Visão Geral de Polkadot e Suas Considerações de Design

Jeff Burges[1], Alfonso Cevallos[1], Peter Czaban[1]
Rob Habermeier[2] , Syed Hosseini[1] , Fabio Lama[1]
Handan Kÿlÿn¸c Alper[1], Ximin Luo[1], Fatemeh Shirazi[1]
Alistair Stewart[1], Gavin Wood[1],[2]
[1] Fundação Web3,
[2] Parity Technologies

24 de Agosto de 2020

Resumo

Neste artigo, descrevemos os componentes de *design* do protocolo *multi-cadeia* heterogêneo Polkadot e explicar como esses componentes ajudam Polkadot resolver algumas lacunas existentes vindas das tecnologias *blockchain*. Atualmente, um grande número de projetos de *blockchain* foi introduzido e empregado com vários recursos que não são necessariamente projetados para funcionar um com o outro. Isto torna difícil para os usuários utilizar um grande número de aplicativos em diferentes projetos de *blockchain*. Além disso, com o aumento do número de projetos a segurança que cada um está fornecendo individualmente se torna mais fraca. A Polkadot tem o objetivo de prover uma estrutura interoperável para múltiplas cadeias com segurança compartilhada que é alcançada pelos componentes descritos neste artigo.

Índice

1 Introdução

A Internet foi originalmente projetada e construída sobre protocolos descentralizados, como o TCP/IP, no entanto, sua comercialização levou à centralização de todas as aplicações populares da *web* nos dias de hoje. Não nos referimos a qualquer centralização da infraestrutura física, mas sim à centralização lógica de poder e controle sobre a infraestrutura. Dois exemplos proeminentes são corporações gigantes como Google e Facebook: enquanto eles mantêm servidores em todo o mundo em um sistema fisicamente descentralizado *fashion*, estes são controlados, em última instância, por uma única entidade.

Uma entidade central que controla um sistema apresenta muitos riscos para todos. Por exemplo, eles podem parar o serviço a qualquer momento, pode vender os dados dos usuários a terceiros, e manipular como o serviço é trabalhando sem o acordo dos usuários. Isto é particularmente crítico para os usuários que confiam fortemente nestes serviços para fins comerciais ou privados.

Com todo o interesse atual relacionado à propriedade de dados pessoais, há uma necessidade crescente de uma melhor segurança, liberdade e controle para os usuários da rede, e com isso uma contra movimentação para mais aplicações descentralizadas onde nenhuma entidade controla o sistema por si só. Esta tendência para a descentralização não é nova; tem sido utilizada em diversas áreas do desenvolvimento da *web* e outros sistemas, tais como o movimento de *software* livre.

Blockchains são uma tecnologia recente proposta para resolver estes problemas, na tentativa de construir uma *web* descentralizada. Entretanto, ela só pode competir com a *web* centralizada se for utilizável por massas de usuários finais. Um aspecto importante disto é que aplicações separadas devem ser capazes de interagir, caso contrário, cada uma delas ficará isolada e não será adotada por tantos usuários. Ter que construir tal mecanismo de interoperabilidade introduz novos desafios, muitos dos quais estão faltando no mecanismo centralizado, devido às diferenças fundamentais no modelo de confiança entre os dois paradigmas. Por exemplo, Bitcoin [26] e Ethereum [10] são *blockchains proof-of-work* (*PoW*) onde a segurança depende de suposições sobre o poder de processamento; em vez disso, a segurança dos sistemas alternativos de *proof of stake* (*PoS*) dependem de incentivos e da capacidade de destruir depósitos de segurança. Estas diferenças apresentam dificuldades para que uma *blockchain* confie em outra. Outro desafio que as tecnologias de *blockchain* precisam enfrentar é a escalabilidade. Os sistemas de *blockchain* existentes geralmente têm alta latência e só podem ter dezenas de transações por segundo [14], enquanto empresas de cartão de crédito como a Mastercard ou Visa realizam milhares de transações por segundo [3].

Uma solução proeminente para a escalabilidade de *blockchains* é executar muitas cadeias em paralelo, muitas vezes chamado "*sharding*". Polkadot é um sistema de múltiplas cadeias que visa reunir o poder de segurança de todas estas cadeias juntas em um sistema de segurança compartilhado. Foi introduzida pela primeira vez em 2016 por Gavin Wood [33], e neste documento expandimos os detalhes.

Em resumo: Polkadot utiliza uma cadeia central chamada *relay chain* que se comunica com múltiplas cadeias heterogêneas e independentes denominadas *parachains* (*portmanteau of “parallel chains*”). A *relay chain* é responsável por fornecer segurança compartilhada para todas as *parachains*, assim como negociação entre *parachains* independentes de confiança de terceiros. Em outras palavras, as questões que Polkadot visa abordar são as discutidas acima: interoperabilidade, escalabilidade e segurança mais fraca decorrente da divisão do poder de segurança.

Organização do artigo: Na próxima seção, apresentamos uma sinopse da rede Polkadot, incluindo sua interface externa com *parachains* de clientes que expandiremos nas seções subsequentes. Nós revisamos informações preliminares, como descrição dos papéis dos participantes de Polkadot e do modelo adversário na Seção 3. Explicamos o quê os subprotocolos e componentes de Polkadot tentam alcançar na Seção 4 e, em seguida, continuamos a revisá-los em detalhes, incluindo criptografia de baixo nível e primitivos de rede. Finalmente, discutimos brevemente alguns trabalhos futuros na Seção 5. Nos apêndices, revisamos trabalhos relevantes, como uma comparação com outro sistema multi-cadeia A.3, uma breve descrição de um esquema de interoperabilidade para fazer a ponte para cadeias externas A.2, um esquema de execução seguro A.1 que usaremos para mensagens e um glossário com termos específicos de Polkadot na Tabela 1.

2 Sinopse

O objetivo desta seção é descrever a principal funcionalidade de Polkadot sem entrar em detalhes sobre o raciocínio e as considerações de projeto.

O sistema Polkadot consiste em uma única rede descentralizada colaborativa aberta chamada *relay chain*, que interage com muitas outras cadeias externas executadas em paralelo chamadas *parachains*. A partir de uma perspectiva de alto nível, *parachains* são clientes da *relay chain*, que fornece um serviço de segurança, incluindo comunicação segura. Esse é o único propósito da *relay chain*; as *parachains* são as entidades que fornecem funcionalidade em nível de aplicativo, tal como as criptomoedas.

Os detalhes internos das *parachains* não são uma preocupação da *relay chain*; as *parachains* precisam apenas aderir a interface que especificamos. Algumas dessas expectativas são componentes naturais das *blockchains*, daí a nomenclatura. No entanto, outros sistemas *non-blockchain* também podem ser executados como uma *parachain* de Polkadot desde que satisfaçam a interface. Isso é descrito abaixo: as partes relevantes estão sublinhadas.

Estes aspectos podem ser abreviados como, *Polkadot é um ecossistema escalável, heterogêneo e multi-cadeia.*

2.1 Modelo de segurança

Assumimos que as *parachains* estão sendo executadas como clientes externos não confiáveis da *relay chain* e que a *relay chain* trabalha apenas com *parachains* por meio de uma interface e não faz suposições sobre suas internas. Por exemplo, internamente elas podem ser autorizadas ou abertas; se alguns usuários internos subverterem a *parachain*, do ponto de vista de Polkadot a parachain inteira (como uma única entidade cliente) será maliciosa.

A *relay chain* de Polkadot foi projetada para operar com um nível de comportamento malicioso internamente, como um requisito de ser uma rede descentralizada aberta. Nós individuais específicos não são confiáveis, mas um subconjunto indeterminável de nós com limites inferiores em tamanho são confiáveis, e o protocolo funciona para garantir que a *relay chain* externa como um todo seja confiável. Consulte a seção 3.2 para obter detalhes.

2.2 Nós e funções

A rede *relay chain* de Polkadot consiste em nós e funções. Os nós são as entidades em nível de rede executando fisicamente o *software* Polkadot e as funções (Seção 3.1) são entidades de nível de protocolo realizando um determinado fim. Os nós podem desempenhar várias funções.

No nível da rede, a *relay chain* está aberta. Qualquer nó pode executar o *software* e participar como qualquer um destes tipos de nós:

01. Cliente leve - recupera determinados dados relevantes do usuário da rede. A disponibilidade de clientes leves é irrelevante - eles não prestam um serviço para os outros.

02. Nó completo - recupera todos os tipos de dados, armazena-os a longo prazo e os propaga para outros. Deve estar altamente disponível.

(a) *Sentry node* (Nó sentinela) - nós completos acessíveis publicamente que executam serviços de *proxy* confiáveis para um nó completo privado, executado pelo mesmo operador.

Às vezes nos referimos a um nó completo de uma *parachain*. No sentido abstrato para *parachains non-blockchain*, isso significa que eles participam dela em um grau suficiente para que possam verificar todos os dados que passam por ele.

Além de distribuir dados, os nós da *relay chain* podem executar certas funções de nível de protocolo listadas a seguir. Algumas dessas funções têm restrições e condições associadas a elas:

1. Validador - executa a maior parte do trabalho de segurança. Deve ser um nó completo da *relay chain*. Interage com coletores da *parachain*, mas não precisa participar de uma *parachain* como um nó completo.

2. Nomeador - parte interessada que apoia e seleciona candidatos a validadores (Seção 4.1). Isso pode ser feito a partir de um cliente leve, e eles não precisam ter nenhum conhecimento de *parachains*.

Parachains podem decidir sua própria estrutura de rede interna, mas espera-se que interaja com Polkadot através dos seguintes papéis:

1. Coletor[1] - coleta e envia os dados da *parachain* para a *relay chain*, sujeito às regras do protocolo descrita abaixo. Eles são escolhidos conforme definido pela *parachain*, e devem ser nós completos.

2. *Fishermen* (Pescadores) - realizam verificações de segurança adicionais sobre o correto funcionamento da *parachain*, em nome da *relay chain* que fornece uma recompensa. Esse papel é autoatribuído e recompensado, e deve ser um nó completo da *parachain*.

