**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.

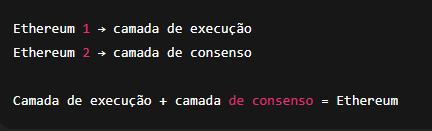
Clientes de execução são novas versões dos clientes Ethereum já em uso, como Geth, Nethermind e Besu. Esses clientes não possuem mais o componente PoW (ETHASH) e agora dependem exclusivamente de clientes de consenso para alcançar consenso.

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.

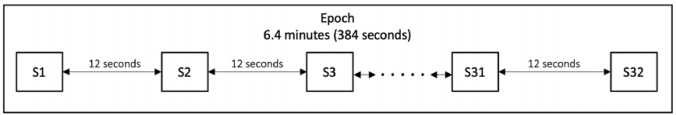
O PoS possui várias vantagens em relação ao mecanismo PoW. Ele é mais eficiente em termos de energia, pois não requer consumo significativo de energia para fins computacionais. Também tem barreiras de entrada mais baixas, já que não exige hardware especializado e caro, como ASICs, para participar do 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.

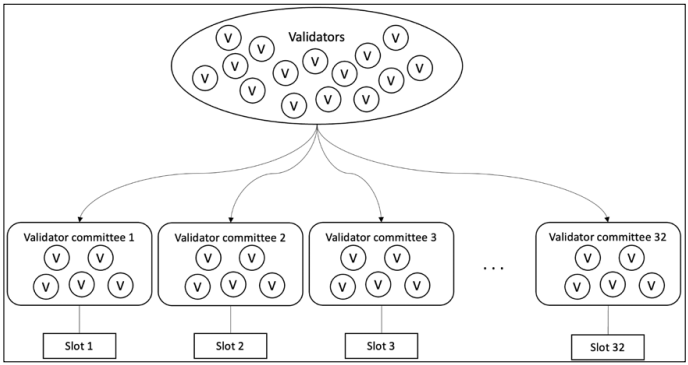
Podemos visualizar isso na **Figura 13.1** abaixo:

**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:

**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.

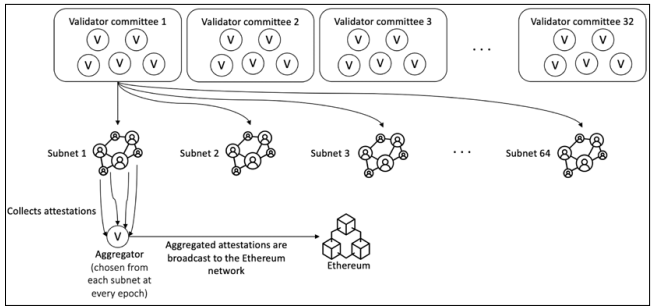
Após criar os dados, o validador pode mudar o bit em *aggregation\_bits* que corresponde ao seu próprio índice de validador de 0 para 1 para indicar sua participação no voto. Para completar o processo, o validador usa sua chave privada para assinar a atestação e a envia para a rede.

**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:



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

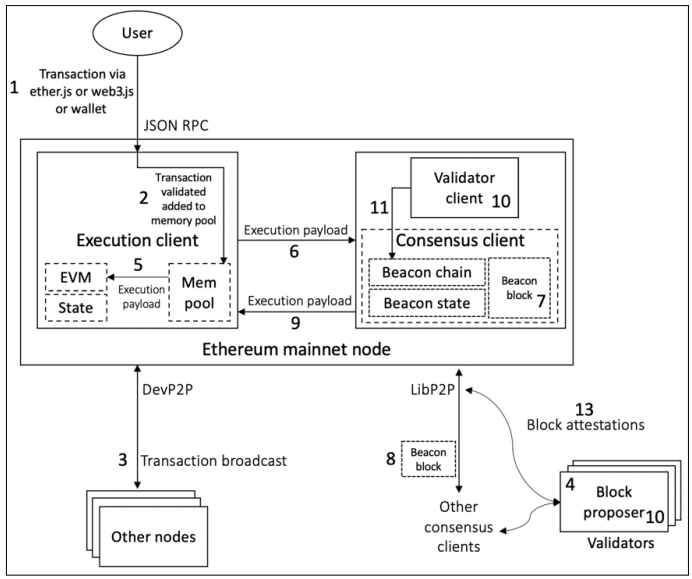
**Fluxo de Transações no PoS Ethereum**

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:

**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

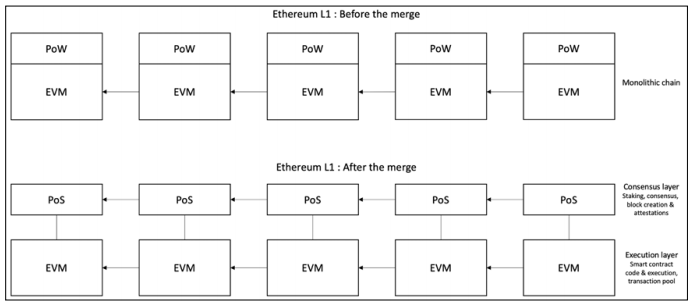
A **criptografia BLS (BLS12-381)** é usada extensivamente no Ethereum para oferecer **segurança** e **verificação eficiente de assinaturas digitais**. As assinaturas BLS permitem a **agregação de assinaturas criptográficas**, contribuindo para a escalabilidade da rede. A BLS é usada por clientes validadores 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>

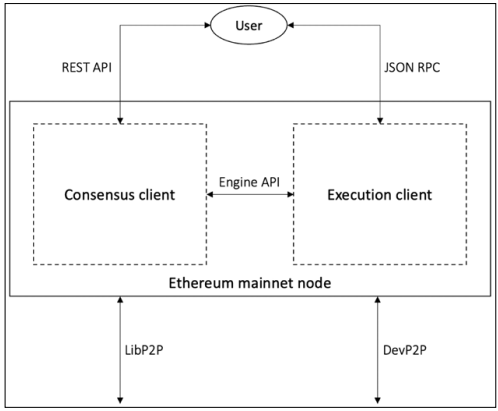
**The Merge (A Fusão)**

A Beacon Chain — a camada de consenso — tem funcionado de forma independente desde dezembro de 2020 e implementa a Prova de Participação (PoS) ao coordenar uma rede de validadores usando dados da rede Ethereum. A Fusão combinou essas redes, permitindo que o Ethereum fizesse a transição para PoS à medida que os 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 está depois da Fusão:



*Figure 13.5: Ethereum before The Merge vs. afer The Merge*

Note que não é mais possível executar um cliente de execução sozinho. Após a Fusão, tanto o cliente de execução quanto o de consenso devem ser executados juntos para permitir acesso à rede Ethereum, como mostrado abaixo na Figura 13.6:



*Figure 13.6: Ethereum full-node architecture afer The Merge*

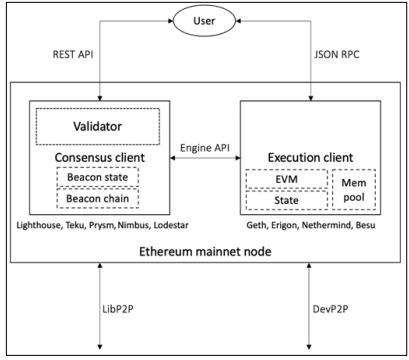
Conforme mostrado no diagrama anterior, após a Fusão, dois clientes distintos são executados no nó completo:

* **Cliente de execução**: os clientes originais do Ethereum com PoW que continuarão a lidar com a execução e processamento de blocos, manutenção de pools de memória e sincronização de blocos. O componente de PoW foi removido.
* **Cliente de consenso**: este é o cliente da Beacon Chain responsável por realizar o consenso PoS, garantir o surgimento da cadeia canônica, disseminação de blocos, atestação de blocos e recebimento de recompensas por parte dos validadores.

