliente Validador**

Um nó validador, também chamado de cliente validador, participa ativamente do mecanismo de consenso para propor blocos e fornecer atestações. Um usuário pode apostar um mínimo de 32 ETH para obter a capacidade de verificar e atestar a validade de blocos. Como os validadores participam do mecanismo de consenso para proteger o protocolo, eles são incentivados financeiramente por seu esforço. Os validadores ganham ether como *stakers* ao criar e validar novos blocos na cadeia.

Se quiser se tornar um *staker*, siga as instruções no link:  
<https://launchpad.ethereum.org/en/>

Para adicionar um validador ao seu cliente de consenso, os operadores de nó devem depositar 32 ETH no contrato de depósito. O cliente validador está incluído com o cliente de consenso e pode ser adicionado ao nó sempre que desejado. O validador é responsável por atestar e propor blocos e permite que o nó receba recompensas, penalidades ou *slashing*. Além disso, executar o software validador torna o nó um candidato a ser escolhido como o propositor de blocos para um slot.

Validadores fazem propostas de blocos, recebem recompensas e penalidades e atestações de blocos.

**Como fazer *staking* no Ethereum**

Existem quatro maneiras principais de fazer *staking* no Ethereum: *staking* solo, *staking* como serviço, *staking* em grupo (*pooled staking*) e *staking* em exchanges centralizadas:

* **Staking solo** é o mais impactante, proporcionando controle completo e recompensas, ao mesmo tempo em que é sem confiança. No entanto, requer conhecimento técnico e um computador dedicado conectado à internet.
* **Staking como serviço** permite que você delegue a parte difícil para um provedor de serviços enquanto ganha recompensas, mas é necessário confiar a operação do nó a um provedor.
* **Staking em grupo** é uma opção simples que permite aos usuários apostar qualquer quantia, mesmo que seja abaixo de 32 ETH, e ganhar recompensas proporcionalmente. Diversas soluções existem, incluindo *staking líquido*, que envolve um token de liquidez ERC-20 representando o ETH apostado. No entanto, *staking* em grupo não é nativo da rede Ethereum e carrega riscos.
* **Exchanges centralizadas** oferecem serviços de *staking*, mas requerem o mais alto nível de confiança e podem criar um alvo centralizado para ataques e um único ponto de falha, tornando a rede mais vulnerável e menos segura.

Um validador precisa realizar várias configurações antes de poder depositar ether como *stake* para se tornar um validador. Uma vez aceito, ele se torna parte do registro de validadores que é armazenado e mantido na Beacon Chain.

As principais funções que um cliente validador desempenha estão listadas a seguir:

* Propostas de novos blocos.
* Fornecimento de atestação para blocos propostos por outros validadores.
* Agregação de atestações.
* Conexão com um nó beacon confiável para ouvir novas atribuições de validadores e reorganizações (*shuffling*).

Um validador recebe um status com base em seu estado atual. Ele pode ter um dos seis status: **depositado**, **pendente**, **ativado**, **punido (slashed)**, **saído** e **retirável**. O status *depositado* significa que um validador fez um depósito válido e está registrado no estado da Beacon Chain; *pendente* significa que o validador está elegível para ativação; *ativado* significa que o validador está ativo; *punido* significa que o validador perdeu parte do seu *stake*; *saído* significa que o validador saiu do conjunto de validadores; e *retirável* indica que o validador pode sacar os fundos.

Instruções sobre como configurar um nó Ethereum, incluindo vários clientes de camada de consenso e de execução, estão disponíveis aqui:  
<https://ethereum.org/en/developers/docs/nodes-and-clients/run-a-node/>

Um estado *retirável* ocorre depois que um validador saiu e aproximadamente 27 horas se passaram. Um validador pode estar em múltiplos estados simultaneamente — por exemplo, um validador pode estar ao mesmo tempo nos estados *ativado* e *punido*.

O comportamento honesto do validador é incentivado como parte do mecanismo PoS, e o comportamento desonesto resulta no que é chamado de *slashing*. O *slashing* resulta na remoção do validador do comitê de validadores (o conjunto ativo de validadores) e na queima de uma parte dos fundos depositados. O *slashing* serve a dois propósitos. Primeiro, torna proibitivamente caro atacar o Ethereum e, segundo, incentiva (ou força) os validadores a realizar ativamente suas funções. Por exemplo, um validador que fica offline quando deveria estar participando ativamente do consenso é penalizado por sua inatividade.

**Três cenários de *slashing***

O *slashing* pode ocorrer em três cenários. O primeiro é o **slashing de propositor**, que acontece quando um validador assina dois blocos diferentes na Beacon Chain no mesmo *epoch*. O segundo caso é quando um validador assina duas atestações conflitantes. Em terceiro lugar, o *slashing* também pode ocorrer quando um validador, ao atestar, assina uma atestação que “envolve” outra atestação. Em outras palavras, isso ocorre quando um validador primeiro atesta uma versão da cadeia e depois atesta outra versão, causando confusão sobre qual cadeia o validador realmente apoia. Quando qualquer um desses três cenários ocorre, o nó infrator é denunciado por um validador denunciante (*whistleblower*). O nó denunciante cria uma mensagem contendo evidência da infração, a envia a um propositor para incluí-la em um bloco e a propaga na rede. Na Fase 0, a recompensa total do *slashing* é dada ao propositor, e o validador denunciante não recebe recompensa. Isso pode mudar no futuro, onde ambos poderão ser recompensados.