2.3 Protocolo

O protocolo *relay chain* de Polkadot, incluindo interação com as *parachains*, funciona da seguinte forma.

1. Para cada *parachain*:

(a) Os coletores observam o progresso dos protocolos de produção de blocos e de consenso, etapas (2) e (5) abaixo, respectivamente, exemplo: participando da *relay chain* como um nó completo. Com base no que eles acham que é o bloco mais recente da *relay chain* que provavelmente será finalizado, eles se baseiam no bloco de *parachain* recente (ou outros dados) que seriam finalizados por ele.

(b) Os coletores assinam a construção de dados em cima do último bloco de *parachain* e os enviam possivelmente indiretamente, para os validadores atribuídos à sua *parachain* (validadores de *parachain* para abreviar), para inclusão na *relay chain*. O ideal é que eles enviem uma única vez para ajudar no desempenho.

(c) Os validadores de *parachain* decidem qual bloco da *parachain* apoiar, e apresentam dados dele como o próximo *candidato* a ser adicionado ao próximo bloco da *relay chain*.

2. Um validador de produção de blocos coleta candidatos de todas as *parachains* e coloca essa coleção juntamente com quaisquer extrínsecos da *relay chain* recentes em um bloco principal da *relay chain*. (Seção 4.3.1). Para desempenho, isso não contém os dados completos de todas as *parachains*, mas apenas metadados e dados parciais, incluindo metadados relacionados à segurança. No caso desfavorável, isso pode resultar em *forks*, resolvidas posteriormente na etapa (5). Este subprotocolo é projetado para que, mesmo com *forks*, os participantes tenham uma ideia do bloco mais provável de ser finalizado, semelhante ao *Proof-of-Work.*

3. Um subprotocolo é executado para garantir que todos os dados estejam realmente disponíveis, incluindo e distribuindo para vários outros nós da *relay chain*. (Seção 4.4.2).

4. Os dados enviados de uma *parachain* podem incluir indicações de que estão enviando mensagens para outra cadeia, incluindo metadados para facilitar isso. Isso agora está incluído na relay *chain head*(s), para que as *parachains* do destinatário estejam cientes de quais novas mensagens foram enviadas para eles. E por sua vez, agora, recuperam os corpos das mensagens das *parachains* de envio. (Seção 4.4.3).

5. Os validadores enviam seus votos no bloco e o finalizam, resolvendo quaisquer *forks* para uma única *parachain head*. (Seção 4.3.2). Esses votos são adicionados aos blocos da *relay chain*.

O restante do artigo se expande sobre os papéis acima - funções a seguir na Seção 3 e subcomponentes de protocolo na Seção 4.

---

[1]Os validadores são, em certo sentido, um coletor da *relay chain*, na medida em que coletam extrínsecos (por exemplo, transações) dentro da rede da *relay chain*. No entanto, normalmente nos referimos a eles apenas como validadores, mesmo ao realizar essas tarefas, e o termo agrupador é reservado apenas para agrupadores de *parachain*.

3 Preparativos

Nesta seção descrevemos com mais detalhes as diferentes entidades envolvidas na execução de Polkadot, bem como nosso modelo de segurança dessas entidades. Isto forma o contexto de design para entender o projeto do protocolo descrito posteriormente, incluindo como ele funciona e por que foi projetado dessa maneira.

3.1 Funções

Os nós que executam a rede Polkadot assumem diferentes papéis e funções que apresentamos a seguir.

Validadores: Um validador é o mais alto no comando e ajuda a selar novos blocos na rede Polkadot. O papel do validador depende do depósito de um título suficientemente alto, embora permitimos que outras partes vinculadas indiquem um ou mais validadores para atuar por eles e, como tal, alguma parte do vínculo do validador pode não ser necessariamente de propriedade do próprio validador, mas sim por esses nomeadores. Um validador deve executar uma implementação de cliente de *relay chain* com alta disponibilidade e largura de banda. Em cada bloco o nó deve estar pronto para aceitar o papel de ratificar um novo bloco em alguma *parachain*, e pode ser necessário verificar mais alguns. Este processo envolve o recebimento, validação e republicação de blocos candidatos. A atribuição da *parachain* é aleatória e muda com frequência. Uma vez que não se pode esperar razoavelmente que o validador mantenha um banco de dados totalmente sincronizado de todas as *parachains*, a tarefa de elaborar uma sugestão de nova *parachain* será delegada a um terceiro, conhecido como coletor. Uma vez que todos os novos blocos de *parachain* foram devidamente ratificados por seus subgrupos de validadores designados, os validadores devem então ratificar o próprio bloco da *relay chain*. Isto envolve a atualização do estado das filas de transações (essencialmente movendo dados da fila de saída de uma *parachain* para a fila de entrada de outra *parachain*), processando as transações do conjunto de transações da *relay chain* configurado e ratificando o bloco final, incluindo as alterações finais da *parachain*. Um validador que reconhecidamente não estiver cumprindo seu papel sofrerá *slash*, ou seja, parte ou todos os seus títulos serão tomados. Em certo sentido, os validadores são semelhantes as *pools* de mineração das atuais *blockchains PoW.*

Nomeadores: Um nomeador é uma parte interessada que contribui para a segurança de um validador. Eles não têm nenhum papel adicional, exceto colocar capital de risco e, como tal, sinalizar que eles confiam em um determinado validador (ou conjunto deles) para agir com responsabilidade na manutenção da rede. Eles recebem um aumento ou redução proporcional em seu depósito de acordo com o crescimento do título para que contribuem. Juntamente com os coletores, em

seguida, os nomeadores são, em certo sentido, semelhantes aos mineradores das atuais redes *PoW.*

Coletores: Os coletores de transações (coletores para abreviar) são partes que auxiliam os validadores na produção de blocos válidos das *parachain.* Eles mantêm um “nó completo" para uma *parachain* específica; significando que eles retem todas as informações necessárias para poder criar novos blocos e executar transações da mesma forma que os produtores de blocos fazem nas *blockchains* atuais. Sob circunstâncias normais, eles irão agrupar e executar transações para criar um bloco não fechado e fornecê-lo, juntos com uma prova de validação, a um ou mais validadores atualmente responsáveis por propor um bloco de *parachain*.

Fishermen: Ao contrário das outras duas partes ativas, os *fishermen* não estão diretamente relacionados ao processo de criação de blocos. Em vez disso, eles são "caçadores de recompensas" independentes motivados por uma grande recompensa. Exatamente devido à existência de *fishermen*, esperamos que eventos de mau comportamento raramente aconteçam, e quando acontecem apenas devido ao descuido da parte vinculada com a segurança da chave secreta, e não por intenção maliciosa. O nome vem da frequência esperada de recompensa, os requisitos mínimos para participar e o eventual tamanho da recompensa. *Fishermen* recebem sua recompensa mediante oportuna comprovação de que pelo menos uma das partes vinculadas agiu ilegalmente. Isto será especialmente valioso para detectar a ratificação de blocos inválidos de *parachain*.

Os *fishermen* são um pouco semelhantes aos “nós completos" nos atuais sistemas *blockchain* que os recursos necessários são relativamente pequenos e o compromisso de tempo de atividade estável e largura de banda não é necessário. Os *fishermen* diferem na medida em que devem pagar uma pequena fiança. Este vínculo impede ataques *sybil* de desperdiçar tempo e recursos de computação dos validadores. Enquanto os *fishermen* fazem parte do modelo de segurança de Polkadot, o projeto seria seguro sem eles. Uma vez que não há modelo de incentivo projetado para *fishermen*, mas precisamos manter Polkadot segura em sua ausência. Adicionar um modelo de incentivo para os *fishermen* faz parte do nosso trabalho futuro.

Os elementos estruturais e os diferentes papéis definidos no protocolo de Polkadot são mostrados na Figura 1, em um exemplo com seis *parachains*, 18 validadores e 5 coletores por *parachain*. Figura 2 mostra a *relay chain* com 5 desses blocos de *relay chain*. Observe que o número de validadores de *parachain* atribuídos a uma *parachain* é dividido pelo número de *parachains*, no entanto, o número de coletores é individual para *parachains*. A ponte é um subprotocolo que permite que cadeias externas interoperem com Polkadot, veja A.2 para mais informações.

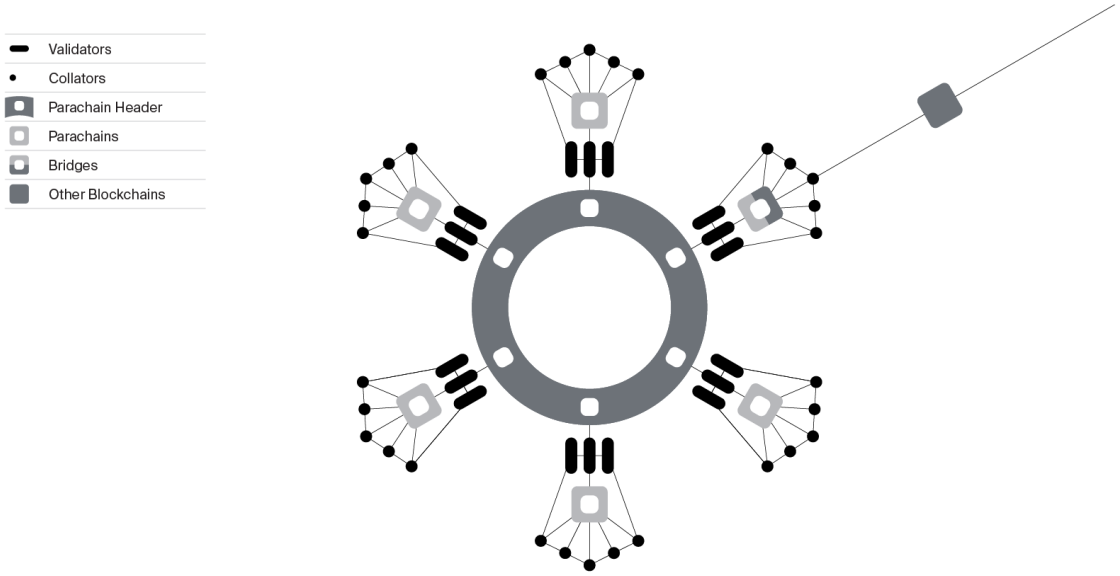

Figura 1: Esta figura mostra um bloco *relay chain* prendendo seis blocos de *parachain*. Cada *parachain* tem 5 coletores e 3 validadores atribuídos a ela (crédito da imagem: Ignasi Albero).