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

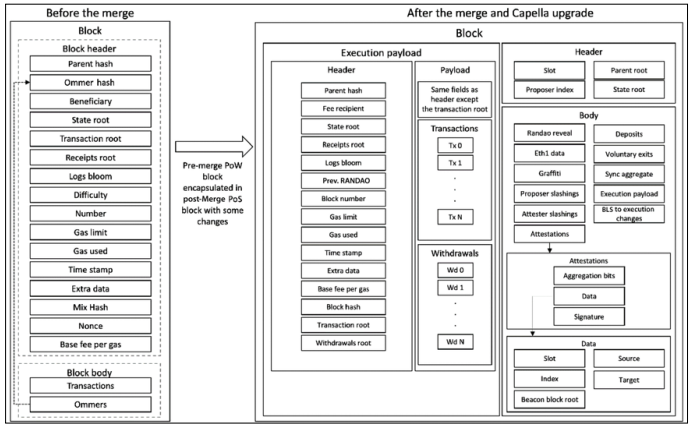
**A Fusão e Além**

O diagrama abaixo mostra os três tipos de nós e suas relações:



*Figure 13.7: Diﬀerent node types in Ethereum and their relationship*

Também há uma mudança na arquitetura dos blocos após a Fusão, ou seja, a junção da Beacon Chain e a antiga cadeia PoW, conforme mostrado abaixo:



*Figure 13.8: Ethereum block architecture before and afer The Merge (showing main fields)*

Como mostrado no diagrama anterior, após a Fusão, a rede não possui mais blocos PoW. Em vez disso, os conteúdos dos antigos blocos PoW são integrados nos blocos produzidos na Beacon Chain. Essa mudança torna a Beacon Chain a nova camada primária de consenso do Ethereum, assumindo o lugar da antiga camada de consenso baseada em PoW.

A frequência de produção de blocos agora é de 12 segundos, em vez dos 13 segundos médios sob PoW.

Os blocos da camada de consenso (Beacon Chain) contêm **ExecutionPayloads**, que servem como o equivalente pós-Fusão dos blocos na antiga cadeia PoW. Para os usuários, as interações com o Ethereum ocorrem por meio desses **ExecutionPayloads**.

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

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

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

O opcode **BLOCKHASH** permanece disponível após a Fusão, mas sua pseudoaleatoriedade é significativamente mais fraca, já que não é mais gerada através do processo de hash do PoW. O opcode **DIFFICULTY** (0x44) foi atualizado e renomeado como **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 tendenciosa. O valor exposto por **PREVRANDAO** é armazenado no **ExecutionPayload**, onde antes era armazenado o *mixHash*, um valor associado ao cálculo do PoW. O campo *mixHash* do payload é renomeado para **PREVRANDAO**.

O tamanho-alvo de cada bloco é definido em 15 milhões de unidades de gás. No entanto, o tamanho real dos blocos pode variar com base na demanda da rede, desde que permaneça dentro do limite de 30 milhões de unidades de gás. Para manter um tamanho de bloco adequado, a quantidade total de gás utilizada por todas as transações em um bloco deve ser menor que o limite de gás do bloco. Essa restrição é essencial porque impede que os blocos se tornem excessivamente grandes. Se os blocos pudessem crescer sem limites, nós com baixa capacidade de processamento não conseguiriam acompanhar a rede devido às suas limitações de hardware.

Quanto maior o bloco, mais poder computacional é necessário para processá-lo. Nós de baixa potência ficariam de fora, pois um bloco deve ser processado dentro do próximo intervalo de tempo designado. Esses requisitos por mais poder computacional, em última instância, centralizariam a rede. Portanto, a restrição de tamanho de bloco ajuda a evitar a centralização, o que poderia ocorrer quando apenas os usuários com hardware de ponta pudessem processar os blocos, resultando em uma rede utilizada apenas por poucos. Limitar o tamanho dos blocos ajuda a descentralizar a rede.

Todas as mudanças na estrutura dos blocos estão descritas aqui no EIP-3675:  
<https://eips.ethereum.org/EIPS/eip-3675#block-structure>

**A Fusão e Além**

Um bloco da blockchain Ethereum contém informações sobre o consenso, identificação, execução e outros metadados necessários para a execução do protocolo. O bloco contém os seguintes campos:

| **Campo** | **Descrição** |
| --- | --- |
| **slot** | O número do slot em que o bloco se encontra |
| **proposer\_index** | Validadores proponentes, ID |
| **parent\_root** | Hash do bloco anterior |
| **state\_root** | Hash raiz da árvore de estado |
| **body** | Contém diversos campos, descritos a seguir |

O corpo contém vários campos conforme mostrado abaixo:

| **Campo** | **Descrição** |
| --- | --- |
| **randao\_reveal** | Um valor aleatório verificável fornecido pelo validador proponente para ser misturado ao RANDAO. Usado para a próxima seleção de validador |
| **eth1\_data** | Informações do contrato de depósito incluindo root de depósito, contagem de depósitos e blockhash |
| **graffiti** | Campo de dados arbitrários – pode ser usado para escrever qualquer informação |
| **proposer\_slashings** | Lista de validadores a serem punidos (*slashed*) |
| **attester\_slashings** | Lista de validadores a serem punidos |
| **attestations** | Lista de atestações para o bloco atual |
| **deposits** | Lista de novos depósitos no contrato de depósito |
| **voluntary\_exits** | Lista de validadores que estão saindo da rede |
| **sync\_aggregate** | Uma assinatura e um vetor de bits representando o comitê de sincronização (subconjunto de validadores) |
| **execution\_payload** | Transações enviadas pelo cliente de execução |
| **bls\_to\_execution\_changes** | Contém uma mensagem para modificar as antigas credenciais BLS para o novo formato de credenciais de saque no formato de endereço de execução, a fim de facilitar retiradas de saldo |

A Beacon Chain depende de um mecanismo para gerar aleatoriedade no protocolo chamado **RANDAO**. O RANDAO funciona como um acumulador que continuamente acumula contribuições de aleatoriedade de várias fontes. O proponente inclui uma contribuição aleatória ao valor atual do RANDAO com cada novo bloco. O valor do RANDAO mantido pela Beacon Chain é atualizado a cada novo bloco adicionado à cadeia. Essa atualização ocorre ao misturar o valor de **randao\_reveal** fornecido pelo validador proponente do bloco. Assim, o valor RANDAO acumula aleatoriedade de todos os proponentes de blocos ao longo do tempo.

O campo **attestations** em um bloco consiste em todas as atestações associadas ao bloco. O próximo validador é selecionado a cada época com base na aleatoriedade do RANDAO.