**Penalização vs. Slashing**

Lembre-se também de que penalização e *slashing* são conceitos relacionados, mas diferentes. A **penalização** resulta em uma redução no saldo de um validador devido, por exemplo, à inatividade ou estar offline. Por outro lado, o **slashing** resulta em uma saída forçada da Beacon Chain juntamente com o depósito do validador responsável sendo penalizado a cada *epoch*, enquanto aguarda sua vez na fila de saída da cadeia.

Os cálculos de *slashing* e penalidade são baseados em vários fatores e variáveis como o tempo de inatividade do validador e o tipo de infração que acionou o *slashing*. Além disso, penalidades são aplicadas em vários pontos do processo de *slashing*. Por exemplo, uma penalidade mínima de **saldo efetivo do validador punido / 32** é aplicada quando um propositor inclui a mensagem de denúncia do denunciante em um bloco. Após isso, no início de cada *epoch*, uma penalidade calculada como **3 × recompensa base** é aplicada. Outra penalidade é aplicada entre o momento da inclusão da mensagem de denúncia em um bloco e o momento em que o validador punido pode sacar.

**Recompensas de Staking**

Ganhar recompensas por apostar ether no contrato de depósito também depende de vários fatores. Um exemplo simples é que, se alguém aposta 32 ETH com um preço atual de, por exemplo, USD 240 por ETH, com um tempo de atividade do validador de 80%, o rendimento anual será em torno de **8%**. A recompensa base é calculada conforme a fórmula da especificação da Fase 0:

recompensa\_base = saldo\_efetivo \* (fator\_recompensa\_base / (recompensas\_por\_epoch \* sqrt(soma(saldos\_ativos))))²

Aqui, fator\_recompensa\_base tem o valor padrão de 64, recompensas\_por\_epoch é 4, e soma(saldos\_ativos) é o total de ether apostado entre todos os validadores ativos.

A recompensa recebida por um validador é determinada por dois fatores: seu **saldo efetivo** e o **número de validadores** na rede. À medida que o número de validadores aumenta, a recompensa total emitida pela rede também aumenta, mas a recompensa base dada a cada validador diminui proporcionalmente. Esses fatores influenciam significativamente a **APR** (taxa de retorno anual) de um nó de staking. A recompensa total é calculada combinando cinco componentes distintos, cada um com um peso específico que determina o quanto ele contribui para a recompensa geral.

Esses cinco componentes são:

1. **voto de origem (source vote)**
2. **voto de destino (target vote)**
3. **voto de cabeça (head vote)**
4. **recompensa do comitê de sincronização (sync committee reward)**
5. **recompensa de propositor (proposer reward)**

Os votos estão relacionados ao voto pontual do mecanismo de consenso para os pontos de verificação de origem e destino. O voto de cabeça está relacionado ao voto correto e pontual do bloco mais recente. A recompensa do comitê de sincronização é dada por participar de um comitê de sincronização, e a recompensa do propositor é por propor um bloco no slot correto. Todos esses componentes têm pesos atribuídos que somam 64.

A recompensa total recebida por um validador é determinada somando os pesos dos cinco componentes de recompensa e dividindo por 64. Um validador que tenha feito votos pontuais de origem, destino e cabeça, proposto um bloco e participado de um comitê de sincronização pode receber **64/64 × recompensa\_base**.

O **saldo efetivo** é usado para calcular a proporção de recompensas e penalidades aplicadas a um validador. Ele é baseado no saldo atual do validador e no saldo efetivo anterior. O saldo efetivo máximo sempre será no máximo **32 ETH**. Mesmo que o saldo real seja de 1.000 ETH, o saldo efetivo ainda será 32 ETH.

**Ações dos Validadores e Frequência de Recompensa**

Os validadores podem ganhar recompensas realizando as seguintes ações, mostradas na tabela abaixo:

| **Ação** | **Frequência da recompensa** |
| --- | --- |
| Nova proposta de bloco | A cada *epoch* |
| Atestação de bloco | A cada *epoch* |
| Participação em comitê de sync | A cada 27,3 horas (256 *epochs*). No entanto, com 300.000 validadores, a chance de um validador ser escolhido é muito baixa e pode ocorrer, em média, a cada 22 meses. À medida que o número de validadores cresce, a probabilidade de seleção diminui proporcionalmente. |

**Comparação entre Nó Beacon e Nó Validador**

No Ethereum Beacon Chain (camada de consenso), existem dois tipos distintos de nós: **nós da Beacon Chain** e **nós validadores**. As principais diferenças entre esses tipos de nós são apresentadas na tabela a seguir:

| **Característica** | **Nós beacon** | **Nós validadores** |
| --- | --- | --- |
| Rede | Conectado via P2P a outros nós beacon | Conexão dedicada com um único nó beacon |
| Staking | Sem exigência de staking | Exige staking de ether para participar |
| Criação de bloco | Atesta validações e propaga blocos | Propõe e assina blocos |
| Operação | Leitura | Escrita |

Sendo uma blockchain baseada em PoS, o Ethereum deve ter a capacidade de receber depósitos dos usuários como staking. Essa exigência é atendida por meio de um **contrato de depósito**.

**Proof-of-Stake**

No mecanismo Proof-of-Stake (PoS), os validadores apostam seus fundos em um contrato inteligente na cadeia do Ethereum. Esse *stake* serve como colateral que pode ser perdido se o validador agir de forma desonesta ou negligente. Os validadores são responsáveis por verificar a validade de novos blocos e, ocasionalmente, criar e propagar novos blocos por conta própria.