3.2 Modelo Adversarial de Polkadot

Funções: Em geral, assumimos que as partes honestas seguem o protocolo, enquanto as mal-intencionadas podem seguir qualquer algoritmo arbitrário. Assumimos que três quartos do *stake* dos nomeadores pertencem aos honestos. Como resultado desta suposição, mais de dois terços dos validadores que são eleitos por nomeadores, são honestos. Não temos nenhum limite para o número de *fishermen* maliciosos, já que seus comportamentos maliciosos são detectáveis e puníveis.

***Parachains*:** Não temos nenhuma suposição de segurança sobre o mecanismo de produção de blocos para *parachains*. Por outro lado, assumimos que uma quantidade significativa de coletores é honesta. A segurança de Polkadot não depende de nenhuma fração precisa e honesta de coletores, mas requer existência de alguns coletores honestos.

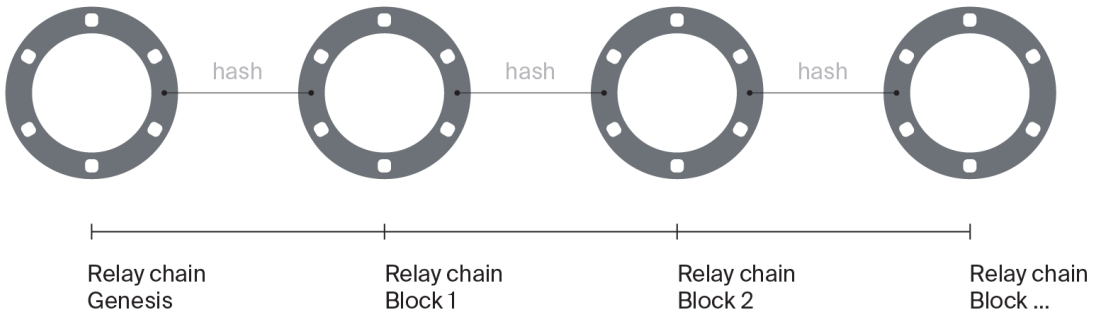

Figura 2: Esta figura mostra a *relay chain* com 5 blocos de *relay chain*. Para simplificar, nesta figura cada bloco de *relay chain* tem blocos de 6 *parachains*, entretanto, o número de blocos de *parachain* por bloco de *relay chain* podem variar (crédito da imagem: Ignasi Albero).

Partes do protocolo assumem que cada *parachain* tem pelo menos um membro honesto alcançável; quando isso não for viável ou não for realista, não seguimos essa suposição e, em vez disso, têm verificações adicionais contra uma adesão totalmente maliciosa.

Chaves: assumimos que partes mal-intencionadas geram suas chaves com um algoritmo arbitrário enquanto os honestos sempre geram suas chaves com segurança.

Rede e Comunicação: Todos os validadores têm seu próprio relógio local e seus relógios não dependem de nenhum relógio central. Assumimos que validadores e coletores estão em uma posição parcialmente síncrona na rede. Significa que uma mensagem enviada por um validador ou um coletor chega a todas as partes na rede na maioria $\triangle$ das unidades de tempo depois, onde $\triangle$ é um parâmetro desconhecido. Assim, assumimos uma eventual entrega de uma mensagem em Polkadot. Também assumimos que os coletores e os *fishermen* podem conectar-se à rede da *relay chain* para enviar seus relatórios.

4 Componentes e subprotocolos

Em seguida, resumimos brevemente a funcionalidade de Polkadot para obter uma visão geral e, continuamos a descrever os subprotocolos e componentes individuais.

Os validadores de Polkadot são selecionados pelo esquema *NPoS* (Seção 4.1). *Nominated Proof-of-Stake* ou *NPoS* é a nossa adaptação do *PoS* onde uma quantidade ilimitada de detentores de *tokens* podem participar como nomeadores, apoiando com sua participação um grande, mas limitado conjunto de validadores. Esse paradigma simultaneamente atinge altos níveis de segurança e escalabilidade, bem como um nível sem precedentes de descentralização garantindo uma propriedade conhecida na teoria do voto como representação proporcional justificada [29, 7]. Os nomeadores, que estão economicamente investidos na segurança do sistema, atuam como vigilantes sobre o desempenho dos validadores. Com base nas preferências expressas dos nomeadores sobre candidatos, a cada era o sistema seleciona um conjunto de validadores com suportes de participação que são tão altos e distribuídos o mais uniformemente possível. Os nomeadores também são economicamente desincentivados de concentrar seus votos em poucos validadores, o que ajuda a manter o sistema descentralizado ao longo do tempo. Além disso, o mecanismo de eleição é altamente adaptável a mudanças repentinas, como alguns validadores sendo expulsos após um *slashing*, pois redistribuem automaticamente os apoios dos nomeadores entre o novo conjunto de validadores, mesmo quando os próprios votos não mudam.

O objetivo de segurança de Polkadot é ser tolerante a falhas Bizantinas quando os participantes são racionais (consulte a Seção 4.5 para obter mais detalhes sobre incentivos e economia). Supomos que com as propriedades que *NPoS* dá, as partes interessadas elegem um conjunto de validadores que tem uma fração de mais de 2/3 de membros honestos.

Os validadores eleitos são responsáveis por executar a *relay chain* (Seção 4.2). Enquanto que os coletores de *parachain* são responsáveis por gerar blocos de *parachain* (Seção 4.4.1), os validadores são divididos em subconjuntos rotativos, um para cada cadeia, e precisam atestar a validade dos blocos *parachain* antes que os *heads* desses blocos sejam incluídos na *relay chain*.

Para obter uma boa escalabilidade, o número de validadores em cada um desses subconjuntos é pequeno. Apesar disso, graças à *NPoS* garante que todo validador é bem apoiado, a disponibilidade e validade (Seção 4.4.2) pode garantir que qualquer ataque à validade de Polkadot seja muito caro em expectativa. De fato, toda a segurança econômica de Polkadot apoia toda *parachain.* Isso está em contraste com ter, digamos, 100 *blockchains* independentes com uma soma total equivalente de participação, onde, em média, cada *blockchain* é apoiada por 1/100 do *stake* e, portanto, apenas se beneficia de 1/100 do nível de segurança. Garantimos a disponibilidade utilizando a codificação de eliminação de cada bloco de *parachain* para tornar os validadores coletiva e solidamente responsáveis pela disponibilidade desses blocos sem quebrar a escalabilidade.

Para que isso funcione, precisamos ser capazes de reverter a cadeia até sabermos com boa probabilidade que todas as *parachains* estão corretas. Isto significa que precisamos ser capazes de reorganizar a cadeia e para isso a cadeia precisa ser capaz de realizar um *fork*. Assim, utilizamos um mecanismo de produção de blocos, BABE (Seção 4.3.1), que, enquanto executado por validadores, tem propriedades semelhantes às cadeias *proof-of-work*. Especificamente, podemos usar a regra da cadeia mais longa como parte de nosso consenso e o próximo produtor de bloco não é conhecido de antemão. Por si só, o BABE exigiria que esperássemos muito tempo a partir do momento que um bloco é produzido até o momento em que é finalizado, ou seja, quando podemos ter certeza que com alta
probabilidade de o bloco nunca ser revertido. A lentidão é exigida em algumas circunstâncias para lidar com desafios de disponibilidade. Na maioria das vezes, no entanto, preferimos finalizar blocos muito mais rápido. Para isso, os validadores

analisam os blocos usando o GRANDPA (Seção 4.3), um dispositivo de finalidade que é claramente separado da produção de blocos. Esta separação o torna muito adaptável e aqui nos permite adiar a finalização dos blocos até que os desafios sejam resolvidos, sem desacelerar a produção de blocos. O GRANDPA obtém acordo bizantino sobre blocos finalizados e nos permitirá provar para uma entidade que acompanha o conjunto de validadores quais blocos são finalizados, quais serão importantes para pontes (Apêndice A.2).

Se uma conta em uma *parachain* envia *tokens* para outra *parachain*, então *XCMP* (Seção 4.4.3) garante que esta mensagem seja entregue corretamente. Ela é enviada a uma velocidade que não depende de quanto tempo demora para finalizar os blocos, o que significa que precisa lidar com a possibilidade de Polkadot realizar um *fork*. Assim, executamos de forma otimizada com base na suposição de que os blocos de *parachain* são corretos. Se não for, então precisamos reverter e para isso, é importante que apenas as *parachains* recebam as mensagens que foram enviadas por blocos gravados neste novo *fork* da *relay chain*, e não o *fork* revertido. Assim, precisamos que a *parachain* e a lógica *XCMP* garantam que um *fork* da *relay chain* defina uma história consistente de Polkadot e, portanto, as mensagens só chegam quando forem enviadas anteriormente no histórico definido por este *fork.*

Se a transferência de *token* for realizada em conjunto com os módulos SPREE (Apêndice A.1), então garante que, desde que as *parachains* sejam executadas corretamente, os *tokens* só podem ser criados e destruídos de uma forma acordada. Por sua vez, a execução correta do código das cadeias é garantida pelo esquema de disponibilidade e validade. O SPREE garante que o código compartilhado necessário para a lógica de transferência de *token* também está correto. Mesmo que as cadeias possam alterar seu próprio código, elas não podem alterar o código dos módulos SPREE. Em vez disso, o código dos módulos SPREE é armazenado centralmente e a execução desse código e seu armazenamento serão colocados em *sandboxs* do resto do estado de transição. Isso garante que essa mensagem de transferência de *token* seja interpretada corretamente e obtenha as garantias sobre os *tokens* que queremos.