A estrutura de dados da **atestation** contém diversos campos conforme mostrado abaixo:

| **Campo** | **Descrição** |
| --- | --- |
| **aggregation\_bits** | Uma lista de validadores atestantes participantes |
| **data** | Um campo que consiste em vários subcampos (descritos a seguir) |
| **signature** | A assinatura agregada de todos os validadores atestantes |

O campo **data** presente na atestação consiste nos elementos mostrados abaixo:

| **Campo** | **Descrição** |
| --- | --- |
| **slot** | O número do slot ao qual a atestação está associada |
| **index** | Os índices dos validadores atestantes |
| **beacon\_block\_root** | Hash raiz do bloco beacon contendo o campo de dados |
| **source** | O último checkpoint justificado |
| **target** | O bloco de época alvo mais recente |

O **execution payload** consiste em um cabeçalho e um conteúdo (payload), ambos contendo diversos campos. O payload contém transações e retiradas a serem processadas. Quando as transações no payload de execução são executadas, isso resulta na atualização do estado global.

Todos os clientes de execução executam as transações usando a EVM assim que a recebem do cliente de consenso para garantir a validade das transações. Na prática, isso significa assegurar que o novo estado resultante da execução da transação coincida com aquele presente no campo *state\_root* do novo bloco. Essa verificação permite aos clientes confirmar que o novo bloco é válido. Se for válido, ele é adicionado à blockchain.

O cabeçalho do execution payload contém diversos campos, que descrevemos a seguir:

| **Campo** | **Descrição** |
| --- | --- |
| **parent\_hash** | O hash do bloco pai |
| **fee\_recipient** | O endereço da conta do beneficiário que recebe as taxas de transação |
| **state\_root** | O hash raiz do estado global após aplicar as mudanças deste bloco |
| **receipts\_root** | O hash da trie de recibos de transações |
| **logs\_bloom** | A estrutura de dados contendo os logs de eventos |
| **prev\_randao** | Um valor usado para seleção aleatória de validadores |
| **block\_number** | O número do bloco atual |
| **gas\_limit** | O máximo de gás permitido neste bloco |
| **gas\_used** | A quantidade de gás usada neste bloco |
| **timestamp** | O horário de criação do bloco |
| **extra\_data** | Dados adicionais arbitrários como bytes brutos |
| **base\_fee\_per\_gas** | O valor base da taxa por gás |
| **block\_hash** | Hash do bloco de execução |
| **transactions\_root** | Hash raiz das transações no payload |
| **withdrawals\_root** | Hash raiz das retiradas no payload |

O **execution payload** contém os mesmos campos do cabeçalho, incluindo *parent\_hash*, *fee\_recipient*, *state\_root*, *receipts\_root*, *logs\_bloom*, *prev\_randao*, *block\_number*, *gas\_limit*, *gas\_used*, *timestamp*, *extra\_data*, *base\_fee\_per\_gas* e *block\_hash*. A diferença é que, em vez dos hashes raiz de transações e retiradas, ele contém as transações e retiradas reais a serem executadas.

| **Campo** | **Descrição** |
| --- | --- |
| **transactions** | Lista de transações a serem executadas |
| **withdrawals** | Lista de retiradas a serem executadas |

Não há código compartilhado entre os clientes de consenso e de execução. Eles são artefatos desenvolvidos separadamente. No entanto, há algumas tarefas comuns que eles realizam, diferindo apenas na pilha tecnológica utilizada. Por exemplo, tarefas relacionadas a rede e blockchain são as mesmas. No aspecto de rede, funções como descoberta de pares, gerenciamento de pares, gossip, resistência a ataques Sybil e prevenção de DoS são comuns em ambos os clientes. No aspecto de blockchain, temos funções como integridade da blockchain (ex.: manipulação de reorganização da cadeia), sincronização de blocos, pools de memória, manipulação de transição de estado e outras funções relacionadas.

Embora as funções que eles executam sejam, em parte, semelhantes e em alguns casos idênticas, a tecnologia usada para realizá-las é significativamente diferente de várias maneiras. Essas diferenças significam que, pelo menos por enquanto, os usuários devem executar ambos os clientes. Essas diferenças são mostradas na tabela a seguir:

| **Atributo** | **Cliente de Execução** | **Cliente de Consenso** |
| --- | --- | --- |
| **Protocolo da camada de rede** | DEVP2P | LIBP2P |
| **Protocolo de descoberta** | Discovery V4 | Discovery V5 |
| **Formato de serialização** | RLP | SSZ |
| **Funções de hash** | SHA256 | KECCAK |
| **Esquema de assinatura** | Esquema de assinatura ECDSA | Esquema de assinatura BLS |

Isso significa que, no futuro, há a possibilidade de que ambos possam ser fundidos e um único cliente emerja para lidar com o consenso e a execução.

Algumas soluções possíveis que foram propostas são:

Um cliente de consenso leve incorporado nos clientes de execução. Essa abordagem tornaria mais fácil para os usuários finais executarem um nó básico e reduziria os requisitos de sistema. No entanto, essa abordagem poderia resultar em maior complexidade arquitetural e vulnerabilidades de segurança.

* Combinar clientes de consenso e de execução. Essa abordagem tornaria simples para os usuários finais executarem um nó, já que seria apenas um executável único. No entanto, isso poderia aumentar a complexidade arquitetural e do código, reduzir a segurança, eliminar a separação de responsabilidades e causar complexidade de desenvolvimento.

Podemos eventualmente ver um único cliente; no entanto, por ora, são dois executáveis que os usuários precisam executar. Seja com clientes combinados ou separados, o ecossistema Ethereum está voltado ao crescimento, e o roteiro futuro parece bastante promissor.

Pós-Fusão, o Ethereum não utiliza mais PoW; ele utiliza **Gasper** — uma combinação de **Casper-FFG** e **LMD GHOST**. Gasper é uma integração do **Casper the Friendly Finality Gadget (Casper-FFG)** e do algoritmo de escolha de bifurcação **LMD-GHOST**, servindo como o mecanismo de consenso que protege o Ethereum PoS. A função do Casper-FFG é promover blocos para um estado “finalizado”, fornecendo garantia aos novos participantes da rede de que estão se sincronizando com a blockchain autoritativa (canônica). O algoritmo de escolha de bifurcação, por outro lado, utiliza a votação acumulada para facilitar a escolha da blockchain correta em caso de bifurcações. Gasper opera em um ambiente PoS no qual os nós (stakers) depositam ether como garantia, que pode ser perdida caso ajam de forma lenta ou desonesta ao propor ou verificar blocos. Gasper define as regras para recompensar e penalizar validadores, determinar quais blocos aceitar ou rejeitar e selecionar qual ramificação da blockchain continuar construindo mais blocos.

Apesar dos muitos resultados positivos da Fusão, há alguns problemas com o PoS do Ethereum, os quais introduzimos a seguir:

* Primeiramente, há uma preocupação crescente com a centralização. Esse problema surgiu porque o PoS do Ethereum exige pelo menos 32 ETH para executar um validador, o que significa que usuários com menos de 32 ETH não podem se tornar validadores. Eles precisam recorrer ao uso de serviços de staking, o que leva à centralização, já que cada vez mais usuários dependem desses serviços. Há alguns provedores de serviço de staking que compõem mais de 50% da participação total. Isso significa que, potencialmente, o controle está nas mãos de poucos provedores, o que resulta em centralização. Além disso, uma única autoridade central em uma rede financeira global não é ideal, e entregar a alguém o controle de suas chaves privadas — alguém que pode cometer erros, ser vulnerável a ataques, penalizado pelo protocolo da rede por comportamento inaceitável, ou mesmo censurado — pode ser bastante prejudicial ao investidor e ao ecossistema Ethereum como um todo.
* Validadores solo podem estar suscetíveis à perda de chaves se não forem gerenciadas adequadamente, especialmente se os usuários forem inexperientes. A custódia das chaves privadas é uma questão crítica, pois um validador solo pode ser vulnerável a ataques de malware ou comportamento descuidado. As chaves privadas poderiam ser mantidas com um provedor de serviço de staking, mas isso implica em abrir mão do controle das chaves — e esses provedores também não estão imunes a ataques e invasões.