O PoS possui várias vantagens em relação ao mecanismo PoW. Ele é mais eficiente em termos de energia, pois não requer consumo significativo de energia para fins computacionais. Também tem barreiras de entrada mais baixas, já que não exige hardware especializado e caro, como ASICs, para participar do processo de consenso. O PoS também reduz o risco de centralização ao encorajar mais nós a proteger a rede. Além disso, é necessário um baixo volume de emissão de ether para incentivar a participação porque o consumo de energia é baixo. No caso de um ataque de 51%, o PoS impõe penalidades econômicas por má conduta, tornando-o muito mais caro para os atacantes do que o PoW. A comunidade também pode usar mecanismos de recuperação social para restaurar uma cadeia honesta se um ataque de 51% ocorrer e sobrepujar as defesas criptoeconômicas.

Para se tornar um validador no sistema PoS do Ethereum, um usuário deve depositar **32 ETH** no contrato de depósito e usar três componentes de software: um **cliente de execução**, um **cliente de consenso** e um **validador**. Após o depósito, o usuário entra em uma **fila de ativação**, que também serve como um mecanismo para limitar o número de novos validadores ingressando na rede. Uma vez ativado, os validadores recebem novos blocos de seus pares na rede Ethereum. O papel do validador aqui é verificar a validade das transações no bloco recebido e sua assinatura antes de votar a favor do bloco e anunciar a atestação à rede.

Diferente do PoW, onde o tempo dos blocos é determinado pela dificuldade de mineração, o PoS possui uma **frequência de produção de blocos fixa**. O tempo é dividido em **slots de 12 segundos**, e **32 slots** compõem um **epoch** de 6,4 minutos.

Um período de **2.048 epochs**, aproximadamente **9,1 dias**, é chamado de *eek* ou “semana do Ethereum”. Ele pode ser usado para medir processos que levam muito tempo.

Podemos visualizar isso na **Figura 13.1** abaixo:

<**IMAGEM**>

**Figura 13.1**: Um *epoch* consiste em 32 *slots* (S1 a S32)

Em cada *slot*, um único validador é escolhido aleatoriamente por meio do RANDAO para propor um novo bloco e transmiti-lo para outros nós. Além disso, um comitê de validadores é selecionado aleatoriamente em cada *slot*, e seus votos são usados para determinar a validade e eventual aceitação do bloco proposto.

Note que nem todo validador participa da votação para cada bloco. Em vez disso, eles são atribuídos a um “comitê” para cada *epoch*, e cada comitê é atribuído aleatoriamente a um *slot* para atestar a validade do bloco proposto. Esses comitês são distribuídos entre os 32 *slots* em um *epoch*, com um máximo de **2.048 validadores por comitê**. Isso significa que cada validador atesta apenas um bloco por *epoch*, especificamente aquele atribuído ao *slot* do seu comitê. Consequentemente, cada *slot* ou bloco é atestado por **1/32 do conjunto total de validadores**.

Podemos visualizar isso na **Figura 13.2** abaixo:

<**IMAGEM**>

**Figura 13.2**: Um comitê atribuído a um *slot*

Além de atestarem o bloco principal atual (*head block*), os validadores também atestam outros dois blocos referidos como blocos de **checkpoint**. Cada *epoch* apresenta um único bloco de checkpoint que designa o bloco mais recente no início do *epoch*.

Durante cada *epoch*, os validadores validam dois checkpoints:

* o **bloco fonte** (*source*), e
* o **bloco alvo** (*target*).

Além dos blocos *head*, *source* e *target*, os validadores também incluem outras informações na mensagem de atestação, descritas a seguir:

* **aggregation\_bits**: Este é um vetor de bits dos validadores, onde a posição do bit mapeia para o índice do validador em seu comitê. O valor 0 ou 1 indica se o validador está ativo e de acordo com o proponente do bloco. Em outras palavras, se o validador assinou (atestou) a mensagem ou não.
* **Data**: Este campo contém vários elementos, incluindo o *slot*, o *index*, o *beacon block root*, o *source* e o *target*:
  + **Slot**: Número do *slot* atestado
  + **Index**: O número do comitê do validador em um dado *slot*
  + **Beacon block root**: Raiz hash do bloco recente (no topo da blockchain) observado pelo validador
  + **Source**: Visão do validador sobre o bloco mais recente justificado
  + **Target**: Visão do validador sobre o primeiro bloco no *epoch* atual
* **Signature**: Uma assinatura agregada de Boneh–Lynn–Shacham (BLS) das assinaturas dos validadores.

Após criar os dados, o validador pode mudar o bit em *aggregation\_bits* que corresponde ao seu próprio índice de validador de 0 para 1 para indicar sua participação no voto. Para completar o processo, o validador usa sua chave privada para assinar a atestação e a envia para a rede.

**Agregação de Atestações e Sub-redes (*Subnets*)**

Nem todo validador escuta todos os outros validadores da rede. Passar dados de atestação de cada validador para todos os outros validadores na rede geraria ruído excessivo e poderia causar congestionamento. Para lidar com esse problema, as atestações são agregadas dentro das chamadas “sub-redes” (*subnets*) antes de serem transmitidas. Essas sub-redes podem ser vistas como redes menores dentro da rede maior, criadas pela divisão da rede principal. Essas sub-redes tornam a comunicação eficiente ao reduzir a distância que os dados precisam percorrer. Cada comitê é dividido em **64 sub-redes** para permitir agregação.

Em cada *epoch*, um **agregador** é escolhido de cada sub-rede, cujo papel é reunir todas as atestações que contêm dados equivalentes aos seus próprios. O campo *aggregation\_bits* registra o remetente de cada atestação correspondente. Isso significa que todos os validadores que concordam com os dados do agregador agregam suas assinaturas em uma única assinatura. Por fim, o agregador transmite as atestações agregadas para a rede mais ampla. Uma vez que um validador é escolhido como proponente de bloco, ele reúne as atestações agregadas das sub-redes até o *slot* mais recente e as inclui no novo bloco.