No lado da economia (Seção 4.5), pretendemos ter uma taxa de inflação controlada quase constante. Como dito anteriormente, é importante para a segurança do sistema que todos os validadores tenham grandes montantes de *stake* apoiando-os. Nosso cronograma de recompensas adaptável para validadores e

nomeadores apoiá-los garante que a participação geral no *NPoS* permaneça alta e que os suportes de *stake* dos validadores sejam distribuídos uniformemente. Em um nível mais granular, pagar ou dar *slash* nos validadores por ação executada, e estendemos as mesmas recompensas ou penalidades aos nomeadores proporcionalmente, para garantir que a estratégia racional seja compatível com um comportamento honesto.

A própria lógica da *relay chain* precisará ser atualizada ocasionalmente. O mecanismo de governança (Seção 4.6) permite que os detentores de *tokens* Polkadot participem do processo de tomada de decisão em vez de ter quaisquer mudanças no sistema impostas por uma autoridade central - ou no caso de alguns sistemas descentralizados, por uma equipe de desenvolvedores. Muitas vezes, uma alteração contenciosa de código levou as *blockchains* para um impasse ou um *fork* permanente. Queremos um mecanismo que equilibre a capacidade de fazer mudanças incontestáveis rapidamente quando necessário, ao mesmo tempo em que fornece as ferramentas para lidar com propostas contenciosas de forma decisiva e justa. Os árbitros finais de Polkadot são os detentores de *tokens* DOT e, portanto, todas as decisões importantes, como alterações de código, são feitas por referendos ponderados pelo estado.

Há um conselho eleito, responsável por tomar decisões menores e definir parcialmente a prioridade dos referendos, de forma que não possam bloquear uma mudança que a maioria queira.

Finalmente, revisamos alguns primitivos que os subprotocolos de Polkadot estão usando, como as chaves criptográficas e esquema de rede na Seção 4.7 e na Seção 4.8, respectivamente. A rede Polkadot precisa estender a rede *gossip peer-to-peer* que é padrão em *blockchains de* cadeias únicas não permissionadas para um sistema multi-cadeia, onde qualquer tráfego de rede de nós não deve ser dimensionado com os dados totais do sistema.

4.1 ***Nominated proof-of-stake*** **e eleição de validador**

Polkadot terá um *token* nativo chamado DOT. Ele usará *Nominated Proof-of-Stake* (*NPoS),* nossa versão própria do *proof-of-stake* (*PoS*). Protocolos de consenso com natureza determinística, como o de Polkadot, requerem um conjunto de validadores

registrados de tamanho limitado. A Polkadot manterá um número n_{val} validadores, na ordem de centenas ou milhares. Este número será, em última instância, decidido pela governança, e pretende crescer linearmente com o número de *parachains*; ainda vai ser independente do número de usuários na rede, garantindo assim a escalabilidade. No entanto, *NPoS* permite um número ilimitado de titulares de DOT como nomeadores, que ajudam a manter altos níveis de segurança, colocando mais valor em *stake*. Como tal, *NPoS* não é apenas muito mais eficiente do que *proof-of-work* (*PoW*), mas também consideravelmente mais segura do que as formas convencionais de *PoS*, como *DPoS* e *BPoS*. Além disso, introduzimos novas garantias de descentralização até agora inigualável por qualquer outra *blockchain* baseada em *PoS*.

Um novo conjunto de validadores é eleito no início de cada era - um período de aproximadamente um dia (veja a Tabela 1 no Apêndice) - para servir naquela era, de acordo com as preferências dos nomeadores. Mais precisamente, qualquer titular de DOT pode optar por se tornar um candidato a validador ou um nomeador. Cada candidato indica o valor de *stake* que está disposto a participar e sua taxa de comissão desejada para custos operacionais. Por sua vez, cada nomeador bloqueia alguma participação e publica uma lista com qualquer número de candidatos em quem ela confia. Em seguida, um protocolo público, discutido abaixo, usa essas listas como entrada e elege os candidatos com mais apoio para servir como validadores para a próxima era.

Os nomeadores compartilham as recompensas, ou eventuais *slashings*, com os validadores que nomearam em uma base por DOT em *stake*; consulte a Seção 4.5 para obter mais detalhes. Os nomeados são, portanto, economicamente incentivados para atuar como cães de guarda para o sistema, e eles basearão suas preferências em parâmetros tais como os níveis de *staking* dos validadores, taxas de comissão, desempenho passado e práticas de segurança. Nosso esquema permite que o sistema eleja validadores com grandes quantidades de *stake* agregado - muito maior do que as participações de DOT de qualquer partido único - e, assim, ajuda a tornar o processo de eleição de validador em uma meritocracia em vez de uma plutocracia. Na verdade, a qualquer momento, esperamos que haja uma fração considerável de toda a oferta de DOTs esteja em *stake* em *NPoS*. Isso torna muito difícil para uma entidade adversária eleger validadores, pois precisa de uma grande

quantidade de DOTs ou alta reputação suficiente para obter o apoio dos nomeadores necessários, além de ser muito caro para atacar porque corre o risco de perder toda a sua participação e sua reputação conquistada.

A Polkadot elege validadores por meio de um protocolo descentralizado com critérios cuidadosamente selecionados, simples e conhecidos publicamente, tomando como entrada as listas de candidatos confiáveis dos nomeadores. Formalmente, o protocolo resolve um problema de eleição de vários vencedores com base em cédulas de aprovação, onde os nomeados têm poder de voto proporcional ao seu *stake*, e onde os objetivos são a descentralização e a segurança.

Descentralização: Nosso objetivo de descentralização se traduz na noção clássica de representação na teoria do voto. Ou seja, um comitê deve representar cada minoria no eleitorado, proporcional à sua força de voto agregada (neste caso, seu *stake*), sem a minoria estar subrepresentada. Destacamos aqui que os indicados e suas listas de candidatos confiáveis constituem um indicador valioso para as preferências da comunidade em geral, e que preferências diversas e facções surgirão naturalmente não apenas por razões econômicas e relacionadas à segurança, mas também política, geográfica, etc. Tal diversidade de pontos de vista é esperada e bem-vinda em uma comunidade descentralizada, e é importante envolver todas as minorias nos processos de tomada de decisão para garantir a satisfação do usuário.

O objetivo de projetar um sistema eleitoral que alcance a representação proporcional tem sido presente na literatura há muito tempo. De especial destaque é o trabalho dos matemáticos escandinavos Edvard Phragmén e Thorvald Thiele no final do século XIX. Muito recentemente, tem havido um esforço considerável na comunidade de pesquisa para formalizar a noção de proporcionalidade e revisitar os métodos de Phragmén e Thiele e otimizá-los algoritmicamente. Nosso protocolo de seleção de validador é uma adaptação dos métodos de Phragmén e é garantido observar a propriedade técnica da *representação proporcional justificada* (PJR) [29, 7]. Formalmente, isso significa que se cada nomeador $n \in N$ tem *stake* $stake_n$ e apoia um subconjunto $C_n \subseteq C$ de candidatos[1] o protocolo elegerá um conjunto $V \subseteq C$ de n_{val} de validadores tal que, se houver uma minoria $N' \subseteq N$ de nomeadores tais que

$$\left|\cap_{n\in N'} C_n\right| \geq t \text{ e } \frac{1}{t}\sum_{n\in N'} stake_n \geq \frac{1}{n_{val}}\sum_{n\in N} stake_n$$

para cerca $1 \leq t \leq n_{val}$, então $\left|V \cap \left(\cup_{n\in N'} C_n\right)\right| \geq t$. Em palavras, se uma minoria N tem pelo menos t candidatos comumente confiáveis, a quem poderia "obrigar" fornecer um apoio médio de pelo menos $\frac{1}{n_{val}}\sum_{n\in N} stake_n$ que por sua vez é um limite superior no suporte médio do validador em o conjunto eleito V), então esta minoria tem uma reivindicação justificada para ser representada em V por pelo menos t candidatos, embora não necessariamente de confiança comum.

Segurança: Se um nomeador conseguir que dois ou mais de seus candidatos confiáveis sejam eleitos como validadores, o protocolo também deve estabelecer como dividir sua participação e atribuir essas frações a eles. Por sua vez, essas atribuições definem o suporte total de participação que cada validador recebe. Nosso objetivo é tornar o suporte desses validadores o mais alto e equilibrado possível.

---

[1] Para facilitar a apresentação, consideramos aqui um modelo em que os candidatos não têm participação própria. O general caso pode ser reduzido a este modelo representando a participação de cada candidato como um nomeador adicional que exclusivamente nomeia esse candidato.

Em particular, nos concentramos em maximizar o apoio mínimo do validador. Intuitivamente, o suporte mínimo corresponde a um limite inferior no custo para um adversário obter controle sobre um validador, bem como um limite inferior vinculado ao valor reduzido por uma má conduta.

Formalmente, se cada nomeador $n \in N$ tem $stake_n$ e apoia um subconjunto candidato $C_n \subseteq C$, o protocolo deve não apenas eleger um conjunto $V \subseteq C$ de n_{val} de validadores com a propriedade *PJR*, mas também definir uma distribuição da participação de cada nomeador entre os validadores eleitos que ela apoia, ou seja, uma função $f: NxV \to R_{\geq 0}$ de modo

$$\sum_{v\in V\cap C_n} f(n,v) = stake_n \text{ para cada nomeador } n \in N,$$

e o objetivo é

$max\ (V, f)\ min\ (v \in V)\ \ support_{f(v)}$, onde $support_{f(v)} := \sum_{n \in N : v \in C_n} f(n, v)$.

O problema definido por este objetivo é chamado de *maximin support* na literatura [30], e é conhecido por ser *NP-hard*. Desenvolvemos para ele vários algoritmos eficientes que oferecem garantias (aproximações de fatores constantes) e também foram bem dimensionados e testados com sucesso em nossa rede de testes. Para mais informações, veja nosso artigo sobre a eleição do validador em *NPoS* [12].