Para lidar com o problema de centralização pós-Fusão, foi proposta a **Tecnologia de Validador Distribuído (DVT)**. Trata-se de uma técnica para distribuir as responsabilidades de um validador entre uma rede de múltiplos nós, o que resultará em um aumento significativo da resiliência do sistema. Em comparação com o método tradicional de depender exclusivamente de uma única máquina para executar o cliente validador, a utilização de DVTs oferece maior segurança e disponibilidade (*liveness*), o que não apenas torna o sistema menos vulnerável a problemas potenciais, como também garante um maior grau de estabilidade e robustez geral.

Para entender melhor a DVT, primeiro vamos compreender que existem dois riscos principais que os operadores de validadores enfrentam no Ethereum: roubo de chaves e falha de nós. No processo de validação, há duas chaves privadas envolvidas: a **chave de assinatura** e a **chave de saque**, que podem ser roubadas se não forem devidamente protegidas. Além disso, falhas nos nós podem ocorrer devido a bugs de software, vírus ou problemas de conectividade com a internet, resultando no *slashing* (penalização) da participação. Esses riscos podem desencorajar alguns usuários de executar um nó validador, especialmente dado o alto custo de participação, ou seja, 32 ETH. Como solução para essa barreira financeira, várias corretoras centralizadas começaram a oferecer soluções de staking custodial que permitem que usuários com menos de 32 ETH participem como validadores.

No entanto, essa conveniência tem o custo de confiar na corretora e dar a ela controle total sobre o ETH depositado. Essa abordagem não está alinhada com o ethos descentralizado do blockchain, resultando em centralização. Além disso, validadores se tornam pontos únicos de falha.

Validadores Ethereum participam do protocolo PoS assinando blocos ou atestações usando suas chaves privadas. Essas chaves só podem ser acessadas pelo software cliente do validador, que é responsável por agendar a criação e assinatura de mensagens de acordo com os deveres atribuídos ao validador. Essa configuração tradicional envolve diversos riscos, incluindo:

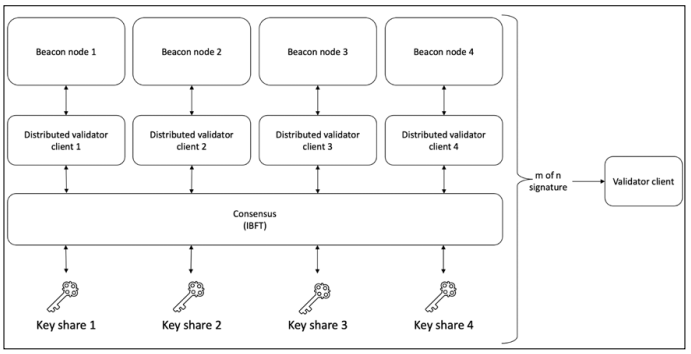
* A centralização da chave privada de staking em um único local, o que aumenta a probabilidade de um adversário potencial obter acesso à chave e causar conflitos por meio da criação de mensagens conflitantes, resultando na penalização (*slashing*) do depósito do validador.
* A necessidade de *stakers* que não operam seu próprio validador confiarem sua chave de staking a um provedor de serviços de staking terceirizado, criando uma dependência do operador para garantir a segurança da chave.
* Se a rede sofrer falhas de software, perda de conexão de rede, falhas de hardware e outros problemas semelhantes, a participação do validador é reduzida devido a “comportamento lento”. A falha do software cliente do validador em criar mensagens pontualmente resulta em um *inactivity leak*, que reduz a participação como penalidade.
* Há também o risco de um validador seguir uma bifurcação minoritária. Isso pode acontecer devido a uma falha no nó beacon ao qual o cliente do validador está conectado, o que faria com que o validador parecesse offline para o restante da rede PoS.

É aqui que a tecnologia DVT ajuda, pois permite que um validador Ethereum seja executado em vários nós em vez de apenas um, resultando em melhor segurança, ausência de ponto único de falha, transparência e responsabilidade devido ao mecanismo de consenso **IBFT (Istanbul Byzantine Fault Tolerant)** rodando entre os nós e melhorando a descentralização. O **Protocolo de Validador Distribuído** fornece uma forma de aliviar os riscos e preocupações mencionados acima e outros problemas presentes nas configurações tradicionais de clientes validadores.

A DVT utiliza quatro componentes principais:

* **Geração de chave distribuída**, para dividir a chave privada do validador em múltiplas partes, de modo que fique espalhada entre vários participantes, retirando o controle da chave de uma única parte;
* **Esquema de compartilhamento secreto de Shamir**, para reconstruir a chave a partir de múltiplas partes;
* **Computação multipartidária (MPC)**, que permite aos participantes assinarem mensagens sem reconstruir a chave privada completa em um único nó, melhorando assim a descentralização e reduzindo o risco de roubo de chave;
* O **algoritmo de consenso IBFT** é usado para chegar a um acordo sobre um bloco utilizando um nó beacon como proponente e, se a maioria aprovar, então o bloco é adicionado à cadeia.

O diagrama abaixo mostra a arquitetura de alto nível da DVT:



*Figure 13.9: Distributed validator technology architecture*

Como mostrado no diagrama da Figura 13.9, a DVT pode ser vista como um esquema de multiassinatura (**M de N**) para executar um nó validador.

Um esquema de assinatura limiar **M de N** permite que a chave de staking de um validador seja dividida em N partes, e cada co-validador mantém uma parte. Quando M dos N co-validadores chegam a um acordo por meio do mecanismo de consenso, a mensagem é assinada e, para os nós beacon, será simplesmente uma mensagem assinada padrão e válida por um validador. No entanto, aqui, a assinatura é aplicada somente depois que os co-validadores concordam, por meio de protocolo de consenso, em assinar a mensagem.

O mecanismo de consenso IBFT permite dividir as responsabilidades de um único validador entre vários co-validadores que coordenam entre si para chegar a um acordo antes de assinar qualquer mensagem. O protocolo de consenso IBFT também garante os requisitos de segurança e *liveness* da rede DV. Esses requisitos incluem segurança contra roubo de chave, proteção contra *slashing* e proteção contra impasses (*deadlocks*), assegurando a produção eventual de uma nova atestação ou bloco sob uma rede parcialmente síncrona com possibilidade de falhas bizantinas.

Essa combinação de consenso e um esquema de assinatura limiar garante que o consenso seja protegido por criptografia e que decisões sejam tomadas somente quando ao menos o número exigido de co-validadores concordar.

**A Fusão e Além**

**Sharding**

**Sharding** é o principal recurso de escalabilidade no Ethereum. Trata-se de uma melhoria em múltiplas fases na rede Ethereum para melhorar sua escalabilidade e capacidade. Essa atualização envolve tornar a cadeia do Ethereum compatível com *rollups*, tornar os *rollups* ainda mais econômicos e simplificar a operação de nós. O sharding permite que soluções de segunda camada ofereçam taxas de transação baixas, mantendo a segurança do Ethereum.