Podemos visualizar o conceito de sub-redes e agregadores na **Figura 13.3** abaixo:

<**IMAGEM**>

**Figura 13.3**: Sub-redes e agregador

**Fluxo de Transações no PoS Ethereum**

Vamos agora ver como funciona o fluxo de transações no Ethereum com Proof-of-Stake, ou seja, no Ethereum após a Fusão.

O fluxo de execução de transações no PoS pode ser dividido em várias etapas, conforme descrito abaixo:

1. Uma **transação é composta e assinada** com a chave privada do usuário. Normalmente, isso é feito usando uma biblioteca como *ethers.js* ou *web3.js*. O usuário especifica a quantidade de *gas* que pagará para incentivar os validadores a incluírem a transação em um bloco.
2. Essa transação é **enviada a um cliente de execução** do Ethereum, que verifica se o saldo de ETH do remetente é suficiente e a validade de sua assinatura. Se a transação for válida, ela é adicionada ao *mempool* do cliente.
3. A transação é então **propagada para outros nós** via a rede de *gossip* P2P da camada de execução.
4. Um **nó aleatório na rede é escolhido usando o RANDAO** como o proponente de bloco para o *slot* atual. O papel desse nó é construir e transmitir o próximo bloco a ser adicionado à blockchain e atualizar o estado global. Observe que esse nó não é escolhido como parte do processo de execução da transação; ele é selecionado aleatoriamente pelo protocolo subjacente para cada *slot*.
5. O **cliente de execução reúne as transações** do *mempool* e as agrupa em um **payload de execução**, que é então executado localmente na EVM para produzir o novo estado.
6. O **cliente de execução envia esse payload de execução** ao cliente de consenso.
7. O **cliente de consenso (também chamado cliente beacon)** o encapsula dentro de um **bloco beacon**. Os blocos beacon também contêm informações sobre recompensas, penalidades, atestações e outros dados semelhantes.
8. Outros **nós de consenso recebem o bloco beacon** através da rede de *gossip* da camada de consenso.
9. Esses clientes **enviam o payload aos seus respectivos clientes de execução**, onde as transações são executadas novamente localmente na EVM para validar a mudança de estado proposta.
10. O **cliente validador determina e verifica se o bloco é válido** e se ele é o próximo bloco lógico correto com base no peso mais pesado de atestações, definido pela regra de escolha de bifurcação, ou seja, **LMD GHOST (Última Mensagem Dirigida à Subárvore Mais Pesada Observada)**.
11. O **bloco é então adicionado ao banco de dados local** de cada nó que o verificou.
12. Uma **transação é considerada “finalizada”** quando se torna parte de uma cadeia com um vínculo de supermaioria entre dois checkpoints.
13. **Uma vez que um número adequado de validadores tenha atestado um bloco**, ele é anexado à cabeça da cadeia e finalizado.

Podemos visualizar o fluxo de transações na **Figura 13.4** abaixo:

<**IMAGEM**>

**Figura 13.4**: Fluxo de transações no Ethereum após a Fusão

Leia a Figura 13.4 usando os números exibidos, que correspondem às etapas descritas acima no fluxo de transações.

Observe que os **checkpoints são estabelecidos no início de cada *epoch*** e exigem **66% de atestação** do total de ETH em staking da rede para se qualificarem como um vínculo de supermaioria.

**Finalidade (*Finality*) e Checkpoints**

Uma transação é considerada **“final”** quando está incluída em um bloco que **não pode ser alterado sem que uma quantidade significativa de ETH seja gasta**. O protocolo PoS do Ethereum alcança isso por meio dos chamados **blocos de checkpoint**.

O primeiro bloco de cada *epoch* serve como um **checkpoint**, e os validadores votam em pares de checkpoints que consideram válidos. Por exemplo, suponha que um par de checkpoints receba votos representando **pelo menos dois terços do ETH total em staking**. Nesse caso, os checkpoints são promovidos, com o bloco mais recente se tornando “justificado” e o bloco anterior, já justificado no *epoch* anterior, se tornando “finalizado”. Para reverter um bloco finalizado, um invasor precisaria comprometer **pelo menos um terço do ETH total em staking**, o que é economicamente proibitivo.

Como a **finalidade requer uma maioria de dois terços**, um invasor poderia impedir a rede de atingir a finalidade votando com apenas um terço do *stake* total. Como defesa contra essa ameaça, um **mecanismo de vazamento por inatividade (*inactivity leak*)** é ativado quando a cadeia falha em finalizar por mais de quatro *epochs*. O vazamento por inatividade **drena gradualmente o ETH em staking dos validadores** como penalidade por votarem contra a maioria, permitindo que a maioria dos validadores recupere a maioria de dois terços e finalize a cadeia.

**Finalidade** pode ser definida como a garantia de que um bloco, uma vez finalizado, **não será revertido**. Para alcançar isso, um mecanismo chamado **Casper the Friendly Finality Gadget (Casper-FFG)** é implementado na camada de consenso. Um artigo sobre o assunto está disponível no seguinte URL:

<https://ethresear.ch/uploads/default/original/1X/1493a5e9434627fcf6d8ae62783b1f687c88c45c.pdf>

**Finalidade no Gasper**

No **Gasper**, a finalidade é alcançada usando a seguinte lógica:

* Se **pelo menos dois terços do ETH total em staking** votarem a favor de um bloco, ele se torna **“justificado”**. Embora blocos justificados sejam geralmente estáveis, ainda podem ser revertidos em determinadas condições.
* Se outro bloco for justificado **em cima de um bloco já justificado**, ele então se torna **“finalizado”**. Uma vez que um bloco é finalizado, ele é **confirmado na cadeia canônica** e **não pode ser revertido**, a menos que um atacante destrua uma quantidade substancial de ether.