4.2 Máquina de Estado da *Relay Chain*

Formalmente, Polkadot é uma máquina de estado fragmentada replicada onde *shards* são as *parachains* e a *relay chain* Polkadot faz parte do protocolo que garante o consenso global entre todas as *parachains*. Portanto, o protocolo da *relay chain* de Polkadot pode ser considerado como uma máquina de estado replicada por conta própria. Nesse sentido, esta seção descreve o protocolo da *relay chain,* especificando o estado de máquina que governa a *relay chain*. Para esse fim, descrevemos o estado da *relay chain* e os detalhes de transição de estado regida por transações agrupadas nos blocos da *relay chain*.

Estado: O estado é representado através do uso de uma estrutura de dados *de matriz associativa* composta por uma coleção de pares (*chave*; *valor*) onde cada chave é única. Não há hipótese de formato da chave ou o valor armazenado sob ela, além do fato de que tanto a chave quanto o valor precisam ser matrizes de *bytes* finitos.

Os pares (*chave*; *valor*) que compõem o estado da *relay chain* são organizados em uma árvore *Merkle radix-16*. A raiz desta árvore identifica canonicamente o estado atual da *relay chain*. A árvore *Merkle* também fornece um meio eficiente para produzir a prova de inclusão de um par individual no Estado.

Para manter o tamanho do estado sob controle, o estado da *relay chain* é usado apenas para facilitar as operações da *relay chain,* como *staking* e

identificação de validadores. A árvore *Merkle Radix* não deve armazenar qualquer informação sobre o funcionamento interno das *parachains*.

Transição de estado: Como qualquer sistema de transição baseado em transações, o estado de Polkadot muda por meio de um conjunto ordenado de instruções, conhecidas como extrínsecos. Estes extrínsecos incluem transações apresentados pelo público. Eles cobrem todos os dados fornecidos “de fora" do estado da máquina que pode afetar a transição de estado. A *relay chain* Polkadot é dividida em dois componentes principais, nomeadamente, o “*Runtime*" e o “*Host*". A lógica de execução da função de transição de estado é principalmente encapsulada no *Runtime* enquanto todas as outras operações genéricas, comumente compartilhadas entre modernas máquinas de estado replicadas baseadas em *blockchain*, são incorporadas ao *Host*. Em particular, este último é responsável pela comunicação em rede, produção de blocos e mecanismos de consenso.

As funções *Runtime* são compiladas em um módulo WebAssembly e são armazenadas como parte do Estado. O *Host* comunica os extrínsecos ao *Runtime* e interage com ele para executar o Estado de transição. Desta forma, a própria lógica de transição de estado pode ser atualizada como parte do estado transição.

Extrínsecos: Extrínsecos são os dados de entrada fornecidos à máquina de estado da *relay chain* Polkadot para transição para novos estados. Os extrínsecos precisam ser armazenados em blocos da *relay chain* para alcançar o consenso entre a réplica da máquina de estado. Os extrínsecos são divididos em duas grandes categorias, a saber: transações e "inerentes" que representam dados inerentes a um bloco de *relay chain*. O *timestamp t* de um bloco é um exemplo de extrínsecos inerentes que devem ser incluídos em cada bloco da *relay chain* Polkadot.

As transações são assinadas e divulgadas (*gossiped*) na rede entre os nós. Em contraste, inerentes não são assinados e não são divulgados individualmente, mas apenas quando são incluídos em um bloco. Os inerentes em um bloco são considerados válidos se uma supermaioria de validadores assume assim. As transações na *relay chain* estão principalmente relacionadas à operação da *relay chain* e o protocolo Polkadot como um todo, como `set code, transfer, bond, validate, nominate, vote.`

Os produtores de blocos da *relay chain* ouvem todas as mensagens da rede de transações. Ao receber uma mensagem de transação, as transações são validadas pelo *Runtime*. As transações válidas então são organizadas em uma fila com base em sua prioridade e dependência e são consideradas para inclusão em blocos futuros de acordo.

Formato de bloco da *relay chain*: Um bloco de *relay chain* típico consiste em um cabeçalho e um corpo. O corpo consiste simplesmente em uma lista de extrínsecos.

O cabeçalho contém o *hash* do bloco pai, número do bloco, a raiz da árvore de estado, a raiz da árvore de Merkle resultante da organização dos extrínsecos em tal árvore e do *digest*. O *digest* armazena informações auxiliares dos mecanismos de consenso que são necessários para validar o bloco e sua origem, bem como informações que ajudam os clientes leves (*light*) a validar o bloco sem ter acesso ao armazenamento do estado.

Construção de blocos da *relay chain*: nesta seção, apresentamos um resumo das várias etapas de operação da *relay chain* que são realizadas por seus validadores. A priori, cada validador conhece em particular os tempos durante os quais é suposto produzir um bloco (ver 4.3.1).

Enquanto isso, as transações vão desde o *hash* do bloco *parachain* validado, transferência, *staking*, nomeação ou *slashing* por violação de protocolo são submetidos aos validadores da *relay chain*. Os validadores examinam a validade das transações e as armazenam em sua *pool* de transações. Uma vez o intervalo de tempo durante o qual se espera que o validador produza o bloco chegou, o validador estima o bloco que mais provavelmente representa o estado que será finalizado pelo protocolo de finalidade e defini-lo como o estado atual da *relay chain*. Em seguida, o validador seleciona transações validas com a *pool* de transações, executa-as e atualiza o estado de acordo. O validador executa e coleta tantas transações quanto a capacidade do bloco permite e anexa um resumo criptográfico do estágio final da *relay chain* após a execução das transações selecionadas. Finalmente, o validador assina e publica o bloco construído.

Ao receber o novo bloco, outros validadores examinam a adesão do produtor ao protocolo bem como a validade das transações incluídas e armazenar o bloco na

árvore do bloco que representa todos os candidatos possíveis para uma transição de estado final da *relay chain*.

Simultaneamente, o conjunto de validadores vota em vários ramos da árvore de blocos (ver 4.3.2) e poda ramos que conflitam com a versão acordada pela maioria dos validadores. Dessa forma, eles eventualmente concordam com um estado canônico da *relay chain*.

4.3 Consenso

Nesta seção, explicamos o protocolo de consenso híbrido de Polkadot que é constituído em BABE: um mecanismo de produção de blocos da *relay chain* que fornece finalidade probabilística e GRANDPA que fornece uma realidade demonstrável e determinista e funciona independentemente do BABE. Informalmente a realidade probabilística implica que, após certo tempo, um bloco na *relay chain* será finalizado com probabilidade muito alta (próximo de 1) e finalidade determinística implica que um bloco finalizado permanece finalizado para sempre. Além disso, a realidade comprovável significa que podemos provar para as partes não envolvidas ativamente no consenso de que um bloco está finalizado.

Precisamos de uma realidade comprovável para facilitar as pontes para as cadeias fora de Polkadot; outra *blockchain*, não fazer parte do consenso de Polkadot poderia ser convencido de quando é seguro agir sobre os dados em uma *relay chain* ou bloco de *parachains* sem qualquer perigo de reversão. A melhor maneira de conseguir isso é ter um acordo Bizantino entre os *validadores* sobre o estado de Polkadot e suas *parachains*. Entretanto, o esquema de disponibilidade e validade 4.4.2 também pode exigir a reversão de blocos, o que significaria que obter um acordo Bizantino em cada bloco, como em Tendermint [8] ou Algorand [23], não seria adequado. No entanto, isto deve acontecer raramente, pois elevado número de *stake* sofrerá *slash* quando nós fazem isso. Como resultado, queremos um esquema que gere blocos e os execute de forma otimizada, mas pode levar algum tempo para finalizá-los. Assim, os eleitores do GRANDPA precisam esperar pelas garantias de disponibilidade e validade de um bloco antes de votar para finalizá-lo. Mesmo a velocidade com que nós finalizamos os blocos podem variar - se não recebermos relatórios de invalidade e indisponibilidade, então podemos concluir rapidamente,

mas se o fizermos, podemos ter que adiar a finalização enquanto executamos mais verificações envolvidas.

Devido a maneira como o protocolo de mensagens de Polkadot (*XCMP* 4.4.3) funciona, a velocidade de transmissão de mensagens é limitada pelo tempo do bloco, mas não pelo tempo de finalização. Assim, se atrasarmos o tempo, mas no final não reverter, então a passagem das mensagens ainda será rápida.

Como resultado desses requisitos, optamos por separar ao máximo os mecanismos de produção de blocos e de finalização de blocos. Nas próximas duas seções, descrevemos os protocolos BABE e GRANDPA que fazem isto, respectivamente.

4.3.1 Atribuição cega para extensão *Blockchain* (BABE)

Em Polkadot, produzimos blocos de *relay chain* usando nosso protocolo *Blind Assignment for Blockchain Extension* (BABE). O BABE atribui validadores aleatoriamente para bloquear *slots* de produção usando a aleatoriedade gerada com blocos. Um *slot* de produção de blocos é uma divisão de tempo em que um produtor de blocos pode produzir um bloco. Observe que esse tempo não é universalmente acordado, sobre o qual abordaremos mais tarde. Essas atribuições são completamente privadas até que os validadores atribuídos produzam seus blocos. Portanto, usamos “Atribuição Cega" no nome do protocolo. O BABE é semelhante ao Ouroboros Praos [15] com algumas diferenças significativas na regra de seleção da cadeia e suposições de tempo.

No BABE, podemos ter *slots* sem nenhuma atribuição que chamamos de *slots* vazios. Para preencher os *slots* vazios, temos um mecanismo de produção de bloco secundário baseado em Aura [31] que atribui validadores para *slots* publicamente. Observamos que estes blocos não contribuem para a segurança do BABE uma vez que a melhor seleção de cadeia e os algoritmos de geração de números aleatórios funcionam como se os blocos Aura não existissem. Portanto, a seguir descrevemos apenas o BABE juntamente com suas propriedades de segurança.