O plano original era introduzir 64 cadeias de fragmentos de dados EVM separadas conectadas à Beacon Chain, mas essa ideia não está mais sendo perseguida devido à sua complexidade de design, má experiência do usuário e desafios de segurança.

A nova abordagem é chamada de **danksharding**. Esta é a mais recente solução de escalabilidade proposta por Dankrad Feist. A ideia aqui é fragmentar dados em vez de criar muitas cadeias EVM de fragmentos separadas. Há uma proposta de melhoria do Ethereum, **EIP-4844**, que propõe isso; no entanto, ela não propõe o danksharding completo, mas sim o **proto-danksharding**, que permite alcançar uma solução de escalabilidade mais viável no curto prazo, antes que todos os desafios técnicos sejam resolvidos, antes de avançar para o danksharding completo. O proto-danksharding implementa alguns elementos do danksharding completo e pavimenta o caminho para sua implementação integral.

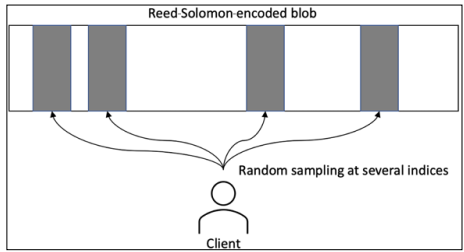
Os *rollups* são a única solução de escalabilidade sem necessidade de confiança (trustless) viável para o Ethereum no futuro próximo e médio — e possivelmente por mais tempo. Altas taxas de transação na cadeia principal do Ethereum, ou seja, a camada 1, representam uma barreira significativa à adoção por novos usuários e aplicações. A implementação do EIP-4844 terá um papel crucial em promover uma mudança para *rollups* em todo o ecossistema Ethereum. Isso também está alinhado com o futuro centrado em *rollups* do Ethereum. Uma vez implementado o EIP-4844, ele tornará as soluções de *rollup* da camada 2 mais compatíveis com a cadeia do Ethereum e alcançará uma grande escalabilidade.

Outro EIP relevante é o **EIP-4488**, que visa reduzir significativamente o custo de gás dos dados da transação (calldata) e também limitar a quantidade total de *calldata* por bloco. Isso significa que o custo das transações para soluções da camada 2 do Ethereum, como Optimism ou zk-rollups como zkSync, também será reduzido. Essa é, no entanto, uma solução de curto prazo, já que a solução definitiva para esse problema virá por meio do sharding. O sharding não foi implementado na atualização de Xangai em abril de 2023.

Em futuras atualizações do Ethereum, o **EIP-4844** implementa o proto-danksharding, que introduz **transações que transportam blobs (blob-carrying transactions)**, marcando o primeiro passo em direção à implementação completa do sharding no Ethereum. Esse EIP introduz um novo formato de transação para essas “transações com blobs”, que podem conter uma grande quantidade de dados que não são executados pela EVM, mas cujo compromisso (commitment) pode ser acessado. Esse formato foi projetado para ser totalmente compatível com o formato que será usado no sharding completo.

Os *rollups* são uma solução de escalabilidade promissora para o Ethereum e já reduziram significativamente as taxas para muitos usuários. No entanto, ainda são caros para alguns. A solução definitiva para as limitações dos *rollups* é o **sharding de dados**, que adicionaria 16 MB de espaço de dados dedicado à cadeia por bloco para uso dos *rollups*. No entanto, o sharding completo de dados provavelmente levará muito tempo para ser totalmente implementado. Esse EIP oferece uma solução provisória ao implementar o formato de transação que eventualmente será usado no danksharding completo, mas sem realmente implementar o sharding das transações. Em vez disso, os dados dessas transações farão parte da Beacon Chain e serão totalmente baixáveis por todos os nós de consenso, podendo ser descartados após um atraso relativamente curto.

Os *rollups* envolvem a publicação de uma grande quantidade de dados, o que poderia sobrecarregar os nós se eles fossem obrigados a baixá-los completamente. Isso aumentaria os requisitos de recursos e impactaria negativamente a descentralização. Para lidar com isso, o **Data Availability Sampling (DAS)** permite que nós, incluindo clientes leves, confirmem fácil e seguramente que todos os dados foram disponibilizados, sem a necessidade de baixá-los por completo:



*Figure 13.10: Data availability sampling*

O danksharding resolve o problema da disponibilidade de dados ao permitir que qualquer pessoa possa baixar os dados completos caso seja necessário. Ele usa uma técnica chamada **DAS**, onde um cliente amostra aleatoriamente um *blob* de dados com codificação de apagamento (usando códigos de Reed-Solomon) para verificar a disponibilidade dos dados. Em outras palavras, os dados são codificados com apagamento e estendidos usando os códigos de Reed-Solomon, o que permite representá-los como um polinômio e avaliá-lo em vários pontos. Essa técnica significa que agora apenas 50% dos dados codificados com apagamento são suficientes para garantir a disponibilidade total dos dados, pois os outros 50% podem ser reconstruídos.

Para que um invasor engane com sucesso o mecanismo DAS e faça os nós pensarem que todos os dados estão disponíveis, ele teria que ocultar mais da metade do bloco. Com muitas amostragens aleatórias realizadas, a probabilidade de menos da metade dos dados estar disponível se torna desprezível. Essa técnica funciona, mas o desafio aqui é como garantir que a codificação de apagamento tenha sido feita corretamente. E se o produtor de blocos tiver estendido o bloco com dados inúteis, e a amostragem retornar como válida, mas os dados reais forem lixo que não podem ser usados para reconstruir os dados? É aí que entra o **esquema de compromisso KZG**.

O esquema de compromisso KZG é um esquema de compromisso polinomial onde um provador pode se comprometer com um polinômio. Isso significa que o provador pode revelar o valor do polinômio em qualquer ponto dado e provar que ele é igual a um valor alegado. Uma vez que o provador tenha se comprometido com um valor, ele não pode alterar o polinômio depois. O provador só poderá fornecer provas válidas para esse polinômio específico e não poderá produzir uma prova de forma alguma ou, se conseguir, ela não será aceita pelo verificador.

O **sharding** também é conhecido por uma fase chamada **The Surge**, após **The Merge**.

As atualizações são divididas em várias fases, listadas abaixo:

* **The Merge**: Já realizada em 15 de setembro de 2022, que substituiu PoW por PoS.
* **The Surge**: Alcançar escalabilidade tornando o Ethereum compatível com *rollups*.
* **The Scourge**: Resolver centralização e Valor Máximo Extraível (MEV), e alcançar inclusão neutra de transações.
* **The Verge**: Tornar a verificação de blocos rápida e fácil utilizando SNARKs.
* **The Purge**: Remover dados históricos em excesso, simplificar o protocolo, eliminar dívidas técnicas e reduzir o custo de participação.
* **The Splurge**: Resolver quaisquer outros problemas restantes.