Vale destacar que **nem todo slot resulta em um estado “justificado” ou “finalizado” para blocos**. Esses estados são alcançados **apenas pelo primeiro bloco de cada *epoch***, conhecido como **bloco de checkpoint**. Quando um bloco de checkpoint é promovido para justificado, ele deve estar ligado ao checkpoint anterior. Isso significa que **pelo menos dois terços do total de ETH em staking** devem votar que o checkpoint B é o descendente legítimo do checkpoint A. Como resultado, o bloco de checkpoint anterior é finalizado, e o mais recente é justificado.

Ser um validador na rede é um compromisso sério que exige **manter hardware apropriado e conectividade com a internet** para participar do processo de validação e proposta de blocos. Em troca por seus serviços, os validadores são **recompensados com ether**, o que aumenta seu saldo em staking.

**Riscos e Penalidades para Validadores Desonestos**

Contudo, abrir a participação publicamente na rede como validador também traz riscos, pois pode **expor a rede a ataques de usuários maliciosos**. Como mecanismo de prevenção contra ataques, os validadores são penalizados por meio de *slashing* do seu *stake* se deixarem de participar da rede conforme esperado. Além disso, eles correm o risco de **perder todo o seu stake** se se envolverem em comportamentos desonestos. As duas principais formas de comportamento desonesto são:

* **Equivocação** (*equivocating*), que significa propor mais de um bloco em um único *slot*.
* **Atestações contraditórias**, onde um validador envia atestações conflitantes.

O **montante de ETH punido (slashed)** depende do número de validadores penalizados simultaneamente, conhecido como **penalidade por correlação** (*correlation penalty*). Essa punição pode variar de uma penalidade menor de cerca de **1% do stake** até a perda de **100% do stake** em um evento de *slashing* em massa. A penalidade é imposta **na metade do período de saída forçada**. Há uma penalidade imediata de até **0,5 ETH no primeiro dia**. No **18º dia**, a penalidade por correlação é aplicada, e a **expulsão da rede ocorre em aproximadamente 5 semanas (36 dias)**. Além disso, validadores que estão presentes na rede mas **não enviam votos** recebem **pequenas penalidades diárias por atestação**. Como resultado, um ataque coordenado à rede seria **proibitivo financeiramente para o atacante**.

**Regra de Escolha de Bifurcação: LMD GHOST**

A **regra de escolha de bifurcação** no Ethereum é chamada de **LMD GHOST**. Lembre-se de que discutimos a **Greediest Heaviest Observed SubTree (GHOST)** no contexto da mineração no Ethereum no Capítulo 9, *Arquitetura do Ethereum*. O **LMD GHOST** é uma variante do GHOST que foi implementada no Ethereum com algumas modificações.

Mais informações também podem ser encontradas em:

<https://ethereum.org/en/whitepaper/#modified-ghost-implementation>  
<https://arxiv.org/pdf/2003.03052>

O **LMD GHOST** governa um mecanismo de tratamento de bifurcações para garantir que, no caso de uma bifurcação, a **cadeia honesta correta seja automaticamente escolhida**. Como regra geral, a cadeia honesta é aquela que tem **o maior número de atestações e stake (peso)**. No evento de uma bifurcação, os clientes usam essa regra de escolha para selecionar a cadeia honesta correta. As bifurcações podem ocorrer devido a ações de participantes colusivos; no entanto, a **seleção aleatória de validadores pela Beacon Chain** mitiga isso até certo ponto, pois os validadores não sabem com antecedência quando serão selecionados.

**Interface P2P (rede)**

Este elemento trata das interfaces e protocolos de rede para a blockchain do Ethereum. Existem **três principais dimensões** abordadas pelo desenvolvimento dos elementos de rede/P2P do Ethereum:

* O domínio de *gossip*
* O domínio de descoberta
* O domínio de solicitação/resposta (*request/response*)

Atualmente, o **libP2P** é utilizado em diversos clientes para esse propósito. Mais detalhes sobre esse tópico estão disponíveis em:

<https://libp2p.io>

A especificação de rede também cobre os aspectos essenciais de uma **rede de testes (testnet)** onde múltiplos clientes podem ser executados simultaneamente, ou seja, a **interoperabilidade da testnet** e o **lançamento da mainnet**. Para alcançar interoperabilidade, todas as implementações de clientes do Ethereum devem **suportar o transporte TCP do libp2p**. Isso também deve estar habilitado para conexões tanto **de entrada** quanto **de saída**. Conexões de entrada também são chamadas de **conexões *inbound*** ou **conexões ouvintes** (*listening connections*). Conexões de saída são chamadas de **conexões *outbound*** ou **conexões de discagem** (*dialing connections*). Os implementadores podem optar por **não implementar um esquema de endereçamento IPv4 para a mainnet**. O Ethereum pode implementar um recurso de conectividade de entrada apenas para IPv6, mas os clientes devem oferecer suporte a **conexões de entrada e saída para endereços IPv4 e IPv6**.