BABE [2] consiste em outra divisão de tempo chamada *épocas* (e_1, e_2, ...), onde cada época consiste de um número de *slots* de produção de blocos

sequenciais até o limite *R*. Cada validador sabe em quais *slots* ele deve produzir um bloco no início de cada época. Quando chega a hora de seu *slot*, o validador produz o bloco provando que ele está atribuído a este *slot.*

A atribuição cega é baseada no primitivo criptográfico chamado função aleatória verificável (VRF) [24] (ver Seção 4.7.2). Um validador em uma época e_m faz o seguinte para saber se é elegível para produzir um bloco no *slot* sl_i^m:

1. obtém a aleatoriedade no bloco gênesis se $m = 1 ou m = 2$. Caso contrário, obtém a aleatoriedade gerada duas épocas antes $(m - 2)$.

2. ele executa o VRF com sua chave secreta e a entrada: aleatoriedade e o número do *slot* sl_i^m.

Se a saída do VRF for menor que o limite τ, então o validador é o líder do *slot*, o que significa que é elegível para produzir um bloco para este *slot.* Selecionamos τ com respeito aos requisitos de segurança do BABE [2] por exemplo, maior τ torna menos provável selecionar apenas validadores honestos para um *slot* do que menor τ. Quando um validador produz um bloco, ele adiciona a saída do VRF e sua prova do bloco que mostra que sua saída VRF é menor τ para convencer outros validadores que tem o direito de produzir um bloco no *slot* correspondente. Os validadores sempre geram seus blocos no topo da melhor cadeia. A melhor regra de seleção de cadeia no BABE diz que ignore os blocos Aura e selecione a cadeia mais longa que inclui o último bloco GRANDPA finalizado. Consulte a seção 4.3.2 para os detalhes de como os blocos são finalizados no GRANDPA.

A aleatoriedade de uma época e_m em que $m > 2$ é gerada usando os blocos BABE da melhor cadeia que pertence a essa época: deixar p ser a concatenação de todos os valores VRF nos blocos de BABE que pertence a e_m. Então, calcule a aleatoriedade por época e_m como $r_m =$) onde H é uma função *hash*. Os validadores executam periodicamente o algoritmo de tempo relativo descrito abaixo para aprender a que horas um *slot* começa de acordo com seus relógios locais.

Protocolo de Tempo Relativo: Os validadores eleitos para um *slot* precisam saber quando é a hora certa para produzir um bloco para a consistência e segurança do BABE. Para isso, os validadores utilizam seus relógios do computador local que não é ajustado por nenhum protocolo de ajuste de relógio centralizado, como o Protocolo de Tempo de Rede (*Network Time Protocol)* [25]. Em vez disso, eles mantêm seu relógio sincronizado com os outros validadores com o protocolo de tempo relativo. O modelo de segurança formal da sincronização do relógio local em *blockchains* sem NTP e mais detalhes sobre o protocolo de tempo relativo podem ser encontrados em [5].

No BABE, assumimos que, após o lançamento do bloco gênesis, os validadores eleitos da primeira época armazenam a hora de chegada do bloco gênesis em relação ao seu relógio local. Então, eles marcam a hora de início do primeiro *slot* e incrementam o número do *slot* a cada *T* segundos. Após este ponto, eles executam periodicamente o algoritmo relativo para não perder a sincronização com os outros por causa de seus relógios locais oscilando. Além disso, um validador que se junta após o bloco de gênesis, executa o algoritmo de tempo relativo a ser sincronizado com os demais validadores.

Em todas as épocas de sincronização (diferentes das épocas no BABE), os validadores atualizam seus relógios de acordo com o resultado do protocolo de tempo relativo e usar o novo relógio até a próxima época de sincronização. A primeira época de sincronização ε_1 inicia logo após o bloco gênesis ser liberado. As outras épocas de sincronização ε_1 iniciam quando o número do *slot* do último bloco (probabilisticamente) finalizado é sl_{ε}, que é o menor número de *slot* tal que $\bar{sl}_{\varepsilon} - \bar{sl}_{\varepsilon-1} \geq s_{cd}$ onde $\bar{sl}_{\varepsilon-1}$é o número do *slot* do último (probabilisticamente) bloco finalizado na época de sincronização $\varepsilon - 1$. Aqui, s_{cd} é o parâmetro da propriedade de densidade da cadeia (CD) que será definido de acordo com o crescimento da cadeia. Mais detalhadamente, cada validador armazena o tempo de chegada t_j de blocos junto com o número do *slot* sl'_jno bloco durante uma época de sincronização. No fim de uma época de sincronização, o validador recupera o tempo de chegada dos blocos probabilisticamente finalizados gerados durante a época de sincronização e calcula os horários de início de alguns candidatos do primeiro *slot* sl

da próxima época de sincronização, por exemplo, dado que $a_j = T\left(sl - sl'_j\right), CT = \{t_j + a_j\}$. Os tempos em CT são considerados como candidatos. Para escolher um candidato, o validador ordena a lista de candidatos CT e gera a mediana da lista ordenada como uma hora de início do sl. Um exemplo de execução do protocolo de tempo relativo na primeira época de sincronização está na Figura 3.

Visão geral de segurança do BABE: Garay et al. [17] define as propriedades abaixo para obter um protocolo *blockchain* seguro. Informalmente, podemos descrever essas propriedades da seguinte forma:

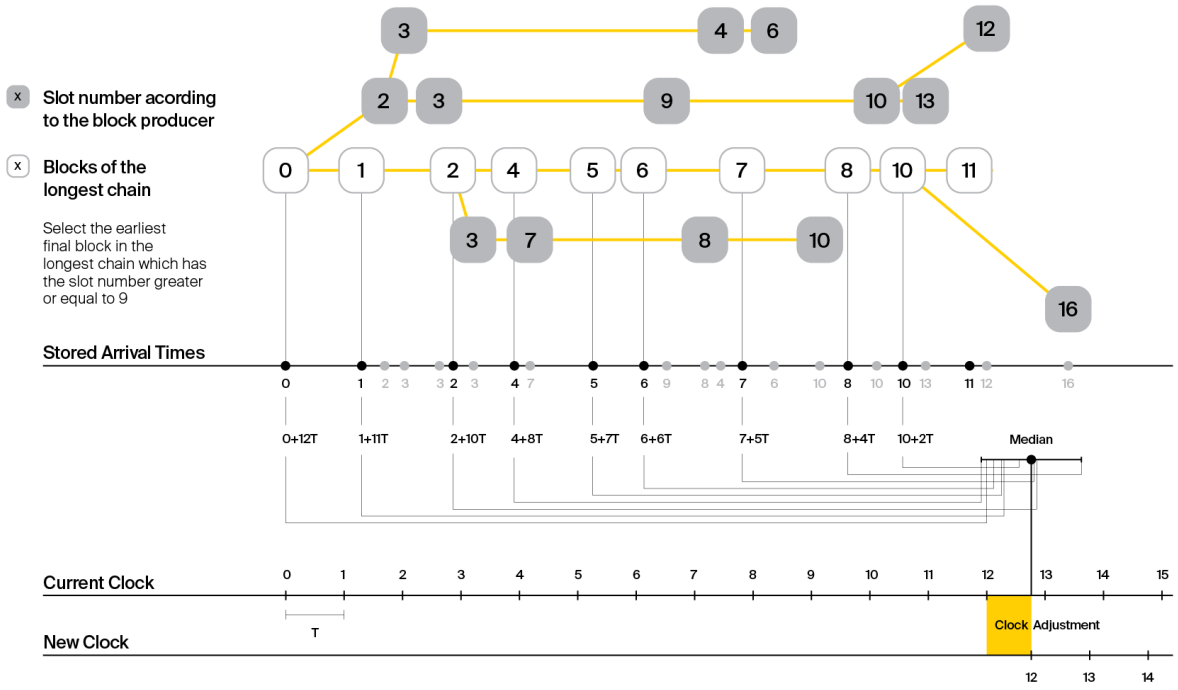

Figura 3: Um exemplo de execução do protocolo de tempo relativo na primeira época em que s_{cd}= 9 (crédito da imagem: Ignasi Albero).

- *Prefixo Comum (Common Prefix (CP):* Garante que os blocos que são $k - blocks$ antes do último bloco de uma *blockchain* de um validador honesto não podem ser alterados. Chamamos todos os blocos imutáveis de blocos finalizados. O BABE satisfaz a propriedade CP graças a super maioria honesta, desde que os validadores maliciosos selecionados para um *slot* probabilisticamente sejam muito menos do que os validadores honestos. Isto

significa que validadores maliciosos não têm o suficiente para construir outra cadeia que não inclua um dos blocos finalizados.

- *Qualidade da Cadeia* (*Chain Quality (CQ):* Assegura uma contribuição mínima de bloco honestos para quaisquer melhores donos de cadeia por uma parte honesta em cada certo número de *slots*. Garantimos mesmo no pior caso em que um atraso de rede será o máximo, haverá pelo menos um bloco honesto na melhor cadeia durante uma época, de modo que a aleatoriedade não pode ser tendenciosa.

- *Crescimento da Cadeia (Chain Growth (CG):* Garante um crescimento mínimo entre *slots*. Graças a super maioria dos validadores honestos, validadores maliciosos não podem impedir o crescimento das melhores cadeias.

- *Densidade da Cadeia (Chain Density (CD):* Garante que em uma porção suficientemente longa da melhor cadeia, mais da metade dos blocos produzidos por validadores honestos. As propriedades CQ e CG implicam nesta propriedade [15].

Mais detalhes sobre o BABE e sua análise de segurança podem ser encontrados em [2].

4.3.2 GRANDPA

Como mencionado acima, queremos um mecanismo de finalização que seja flexível e separado da produção de bloco, o que é alcançado pelo GRANDPA. A única modificação necessária no BABE para ele trabalhar com o GRANDPA é mudar a regra da escolha do *fork*: em vez de construir sobre a cadeia mais longa, um validador que produz um bloco deve construir sobre a cadeia mais longa, incluindo todos os blocos que ele vê como finalizado. O GRANDPA pode trabalhar com muitos mecanismos diferentes de produção de blocos e será possível trocar o BABE por outro.