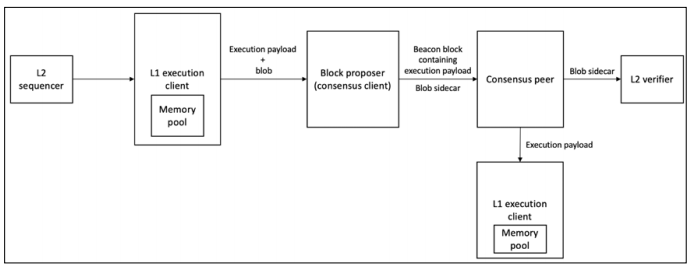
Atualmente, nas soluções de camada 2, o lote de transações é submetido à camada 1 via *calldata*. No entanto, com a introdução do **EIP-4844**, um novo tipo de transação chamado **"transação que transporta blob"** (*blob-carrying transaction*) foi adicionado, que carrega dados em forma de *blob* e é responsável por pagar a taxa de transação. A transação que transporta *blob* inclui um compromisso (commitment) para provar a existência dos dados no *blob*. É importante observar que o conteúdo adicional, que é o dado do *blob*, é separado da "transação do *blob*" e pode ser considerado um **sidecar** (componente acessório).

O **danksharding** é um design de sharding mais simples em comparação com abordagens anteriores como múltiplas cadeias de fragmentos. A ideia central aqui é que, em vez de aumentar o espaço para transações, o sharding fornece mais espaço para *blobs* de dados, os quais não são interpretados pelo protocolo Ethereum. A verificação dos *blobs* envolve apenas verificar que o *blob* está disponível e pode ser baixado, o que é obtido por meio do **DAS**. Esses *blobs* serão usados por *rollups* de camada 2 para permitir processamento de transações mais rápido e, assim, escalabilidade.

O ciclo de vida de uma transação com *blob* é descrito abaixo:

1. A transação de camada 2 é submetida pelos usuários da camada 2.
2. Um provedor de *rollup* então cria um lote de transações e o submete à camada 1.
3. Essa transação reside no pool de transações da camada 1 como uma transação com *blob* contendo dados de *blob*.
4. A transação com *blob* é encapsulada no **execution payload** do cliente de consenso (Beacon Chain), que processa isso e separa a transação com *blob* no payload de execução para execução na camada 1 e dados de *blob*.
5. O **execution payload** é executado na camada 1 e armazenado ali.
6. Os dados do *blob* são armazenados nos nós beacon por um mês, e a camada 2 mantém esses dados para uso futuro.

Podemos visualizar esse fluxo no nível de rede na Figura 13.11 abaixo:



*Figure 13.11: Blob transaction network flow*

A Figura 13.11 mostra várias etapas, que são explicadas a seguir:

1. O **sequenciador** da camada 2 submete transações regulares e transações com *blob* para a camada 1, que acabam no pool de transações da camada 1.
2. O mecanismo de execução da camada 1 pega a transação e lê o *execution payload* e os *blobs*.
3. O proponente de bloco (cliente de consenso) coleta isso e cria um bloco beacon contendo o *execution payload*. Ele também separa o *sidecar* do *blob* da transação com *blob*.
4. O par de consenso o recebe, e o *execution payload* é enviado ao mecanismo de execução da camada 1. O *execution payload* contém as transações para execução.
5. O *sidecar* do *blob* vai para o verificador da camada 2, que baixa esses *sidecars* e sincroniza os nós da camada 2.

Observe que há uma distinção entre o tempo de vida do *calldata* e o dos dados *blob*. O *calldata* está presente no *execution payload* de uma transação padrão da camada 1, enquanto os dados *blob* são mantidos na camada de consenso. Isso significa que o *blob* é armazenado em um cliente da camada de consenso como o Lighthouse ou outro nó similar, e não na camada de execução como o Geth ou cliente equivalente. Os nós da camada de consenso descartam os dados *blob* após um mês.

A principal inovação introduzida pelo **danksharding** é o **mercado de taxas unificado** (*merged fee market*), onde um único proponente seleciona todas as transações e dados para um slot específico, em vez de múltiplos fragmentos com diferentes proponentes de blocos. No entanto, isso pode levar a altos requisitos de sistema para os validadores. Para resolver essa questão, um conjunto separado de atores chamados **construtores de blocos** (*block builders*) fazem lances pelo direito de escolher o conteúdo do slot. O proponente de bloco apenas seleciona o cabeçalho válido com o maior lance, e o construtor de blocos processa o bloco inteiro. Essa abordagem permite que outros validadores e usuários verifiquem eficientemente os blocos por meio do DAS, enquanto reduz a carga computacional dos validadores.

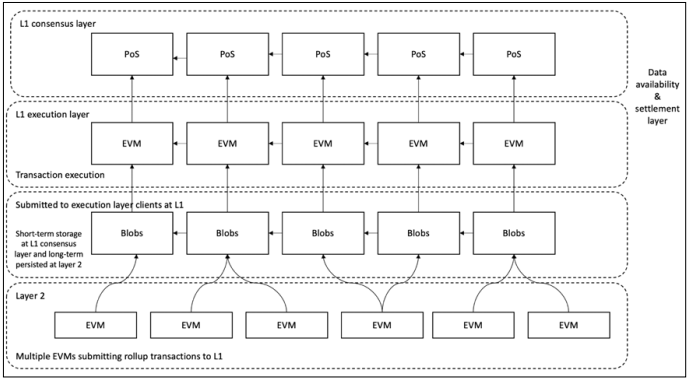
Essa separação entre **proponentes** e **construtores** é chamada de **Proposer-Builder Separation (PBS)**. Trata-se de uma solução promissora para lidar com o problema da censura e dos ataques de **MEV** (*Maximal Extractable Value*) na tecnologia blockchain. Com o PBS, as responsabilidades de construir blocos e propor blocos são separadas, sendo atribuídas a entidades/papéis distintos dentro da rede. Atualmente, no Ethereum, o proponente de bloco tem a liberdade de escolher quais transações incluir no bloco candidato avaliando as transações no *mempool* e selecionando aquelas que pagaram as maiores taxas.

Essa liberdade do proponente de bloco permite que ele empregue diferentes abordagens sofisticadas para priorizar injustamente transações ou até mesmo incluir as suas próprias, para explorar oportunidades como arbitragem em DEXs e liquidações, a fim de maximizar indevidamente seus lucros. Isso é chamado de **MEV**. Normalmente, devido à complexidade e ao alto custo de executar essas estratégias, validadores individuais não realizam essas atividades; em vez disso, *pools* centralizados veem isso como uma vantagem e agem oportunisticamente em nome dos participantes do *pool*. Para lidar com esse problema, o **PBS** foi proposto, que dissocia o papel de construção de blocos daquele de proposição de blocos. Nesse esquema, há uma nova categoria de entidades referidas como **construtores**, que:

1. Criam unidades de blocos executáveis, compostos por uma sequência de transações;
2. Em seguida, os construtores fazem lances, e o papel do proponente é simplesmente selecionar o maior lance e aceitar o corpo do bloco correspondente.

É fundamental observar que o proponente e todas as outras partes permanecem não informadas sobre os detalhes do corpo do bloco até depois de o cabeçalho ter sido selecionado. Essa confidencialidade pré-confirmação é a chave para evitar explorações de MEV.

Podemos visualizar a visão geral do **proto-danksharding** e do EIP-4844 na Figura 13.12 abaixo:



*Figure 13.12: Proto-danksharding overview*

Como os *rollups* publicam grandes volumes de dados, não é desejável forçar os nós a baixarem todos esses dados para garantir sua disponibilidade. Essa imposição aumentaria a carga sobre os nós, resultando em exigência de hardware de alto desempenho e, assim, risco de centralização, já que hardware de ponta pode estar ao alcance de apenas alguns usuários.