**SSZ e Criptografia BLS**

O **Simple Serialize**, ou **SSZ**, é o padrão de algoritmo para fornecer capacidade de serialização para todas as estruturas de dados no software cliente do Ethereum. O SSZ tem suporte para muitos tipos de dados, como:

* Booleanos (*bool*)
* Inteiros sem sinal (*uint8*, *uint16*, *uint32*, *uint64*)
* *Slices*, arrays, estruturas (*structs*) e ponteiros

A **criptografia BLS (BLS12-381)** é usada extensivamente no Ethereum para oferecer **segurança** e **verificação eficiente de assinaturas digitais**. As assinaturas BLS permitem a **agregação de assinaturas criptográficas**, contribuindo para a escalabilidade da rede. A BLS é usada por clientes validadores para assinar mensagens, que então são agregadas e verificadas de forma eficiente na rede distribuída, aumentando assim a eficiência geral da rede.

Uma implementação em Go do emparelhamento BLS12-381 está disponível em:

<https://github.com/phoreproject/bls>

**The Merge (A Fusão)**

A **Beacon Chain** – a camada de consenso – está funcionando de forma independente desde dezembro de 2020 e implementa PoS ao **coordenar uma rede de validadores** usando dados da rede Ethereum. A **Fusão** combinou essas redes, permitindo ao Ethereum fazer a transição para PoS à medida que **clientes de execução e consenso trabalham juntos** para verificar o estado da rede.

Na **Figura 13.5** abaixo, podemos visualizar como o Ethereum era antes da Fusão e como ele se apresenta após a Fusão:

<**IMAGEM**>

**Figura 13.5**: Ethereum antes da Fusão vs. depois da Fusão

Observe que **não é mais possível executar apenas um cliente de execução**. Após a Fusão, tanto os **clientes de execução quanto os de consenso devem ser executados juntos** para permitir o acesso à rede Ethereum, conforme mostrado abaixo na **Figura 13.6**:

<**IMAGEM**>

**Figura 13.6**: Arquitetura de nó completo do Ethereum após a Fusão

Como mostrado no diagrama anterior, após a Fusão, **dois clientes separados** são executados no nó completo:

* **Cliente de execução**: Os clientes Ethereum originais baseados em PoW, que continuam lidando com a execução e o processamento de blocos, manutenção de *mempools* e sincronização de blocos. O componente PoW foi removido.
* **Cliente de consenso**: Este é o cliente da Beacon Chain responsável por realizar o consenso PoS, garantindo a formação da cadeia canônica, propagação de blocos, atestação de blocos e recebimento de recompensas de validadores.

Esses clientes se comunicam entre si por meio da **Engine API**.

A seguir, o diagrama mostra os três tipos de nós e seus relacionamentos:

<**IMAGEM**>

**Figura 13.7**: Diferentes tipos de nós no Ethereum e seu relacionamento

Também há uma mudança na **arquitetura do bloco** após a Fusão, ou seja, a junção da Beacon Chain com a antiga cadeia PoW, mostrada abaixo:

<**IMAGEM**>

**Figura 13.8**: Arquitetura do bloco do Ethereum antes e depois da Fusão (mostrando os campos principais)

Como mostrado no diagrama anterior, após a Fusão, a rede **não possui mais blocos PoW**. Em vez disso, o conteúdo dos antigos blocos PoW é **integrado aos blocos produzidos na Beacon Chain**. Essa mudança faz da Beacon Chain a **nova camada de consenso primária do Ethereum**, substituindo a camada de consenso PoW anterior.

A **frequência de produção de blocos** agora é de **12 segundos**, em vez da média de 13 segundos do PoW.

**Estrutura dos Blocos Após a Fusão**

Os blocos da camada de consenso da Beacon Chain contêm **ExecutionPayloads**, que servem como o equivalente, após a Fusão, dos blocos na antiga cadeia PoW. Para os usuários, a interação com o Ethereum continua ocorrendo por meio desses **ExecutionPayloads**.

As **transações nessa camada ainda são processadas pelos clientes da camada de execução**.

Alguns campos que estavam presentes nos cabeçalhos de blocos PoW **não são mais relevantes para o PoS** e, portanto, não são utilizados. Para evitar causar muita disrupção nas ferramentas e infraestrutura existentes, esses campos **não são removidos completamente da estrutura de dados**, mas sim definidos como **0** ou seu equivalente na estrutura. Esses campos incluem:

* **ommers**
* **ommer hash**
* **difficulty**
* **nonce**

O campo **mixHash** agora contém o valor do **RANDAO**.

O **opcode BLOCKHASH** permanece disponível após a Fusão, mas sua **pseudoaleatoriedade é significativamente mais fraca**, já que não é mais gerada por meio do processo de hashing do PoW. O **opcode DIFFICULTY** (0x44) foi atualizado e renomeado para **PREVRANDAO**. Ele retorna a saída do **beacon de aleatoriedade fornecido pela Beacon Chain**, tornando-se uma fonte de aleatoriedade mais forte do que o BLOCKHASH, embora ainda possa ser enviesada. O valor exposto por PREVRANDAO é armazenado no **ExecutionPayload**, onde antes o mixHash – um valor associado ao cálculo do PoW – era armazenado. O campo mixHash do *payload* foi renomeado para PREVRANDAO.

**Tamanho de Bloco e Limites de Gas**

O **tamanho-alvo de cada bloco** é definido em **15 milhões de unidades de gas**. No entanto, o **tamanho real dos blocos pode variar** com base na demanda da rede, desde que permaneça **dentro do limite máximo de 30 milhões de unidades de gas por bloco**.

Para manter um tamanho de bloco apropriado, o **total de gas utilizado por todas as transações em um bloco deve ser menor que o limite de gas do bloco**. Essa restrição é essencial porque evita que os blocos se tornem excessivamente grandes. Se os blocos pudessem crescer sem limites, nós com menos poder computacional não conseguiriam acompanhar a rede devido às suas limitações de hardware.