Intuitivamente GRANDPA é um protocolo de acordo Bizantino que trabalha para chegar a um acordo sobre uma cadeia, de muitos possíveis *forks*, seguindo uma regra mais simples de escolha de *forks*, que juntamente com a mecanismo de produção em bloco daria uma finalidade probabilística se o próprio GRANDPA parasse de finalizar blocos. Queremos ser capazes de aceitar muitos blocos ao mesmo tempo, ao contrário do que acontece com os protocolos de acordo Bizantinos *single-block*.

Supomos que podemos pedir a regra da escolha do *fork* para o melhor bloco, dado um determinado bloco. A ideia básica é que queremos chegar a um acordo Bizantino sobre o prefixo da cadeia que todos aceitem. Para tornar isto mais robusto, tentamos concordar com o prefixo da cadeia onde 2/3 dos validadores concordam.

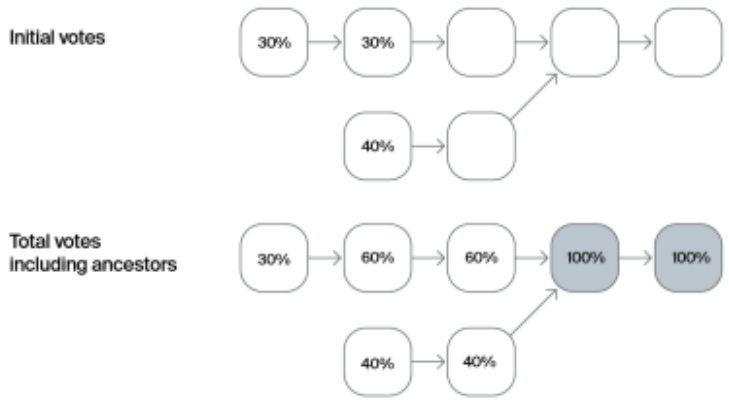

Figura 4: Votos do GRANDPA e como são agregados (crédito da imagem: Ignasi Albero).

Usamos o protocolo GHOST (*Greedy Heaviest Observed Subtree*) na regra de votos, bem como Casper TFG [27] ou algumas das regras de escolha de *fork* sugeridas para uso com Casper FFG [11]. Usamos esta regra dentro do que está estruturado como um protocolo de acordo Bizantino mais tradicional, para processar votos. A regra GHOST dos 2/3 (ilustrada na Figura 4) funciona da seguinte maneira. Temos um conjunto de votos, dados por *hashes* de blocos nos quais os validadores honestos não devem ter mais de um voto, e nós pegamos a ponta da cadeia formada indutivamente como se segue. Começamos com o bloco gênesis e depois incluímos o filho daquele bloco que 2/3 dos eleitores votaram em descendentes, desde que haja exatamente uma criança assim. O líder desta cadeia é $g(V)$ onde V é o conjunto de votos. Por exemplo, na Figura 4, o lado esquerdo dá os votos para

blocos individuais e o lado direito o total de votos para cada bloco e todos os seus descendentes. O bloco gênesis está no topo e levamos seu filho com 100% > 2/3 dos votos. Os filhos desse bloco têm 60% e 40% dos votos respectivamente e como estes estão abaixo de 2/3 paramos e retornamos ao segundo bloco.

Há duas fases de votação em uma rodada do GRANDPA: pré-votação e pré-comprometimento. Em primeiro lugar, os validadores votam na melhor cadeia. Então eles aplicam a regra $\frac{2}{3}$-GHOST, g, ao conjunto de pré-votos V eles veem e se comprometem com $g(V)$.Então, da mesma forma, eles pegam o conjunto de pré-compromissos C que veem e finalizam $g(C)$.

Para garantir a segurança, garantimos que todos os votos sejam descendentes de qualquer bloco que possa ter finalizado na última rodada. Os nós mantêm uma estimativa do último bloco que poderia ter sido finalizado em uma rodada, que é calculada a partir dos pré-votos e pré-comprometimentos. Antes de iniciar uma nova rodada, um nó espera até ver pré-compromissos suficientes para ter certeza de que nenhum bloco em uma cadeia ou posteriormente na mesma cadeia que a estimativa desta rodada pode ser finalizado. Em seguida, ele garante que apenas pré-vota e pré-compromete na próxima rodada para blocos que são descendentes da última rodada, estimar que continua atualizando e ouvindo os pré-compromissos da última rodada. Isso garante segurança

Para garantir a vivacidade, selecionamos um validador em rotação para ser o principal. Eles começam a rodada transmitindo sua estimativa para a última rodada. Então, quando os validadores votam a favor, se o principal bloco passa duas verificações, que é pelo menos a estimativa do validador e que obteve $\frac{2}{3}$do pré-votos para ele e seus descendentes na última rodada, então ele prefere a melhor cadeia incluindo os blocos primários. A ideia aqui é que se o bloco primário não foi finalizado, então o progresso é feito finalizando o bloco. Se o bloco primário não tiver sido finalizado e todos os validadores concordarem com a melhor cadeia incluindo o último bloco finalizado, o que devemos fazer eventualmente porque BABE dá finalidade probabilística por si só, então agora fazemos progresso finalizando essa cadeia.

4.4 *Parachains*

Nesta seção, revisamos a produção de blocos das *parachains*, sua disponibilidade e esquema de validade e seu esquema de mensagens.

4.4.1 Produção de Blocos

Discutiremos a produção de blocos para uma *parachain* generalista. Ao final da seção, discutiremos alternativas.

Em linhas gerais, um coletor produz um bloco da *parachain*, envia-o aos validadores da *parachain,* que assinam seu cabeçalho como válido e o cabeçalho com assinaturas suficientes é colocado na *relay chain*. Neste ponto, o bloco da *parachain* é tão canônico quanto o bloco da *relay chain* em que seu cabeçalho apareceu. O bloco da *relay chain* está na melhor cadeia de acordo com o BABE (consulte a Seção 4.3.1), assim é o bloco *parachain* quando este bloco da *relay chain* é finalizado, o mesmo acontece com o bloco da *parachain.*

Como os validadores de *parachain* trocam de *parachains* com frequência, eles são clientes sem estado da *parachain.* Assim, distinguimos entre o bloco da *parachain* B, que normalmente é suficiente para nós completos da *parachain*, como coletores, para atualizar o estado da *parachain* e o bloco *Proof of Validity* (PoV) $B_{Po}V$, que um validador que não possui o estado da *parachain* pode verificar.

Qualquer validador deve ser capaz de verificar $B_{Po}V$ dado o estado da *relay chain* usando a função de validação de transição de estado (STVF) da *parachain*, o código *WebAssembly* para o qual é armazenado na *relay chain* de maneira semelhante ao tempo de execução da *relay chain*. O STVF recebe como entrada o bloco *PoV*, o cabeçalho do último bloco da *parachain* desta *parachain* e uma pequena quantidade de dados do estado da *relay chain*.

O STVF emite a validade do bloco, o cabeçalho deste bloco e suas mensagens de saída. O bloco *PoV* contém todas as mensagens de saída e o bloco da *parachain* B. Os validadores da *parachain* devem conversar com o bloco da *parachain* para a rede *parachain,* como um *backup* para o próprio coletor fazer isso.

O bloco *PoV* será o bloco *parachain,* suas mensagens de saída, seu cabeçalho e cliente leve testemunhas. Essas testemunhas são provas de Merkle que

fornecem todos os elementos do estado de entrada e saída, que são usados ou modificados pela transição de estado das raízes do estado de entrada e saída.

Para ajudar na resistência à censura, uma *parachain* pode querer usar prova de trabalho ou prova de participação para selecionar coletores, onde a estratégia de seleção depende da *parachain* dada. Isso pode ser implementado no STVF e não precisa fazer parte do protocolo Polkadot. Então, para prova de trabalho, o STVF verifique se o *hash* do bloco é suficientemente pequeno. No entanto, para a velocidade, seria útil garantir que a maioria dos blocos da *relay chain* possam incluir um bloco da *parachain*. Para *PoW*, isso exigiria sendo provável que vários coletores possam produzir um bloco. Como tal, ainda precisaremos de um desempate para os validadores de *parachain* coordenarem a validação do mesmo bloco de *parachain* primeiro. Este pode ser o esquema do bilhete de ouro [33]. Para prova de participação, isso pode não ser necessário.

Opcionalmente, para algumas *parachains*, o bloco de *parachains* B pode não ser suficiente para os coletores atualizarem seu estado. Isso pode acontecer para cadeias que usam provas sucintas de conhecimento zero para atualizar seu estado, ou mesmo para redes autorizadas que apenas dão assinaturas de autoridades para a validação. Tais cadeias podem ter outra noção de bloco de *parachain* que é realmente necessária para atualizar seu estado e devem ter seu próprio esquema para garantir a disponibilidade desses dados.

4.4.2 Validade e Disponibilidade

Uma vez que um bloco de *parachain* é criado, é importante que a bolha (*blob)* de *parachains* que consiste no bloco *PoV* e no conjunto de mensagens de saída da *parachain* esteja disponível por um tempo. A solução ingênua para isso seria transmitir/conversar com as bolhas de *parachain* para todos os nós da *relay chain*, o que não é uma opção viável porque existem muitas *parachains* e os blocos PoV podem ser grandes. Nós queremos encontrar uma solução eficiente para garantir que os blocos *PoV* de quaisquer blocos de *parachain* criados recentemente sejam acessíveis.

Para uma única cadeia, como Bitcoin, desde que 51% do poder de *hash* seja honesto, não gerando dados de bloco disponível garante que nenhum minerador

honesto se baseie nele para que ele não fique na cadeia final. No entanto, o consenso da *parachain* em Polkadot é determinado pelo consenso da *relay chain*. Um bloco de *parachain* é canônico quando seu cabeçalho está na *relay chain*. Não temos garantias de que qualquer pessoa além dos coletores e validadores de *parachain* viram o bloco *PoV*. Se estes conspirarem, então o resto da rede *parachain* não precisa ter o bloco *parachain* e a maioria dos coletores não pode construir um novo bloco e a invalidade deste bloco pode não ser descoberta. Gostaríamos que os participantes do consenso, aqui os validadores, para garantir coletivamente a disponibilidade em vez de depender de alguns nós.