Para resolver isso, foi proposto o **DAS**. Uma abordagem direta para o DAS é verificar aleatoriamente algumas partes do bloco e, se estiverem válidas, então decide-se aceitar o bloco. No entanto, e se justamente aquele único fragmento que contém uma informação maliciosa não for verificado? Por exemplo, uma transferência ilegal de fundos para outra parte? Isso seria prejudicial à integridade do blockchain.

A solução é usar **códigos de correção de erro**, especificamente **códigos de apagamento de Reed-Solomon**, que expandem os dados e permitem a recuperação por correção de erro. Esta não é uma técnica nova; os códigos de Reed-Solomon são usados para correção de erros em muitos sistemas, incluindo tecnologias de comunicação e armazenamento como CDs, DVDs, códigos de barras, comunicação via satélite e tecnologia DSL.

Usando os códigos de apagamento de Reed-Solomon, todo o bloco pode ser reconstruído desde que 50% dos dados com codificação de apagamento estejam disponíveis para um bloco. Como podemos ver, isso reduz imediatamente os requisitos de armazenamento. Do ponto de vista da disponibilidade de dados, podemos concluir com altíssima probabilidade, após diversas amostragens aleatórias, que o bloco completo está disponível sem necessidade de fazer o download de todo o bloco. Até aqui, tudo bem; no entanto, pense em um cenário em que o produtor do bloco é malicioso e produz um bloco corrompido com dados inválidos. Isso significa que, embora a amostragem esteja correta, os dados reais dentro do bloco são lixo e não estão codificados com apagamento corretamente.

Para garantir que o proponente estendeu o bloco corretamente com os códigos de Reed-Solomon, utiliza-se o **esquema de compromisso KZG**, um esquema de compromisso polinomial. Compromissos polinomiais são discutidos no Capítulo 17, “Escalabilidade”, com mais detalhes.

Matematicamente, queremos garantir que os dados originais e os dados estendidos estão sobre o mesmo polinômio. O esquema de compromisso polinomial KZG permite que nos comprometamos com os dados originais do bloco e sua extensão (proveniente dos códigos de Reed-Solomon) e provemos que eles pertencem ao mesmo polinômio de grau baixo. Em termos simples, isso significa que os dados estendidos com os códigos de Reed-Solomon são válidos. Portanto, em resumo, o **DAS** garante a disponibilidade dos dados codificados com apagamento, e o **compromisso KZG** prova que os dados originais foram estendidos corretamente.

Como primeiro passo no **proto-danksharding**, os validadores do Ethereum devem fazer o download completo dos *blobs* dos fragmentos; no entanto, quando o **danksharding completo** for finalmente implementado, o validador só precisará realizar a amostragem de disponibilidade de dados (*DAS*).

Atualmente, no Ethereum, os validadores executam duas tarefas: construir e propor blocos, o que leva ao **MEV**. O **PBS** separa essas duas tarefas e atribui a construção de blocos aos construtores, e os proponentes selecionam os blocos. A ideia aqui é que os construtores possam ser nós especializados e de alta potência — ou até mesmo um único nó poderoso — que forneçam resistência à censura e garantam *liveness*. Ainda assim, os proponentes são de propósito geral e leves, o que significa que até mesmo usuários com hardware comum podem participar, melhorando assim a descentralização. Os construtores no PBS têm mais poder, o que pode levar à censura de transações. A solução **crList**, ou *censorship resistance list* (lista de resistência à censura), é proposta para lidar com essa questão. A ideia é que os proponentes publiquem transações do *mempool*, e os construtores sejam obrigados, por meio de algoritmos, a incluí-las em um bloco.

O esquema proposto é um PBS híbrido e compreende várias etapas descritas abaixo:

1. O proponente publica uma **crList** e seu resumo com todas as transações qualificadas do *mempool*;
2. O construtor propõe um corpo de bloco e submete um lance que inclui o hash do resumo da crList, o que prova que ele a observou;
3. O construtor vence o lance e submete o corpo do bloco. Observe que o construtor ainda não viu o corpo do bloco — ele não tem conhecimento de nenhuma transação contida nele;
4. O construtor publica seu bloco com prova de que incluiu todas as transações da **crList**;
5. Se a prova for válida, a regra de escolha de bifurcação **LMD GHOST** aceita o bloco, e os atestadores validam o corpo do bloco publicado.

Em resumo, o **PBS** permite um cenário onde, efetivamente, mesmo que a produção de blocos se torne centralizada devido a altos requisitos de hardware, a **verificação de blocos (validação)** continua sendo descentralizada e sem necessidade de confiança, ao mesmo tempo em que evita a censura de transações. O efeito geral desse esquema ainda é a descentralização da rede, mesmo que os **construtores** sejam centralizados. A ideia principal a se lembrar aqui é que **a descentralização da verificação de blocos é mais importante do que a descentralização da produção de blocos**. Espera-se que o Ethereum eventualmente evolua para tal mecanismo adotando o **danksharding** e o **PBS**. Com o danksharding, o Ethereum se tornará uma camada unificada de disponibilidade de dados e liquidação.

O danksharding cria uma integração mais estreita entre o bloco de execução da Beacon Chain e os shards. Um único construtor cria o bloco inteiro, e um proponente e um comitê votam nele por vez. O **PBS** é necessário para o **danksharding**, pois validadores comuns não conseguem lidar com um bloco cheio de *blobs* de dados. Com essa integração mais estreita dos blocos de execução e os shards, a disponibilidade de dados pode ser garantida de uma só vez. Com a introdução de grandes *blobs* de dados, efetivamente, obtém-se o mesmo efeito de um bloco de tamanho aumentado; no entanto, a EVM não interage com os dados dos *blobs*.

Apresentamos anteriormente o **LMD GHOST**. Ele faz parte do protocolo de consenso **Gasper**. Gasper é o novo mecanismo de consenso que protege a blockchain PoS do Ethereum após a Fusão. É a combinação do **Casper-FFG** com o algoritmo de escolha de bifurcação **LMD GHOST**. O mecanismo Casper-FFG finaliza blocos, garantindo que novos participantes da rede se sincronizem com a cadeia correta.

Por outro lado, o algoritmo de escolha de bifurcação usa votos acumulados para ajudar os nós a selecionar a cadeia correta em caso de bifurcações. O Gasper determina os incentivos e penalidades dos validadores, escolhe quais blocos aprovar ou rejeitar e define qual bifurcação da blockchain deve ser usada e expandida. Como o Ethereum pós-Fusão é baseado em PoS, os validadores fornecem *ether* como garantia de segurança, que pode ser confiscada se forem inativos ou desonestos ao validar ou propor blocos.

Vale notar que **não há necessidade de uma regra de escolha de bifurcação em condições normais**, onde há um único proponente de bloco para cada slot e validadores honestos o confirmam. No entanto, o algoritmo de escolha de bifurcação torna-se essencial em casos de assincronia significativa na rede ou quando um proponente de bloco age desonestamente, fazendo alegações inconsistentes. Nesses casos, o algoritmo de escolha de bifurcação é essencial para garantir a cadeia correta.