Quanto maior o bloco, **mais poder computacional é necessário para processá-lo**. Nós com menor capacidade perderiam blocos, pois um bloco deve ser processado dentro do próximo *slot* de tempo designado. Esses requisitos por mais capacidade computacional acabariam **centralizando a rede**. Portanto, a restrição no tamanho do bloco ajuda a **prevenir a centralização**, que poderia ocorrer se apenas usuários com hardware de ponta pudessem processar os blocos, resultando no uso da rede apenas por uma minoria. **Limitar o tamanho dos blocos ajuda a descentralizar a rede**.

Todas as mudanças na estrutura do bloco estão descritas na **EIP-3675**:  
<https://eips.ethereum.org/EIPS/eip-3675#block-structure>

**Campos dos Blocos na Beacon Chain**

Conforme discutido, após a Fusão, os blocos contêm **ExecutionPayloads**. O *payload* de execução substitui os antigos blocos PoW como o mecanismo de envio de transações e atualizações de estado. Os blocos da Beacon Chain, portanto, contêm tanto a lógica da Beacon quanto os dados de execução.

O cabeçalho do bloco da Beacon Chain contém os seguintes campos principais:

* **slot** – Número do *slot* em que o bloco foi proposto.
* **proposer\_index** – Índice do validador que propôs o bloco.
* **parent\_root** – Raiz do bloco pai (anterior).
* **state\_root** – Raiz do estado após o bloco ser processado.
* **body\_root** – Raiz Merkle do corpo do bloco, que contém dados como atestações, *slashings*, votos e o *payload* de execução.

O corpo do bloco da Beacon Chain contém diversas partes, incluindo:

* **Randao reveal** – O valor de aleatoriedade revelado pelo propositor.
* **eth1\_data** – Dados da cadeia de origem (anteriormente PoW).
* **graffiti** – Campo de dados arbitrário de 32 bytes.
* **proposer slashings** – Lista de evidências de proposições múltiplas por validadores.
* **attester slashings** – Lista de evidências de atestações duplas ou cercadas (*surrounding*).
* **attestations** – Lista de atestações recebidas.
* **deposits** – Depósitos de ETH vindos do contrato de depósito.
* **voluntary exits** – Validadores que desejam sair voluntariamente.
* **sync aggregate** – Agregações de comitês de sincronização.
* **execution\_payload** – Contém os dados de execução (transações, raiz do estado, recibos etc.).
* **bls\_to\_execution\_changes** – Lista de alterações de credenciais de retirada.
* **execution\_payload\_header** – Cabeçalho com hash do *payload* de execução (incluído em blocos não-executáveis).
* **withdrawals** – Lista de retiradas validadas.

**Campos do Payload de Execução**

O campo **execution\_payload** contém os seguintes subcampos:

* **parent\_hash** – Hash do bloco pai.
* **fee\_recipient** – Endereço do beneficiário das taxas (anteriormente minerador, agora validador).
* **state\_root** – Raiz Merkle do estado após execução.
* **receipts\_root** – Raiz Merkle dos recibos das transações.
* **logs\_bloom** – Filtro Bloom para logs de eventos.
* **prev\_randao** – Aleatoriedade do último *epoch*.
* **block\_number** – Número do bloco.
* **gas\_limit** – Limite de gas do bloco.
* **gas\_used** – Total de gas utilizado no bloco.
* **timestamp** – Carimbo de tempo da criação do bloco.
* **extra\_data** – Campo de dados arbitrário.
* **base\_fee\_per\_gas** – Tarifa base (de EIP-1559).
* **block\_hash** – Hash do bloco.
* **transactions** – Lista de transações.
* **withdrawals** – Lista de retiradas (EIP-4895).

Esses campos mantêm a compatibilidade com as estruturas de bloco anteriores, embora alguns, como *difficulty*, *nonce* e *uncles*, tenham sido desativados ou tornados obsoletos.

**Transações e Retiradas**

As **transações** no Ethereum pós-Fusão permanecem compatíveis com as versões anteriores. Os tipos de transação incluem:

* Transações legadas (pré-EIP-2718)
* Transações com tipo definido (EIP-2718)
* Transações com *access list* (EIP-2930)
* Transações com tarifa base (EIP-1559)

As **retiradas (withdrawals)** foram introduzidas com a atualização **Shanghai/Capella**, que ativou a capacidade de retirar fundos do staking.

Cada entrada na lista de retiradas contém:

* **index** – Índice da retirada.
* **validator\_index** – Índice do validador que retira.
* **address** – Endereço de execução para o qual os fundos são enviados.
* **amount** – Quantia em gwei a ser retirada.

As retiradas podem ser automáticas (quando o validador sai) ou parciais (quando o saldo excede 32 ETH e o excedente pode ser retirado).

**Atualizações Pós-Fusão**

Após a Fusão, o Ethereum continuou a evoluir com várias atualizações importantes de rede. Essas atualizações visam não apenas **melhorar a eficiência e segurança da rede**, mas também **introduzir recursos adicionais**, como a capacidade de retirada de *staking*.

**Atualização Capella/Shanghai**

A atualização **Shanghai/Capella**, ativada em **abril de 2023**, foi uma das primeiras grandes atualizações após a Fusão. Ela introduziu **retiradas de staking**, permitindo que os validadores que apostaram ETH na Beacon Chain pudessem, finalmente, sacar seus fundos. A atualização consistiu em duas partes:

* **Shanghai**: Implementada na **camada de execução**.
* **Capella**: Implementada na **camada de consenso**.

A seguir estão os principais recursos introduzidos por essa atualização:

* **Retiradas de staking** (parciais e completas) — **EIP-4895**.
* **Melhorias no custo de gas** para certas operações.
* **Modificações nos testes e validações de contrato inteligente**.