Como primeiro passo para alcançar o danksharding, o **proto-danksharding** é introduzido no **EIP-4844**. Ele implementa a lógica, os formatos de transação e as regras de verificação que estão na especificação completa do danksharding, mas **ainda não implementa o sharding real**. Isso significa que os usuários ainda precisam validar diretamente a disponibilidade completa dos dados. Um recurso importante introduzido por esse EIP é um novo tipo de transação chamada **“transação que carrega blob”** (*blob-carrying transaction*). Essa transação é semelhante a uma transação Ethereum normal, mas também possui um dado adicional chamado **blob**, que possui cerca de 125 kilobytes.

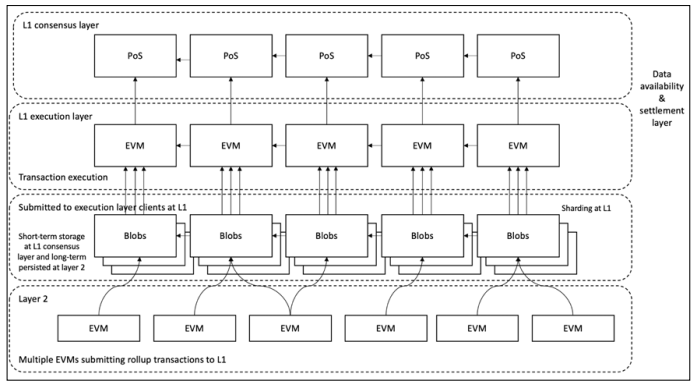
A principal vantagem aqui é que, em vez de usar *calldata* para escrever dados vindos dos *rollups*, esses *blobs* podem conter dados desses *rollups*, sendo uma opção muito mais barata em comparação ao *calldata*. Note que a EVM não tem acesso a esse *blob*; ela pode apenas ver o compromisso do *blob*. Essa abordagem resulta em uma melhoria de escalabilidade porque os *blobs* em si **não consomem gás** e os *rollups* podem usá-los para armazenar dados.

Então, em resumo, apenas alguns pontos-chave para lembrar:

* **Danksharding** foca primariamente na **disponibilidade de dados**, e não na execução de código.
* O objetivo final do danksharding é **reduzir significativamente o custo dos rollups** para camadas 2 e consolidar o Ethereum como uma camada de liquidação e disponibilidade de dados.
* O **proto-danksharding** é o primeiro passo em direção ao danksharding, implementando muitas das funcionalidades fundamentais.
* O **EIP-4488** reduz o custo do *calldata* de 16 unidades de gás para 3 unidades por byte e impõe um limite ao máximo de *calldata* por bloco.
* O **EIP-4844** implementa o proto-danksharding, que introduz um novo tipo de transação chamado **transação que carrega blob**, que serve a um propósito semelhante ao do *calldata*, ou seja, submeter lotes de *rollups* à camada 1 para disponibilidade de dados — **mas usando blobs de dados em vez de calldata**, como se fosse uma transação de grande porte, mas **sem afetar a camada de execução**, pois os blobs são armazenados nos nós beacon.

No futuro, o **danksharding completo será implementado**, tornando o Ethereum uma **camada de liquidação e disponibilidade de dados focada em descentralização e segurança**.

Podemos visualizar isso na Figura 13.13 abaixo:



*Figure 13.13: Danksharding – what Ethereum looks like afer full danksharding*

Conforme mostrado no diagrama acima, eventualmente o sharding será alcançado na camada 1, com as EVMs de camada 2 escrevendo blobs de dados na camada 1, enquanto a camada 1 do Ethereum torna-se a **camada base de liquidação e disponibilidade de dados**, composta pela camada de consenso e pela camada de execução.

**O futuro roteiro do Ethereum**

A atualização *hard fork* de Xangai (também chamada de **Shapella**, devido à combinação da atualização Capella na Beacon Chain e a atualização Shanghai na camada de execução) está programada para **abril de 2023** e é a atualização mais importante desde a Fusão. Ela inclui várias melhorias para reduzir os custos de gás no Ethereum.

Como sabemos, a Fusão ocorreu em setembro de 2022 e a rede Ethereum migrou de um mecanismo de consenso **PoW** para **PoS**. Com a troca para PoS, os usuários participam da validação ao apostar (*staking*) 32 ETH, em vez de usar hardware de mineração especializado. No entanto, após a Fusão com a Beacon Chain PoS, os usuários **não tinham opção de retirar os fundos apostados**. Isso muda com a atualização Shanghai (**EIP-4895** – <https://eips.ethereum.org/EIPS/eip-4895>), que adiciona a tão necessária funcionalidade de **retirada**.

É importante reconhecer que, eventualmente, algum grau de centralização pode ser necessário para alcançar alta escalabilidade. No entanto, a comunidade Ethereum está tentando **evitar isso o máximo possível**. Mesmo que alguma descentralização seja sacrificada, é crucial manter as propriedades de **validação sem confiança e resistência à censura** no blockchain. A implementação de conceitos como **PBS** e **statelessness fraca** facilita uma divisão entre os processos de construção e validação, permitindo escalabilidade enquanto preserva segurança e descentralização.

Os problemas atuais enfrentados pelo Ethereum são **congestionamento da rede e uso excessivo de espaço em disco**. Ambos indicam a necessidade urgente de aumentar a velocidade da rede e reduzir o uso de espaço em disco. Uma abordagem simples para resolver esses problemas seria aumentar a centralização; no entanto, a **descentralização é crítica**, pois oferece imparcialidade, resistência à censura, abertura, propriedade de dados e segurança robusta. O objetivo é ser mais escalável e seguro, mas também manter a descentralização — o que **não é fácil de alcançar** e tem sido chamado de **“trilema da escalabilidade”**. No entanto, o Ethereum parece estar no caminho certo, e suas atualizações visam resolver o trilema ao tornar o Ethereum mais escalável, seguro e descentralizado.

A escalabilidade será alcançada por meio do **sharding**, que aumentará a taxa de transferência de transações e também **reduzirá os requisitos de energia dos nós validadores**. Além disso, o staking reduz as barreiras à participação, o que resultará em uma rede mais descentralizada. Discutiremos o **trilema da escalabilidade** com mais detalhes no **Capítulo 17, Escalabilidade**.

Para que os *rollups* funcionem de maneira ideal, é necessário **armazenamento econômico na camada 1**. As atualizações pós-Fusão, especialmente o **danksharding**, fornecerão isso e criarão amplo espaço para armazenamento, permitindo que o Ethereum cresça ao **maximizar a eficiência dos rollups**, possibilitando **acelerações exponenciais**. Mais sobre *rollups* será abordado no Capítulo 17, “Escalabilidade”.

**Capítulo 13 – Resumo**

Neste capítulo, examinamos o Ethereum após a Fusão e seu roteiro. Exploramos vários aspectos de sua arquitetura e a visão geral por trás das atualizações do Ethereum para resolver o **trilema da escalabilidade**. Cobrimos a Beacon Chain, a Fusão e o **sharding**. Também abordamos conceitos relevantes como **danksharding**, **PBS** e o **protocolo de validador distribuído**, todos conduzindo a um futuro mais **escalável e seguro** para o Ethereum. No próximo capítulo, vamos introduzir o **Hyperledger**, que é um projeto de blockchain muito ativo e popular sobre o qual diversos outros projetos estão sendo construídos.