Com o **EIP-4895**, a camada de consenso pode enviar informações de retirada diretamente para o *payload* de execução como uma lista de saques. Isso permite que o ETH seja **automaticamente creditado na camada de execução**, sem exigir que o validador envie uma transação por conta própria.

**Retirada de Staking**

Existem dois tipos de retirada para validadores:

* **Retirada parcial**: quando o saldo de um validador excede 32 ETH, o excedente é automaticamente transferido para o endereço de retirada configurado. O validador continua participando da rede.
* **Retirada completa**: quando um validador envia uma solicitação de saída voluntária (voluntary exit). Após passar por um período de espera, seu saldo total é transferido para o endereço de retirada, e ele **deixa o conjunto de validadores ativos**.

Para que as retiradas ocorram, o validador precisa configurar uma **chave de credencial de retirada compatível com a camada de execução**. Isso geralmente envolve converter uma chave BLS de retirada para uma chave de tipo de endereço Ethereum (0x01). A Beacon Chain monitora esses endereços e inicia as retiradas com base na fila.

**Fila de Retirada**

A capacidade de processar retiradas é limitada para proteger a rede de **ataques de massa ou instabilidade**. Assim como existe uma **fila de entrada** para novos validadores, há também uma **fila de saída**. O número de retiradas permitidas por *epoch* é limitado, e a ordem é determinada por um **índice sequencial de retirada** atribuído a cada validador que sai.

**Próximas Atualizações: Cancun–Deneb e Além**

As atualizações planejadas para o futuro incluem **Cancun (camada de execução)** e **Deneb (camada de consenso)**. A principal proposta dessas atualizações é a implementação do **proto-danksharding**, introduzido pela **EIP-4844**.

**EIP-4844 (Proto-Danksharding)**

O **EIP-4844** é um passo intermediário rumo ao **sharding completo**, que implementa **blobs de dados temporários** para aliviar a carga de dados no Ethereum. Ele introduz:

* **Blobs de dados não permanentes** que podem ser adicionados a blocos.
* Um novo tipo de transação que carrega blobs fora do espaço do estado principal.
* Custos reduzidos de *gas* para operações que exigem grandes volumes de dados, como rollups.

Esses blobs não são acessíveis via EVM e **são armazenados apenas temporariamente**, o que os torna ideais para soluções de camada 2 que necessitam de disponibilidade de dados, mas não de armazenamento permanente.

O proto-danksharding é um precursor para o **danksharding completo**, que envolverá múltiplas “partes” de dados (shards) processadas em paralelo.

**Danksharding e Rollups**

O futuro da escalabilidade do Ethereum está fundamentado em dois pilares principais:

* **Rollups**
* **Danksharding**

Esses mecanismos permitirão que o Ethereum processe **milhares de transações por segundo**, mantendo a **descentralização e segurança**.

**Rollups**

**Rollups** são uma solução de escalabilidade de **segunda camada (L2)**. Eles funcionam **executando transações fora da cadeia principal** (off-chain), mas **postando dados de prova e/ou resultados** na cadeia principal (on-chain), garantindo segurança por herança.

Existem dois tipos principais de rollups:

* **Optimistic Rollups**: Assumem que as transações são válidas por padrão. No entanto, existe um **período de contestação**, durante o qual qualquer um pode provar que uma transação está incorreta. Exemplos: **Optimism, Arbitrum**.
* **ZK Rollups**: Utilizam **provas de conhecimento zero** para validar transações antes de enviá-las à camada 1. Essas provas garantem a validade computacional com um custo menor. Exemplos: **zkSync, StarkNet**.

Rollups já estão amplamente implantados no Ethereum e são considerados a **estratégia principal de escalabilidade de curto e médio prazo**.

**Danksharding**

**Danksharding** é o plano de **sharding de dados do Ethereum**, que visa **aumentar enormemente a capacidade de dados da rede**. Em vez de dividir a execução em múltiplos shards (como proposto inicialmente), o danksharding foca em **sharding apenas os dados**, mantendo um **único ambiente de execução**.

No danksharding:

* Os **blocos são estendidos com "blobs" de dados** – pedaços grandes que não são armazenados permanentemente nem acessíveis pela EVM.
* Esses blobs fornecem espaço barato para **dados de rollups**, aumentando sua eficiência.
* Um único **propositor constrói o bloco inteiro**, incluindo os blobs, o que elimina a complexidade de múltiplos shards independentes.
* Um novo papel chamado **construtor de blocos** é introduzido, permitindo que usuários especializados otimizem a inclusão de dados e transações.

A principal motivação para o danksharding é suportar rollups em larga escala. O Ethereum, com danksharding, **torna-se uma cadeia de dados para rollups**, enquanto a execução é cada vez mais movida para L2.

A Figura 13.9 abaixo ilustra essa visão do futuro do Ethereum:

<**IMAGEM**>

**Figura 13.9**: Visão do futuro do Ethereum com danksharding e rollups

**Resumo**

Este capítulo apresentou uma visão detalhada da **Fusão do Ethereum** e suas **implicações técnicas**, cobrindo:

* A arquitetura de **clientes de execução e consenso**
* O funcionamento da **Beacon Chain**
* O novo mecanismo de **staking com penalidades e recompensas**
* O fluxo de transações em PoS
* A regra de escolha de bifurcação **LMD GHOST**
* Atualizações como **Capella**, **Shanghai**, e os planos para **proto-danksharding**

A transição para Proof-of-Stake representa um marco significativo na história do Ethereum, consolidando sua posição como uma **plataforma de contratos inteligentes eficiente, segura e escalável**.