Para este fim, projetamos um esquema de disponibilidade que utiliza *erasure coding* (ver exemplo, [4]) para distribuir o bloco *PoV* para todos os validadores. Quando qualquer mau comportamento, particularmente em relação à invalidade, é detectado, o *blob* pode ser reconstruído a partir das peças codificadas de apagamento distribuídas.

Se um bloco estiver disponível, os nós completos da *parachain* e qualquer cliente leve que tenha o bloco *PoV*, pode verificar sua validade. Temos três níveis de verificações de validade em Polkadot. A primeira validade a verificação de um bloco *PoV* é executada pelos validadores de *parachain* correspondentes. Se eles verificarem o bloco *PoV*, então eles assinam e distribuem os códigos de apagamento do *blob*, incluindo o bloco *PoV,* para cada validador. Contamos com nós que atuam como *fishermen* para relatar a invalidade de um *blob* como um segundo nível de verificação de validade. Eles precisariam respaldar qualquer reivindicação com sua própria participação em DOTs. Nós assumiríamos que a maioria dos coletores serão *fishermen*, pois eles têm interesse na validade contínua da cadeia e já estão executando nós completos, então tudo o que eles precisam é fazer *stake* de DOTs. O terceiro nível de verificação de validade é executado por alguns validadores designados de forma aleatória e privada. Nós determinamos o número de validadores no terceiro nível de verificação de validade considerando a quantidade de relatórios de invalidez dados pelos *fishermen* e relatórios de indisponibilidade dados por coletores. Se um bloco inválido de *parachain* é detectado, os validadores que assinaram para sua validade são cortados. Esperamos o suficiente esses controladores designados aleatoriamente para verificar o bloco antes de votar nele no GRANDPA. Nós também queremos garantir que o bloco esteja disponível antes de selecionar os validadores atribuídos aleatoriamente. Isto significa que os

validadores de *parachain* têm que se comprometer a correr um alto risco de serem cortados por uma pequena probabilidade de obter um bloco inválido finalizado. Isso significa que o custo esperado de obter um bloco inválido em Polkadot é maior do que a quantidade de participação que apoia uma única *parachain.*

A segurança da nossa disponibilidade e esquema de validade é baseado na segurança do *gadget* de finalidade GRANDPA (consulte a Seção 4.3.2) e a qualidade da aleatoriedade gerada em cada época no BABE (consulte a Seção 4.3.1). Consulte [1] para obter mais detalhes sobre o esquema de disponibilidade e validade.

4.4.3 Passagem de Mensagens em Cadeia Cruzada (XCMP)

XCMP é o protocolo que as *parachains* usam para enviar mensagens entre si. Tem como objetivo garantir quatro coisas: primeiro, que as mensagens cheguem rapidamente; segundo, que as mensagens de uma *parachain* cheguem em ordem em outra; terceiro, que as mensagens recebidas foram de fato enviadas no histórico finalizado da cadeia de envio; e quarto, que os destinatários receberão mensagens de forma justa entre os remetentes, ajudando a garantir que os remetentes nunca esperem indefinidamente para que suas mensagens sejam vistas.

Existem duas partes no XCMP. (1) Metadados sobre mensagens de saída para um bloco *parachain* são incluídos na *relay chain* e, posteriormente, esses metadados são usados para autenticar mensagens pela *parachain* receptora. (2) Os corpos das mensagens correspondentes a esses metadados precisam ser realmente distribuídos pelos remetentes aos destinatários, juntamente com uma prova de que o corpo da mensagem está realmente associado a um relevante metadado. Os detalhes da distribuição são cobertos como um protocolo de rede em mensagens de cadeia cruzada; o restante é coberto abaixo.

A maneira como os blocos da *relay chain* incluem cabeçalhos de blocos de *parachain* dá uma noção síncrona de tempo para blocos de *parachain,* apenas por números de bloco de *relay chain*. Além disso, permite-nos autenticar mensagens como sendo enviadas no histórico fornecido pela *relay chain*, ou seja, é impossível que uma *parachain* envia uma mensagem, então reorganiza² para que essa mensagem não tenha sido enviada, mas tenha recebido. Isso é válido mesmo que o

sistema não tenha chegado ao fim sobre se a mensagem foi enviada, porque qualquer *relay chain* fornece um histórico consistente.

Como exigimos que as *parachains* atuem em todas as mensagens eventualmente, a não entrega de uma única mensagem pode potencialmente impedir que uma *parachain* seja capaz de construir blocos. Consequentemente precisamos de redundância suficiente em nosso sistema de entrega de mensagens. Qualquer validador que valida o bloco *PoV* deve manter disponível quaisquer mensagens enviadas desse bloco por um ou mais dias e todos os nós completos de parachain de envio também armazena as mensagens enviadas até que eles saibam que foram executadas.

Para obter consistência, quando uma *parachain* de origem *S* envia mensagens em um bloco *B* de *parachain* para uma *parachain* de destino *D*, precisaremos autenticá-las usando o estado da *relay chain*, que é atualizado com base no cabeçalho da *parachain PH* correspondente ao bloco da *parachain B* que foi incluído na *relay chain*. Precisamos limitar a quantidade de dados em cabeçalhos como *PH*, no estado da *relay chain* e também para limitar o que a *relay chain* precisa fazer para autenticação ao processar tais cabeçalhos de *parachain.*

Para este fim, o cabeçalho da *parachain PH* contém uma mensagem raiz *M* de mensagens de saída, bem como um campo de *bits* indicando para quais outras *parachains* foram enviadas mensagens neste bloco. A mensagem raiz *M* é a raiz de uma árvore de Merkle do *hash* principal *Hp* de uma cadeia de *hash* para cada *parachain p* que este bloco envia mensagens para. A cadeia de *hash* com cabeça *HD* tem o *hash* de todas as mensagens enviadas para *S* para *D*, não apenas no bloco *B*, mas sempre enviado de *S* para *D* em qualquer bloco até o bloco *B*. Isso permite que muitas mensagens de *S* para *D* sejam verificadas de uma vez de *M*.

---

[2] reorganização da cadeia

No entanto, as próprias mensagens em si são transmitidas, elas também devem ser enviadas com a prova Merkle que permite que os nós da *parachain* receptora autentique que foram enviadas por um *B* cujo cabeçalho *PH* estava em um bloco de *relay chain* específico.

As *parachains* recebem as mensagens recebidas em ordem. Internamente, as *parachains* podem adiar ou reordenar agindo em mensagens de acordo com sua

própria lógica (possivelmente restringida pelo SPREE, veja A.1). Entretanto, eles devem receber mensagens na ordem determinada pelo histórico consistente fornecido pelo *relay chain*. Uma *parachain D* sempre recebe mensagens enviadas por blocos de *parachain* cujo cabeçalho estava nos primeiros blocos da *relay chain*. Quando várias dessas *parachains* de origem têm um cabeçalho no bloco da *relay chain*, as mensagens dessas *parachains* são recebidas em alguma ordem predeterminada das *parachains*, ou sequencialmente em ordem crescente de identificação de *parachain*, ou alguma versão embaralhada disso.

Uma *parachain D* recebe todas as mensagens enviadas por uma *parachain S* em um bloco de *parachain* ou nenhum deles. Um cabeçalho de *parachain PH'* de *D* contém uma marca d'água. Esta marca d'água consiste em um número de bloco de *relay chain R* e uma identificação de *parachain* de uma *parachain* de origem *S*. Isso indica que *D* recebeu todas as mensagens enviadas por todas as cadeias antes do bloco *R* da *relay chain* e agiu nas mensagens enviadas no bloco *R* de *parachains* até e *S* incluindo no ordenado.

A marca d'água deve avançar em pelo menos uma *parachain* de envio em cada um dos blocos de *parachain* de *D's*, o que significa que o número do bloco da *relay chain* da marca d'água avança ou permanece o mesmo e apenas avançamos a *parachain.* Para produzir um bloco de *parachain* na *parachain D* que se baseia em um bloco de *relay chain* específico *R*, um coletor precisaria observar quais cabeçalhos de *parachain* foram construídos entre o bloco de *relay chain* em que o último bloco de *parachain* dessa cadeia foi construído. Além disso, precisa dos dados de mensagem correspondentes para cada um daqueles que indicaram que enviaram mensagens para *D*. Assim, pode construir um bloco *PoV* para que o STVF possa validar que todas essas mensagens foram tomadas medidas.

Como uma *parachain* deve aceitar todas as mensagens que são enviadas a ela, implementamos um método para *parachains* tornar ilegal para outra *parachain* enviar qualquer mensagem que possa ser usada no caso de ocorrência de *spam*. Quando o cabeçalho *parachain* de um bloco *parachain* que envia uma mensagem é incluída em um bloco de *relay chain*, então quaisquer nós conectados à fonte e as redes *parachain* de destino devem encaminhar mensagens, juntamente com suas provas, do remetente ao receptor. A *relay chain* deve, pelo menos, atuar como um *backup*: os validadores da *parachain* receptora de *D* estão conectados à rede *parachain* de *D* e se não receberem mensagens nela, então elas podem solicitá-los

aos validadores de *parachain* da cadeia de envio S no momento em que a mensagem foi enviada.

4.5 Economia e Camada de Incentivo

Polkadot terá um *token* nativo chamado DOT. Suas várias funções são descritas nesta seção.

4.5.1 Recompensas de *staking* e inflação

Começamos com uma descrição das recompensas de *staking*, ou seja, pagamentos a *stakers* – validadores e nomeadores – provenientes da cunhagem de novos DOTs. Ao contrário de alguns outros protocolos *blockchain*, a quantidade de *tokens* em Polkadot não será limitado por uma constante absoluta, mas sim taxa de inflação anual controlada. De fato, pesquisas recentes [13] sugerem que em um protocolo baseado em *proof-of-stake*, as recompensas de *staking* devem permanecer competitivas, a fim de manter altas taxas de *staking* e altos níveis de segurança, por isso as políticas deflacionárias são desaconselhadas.

Em nosso projeto, as recompensas de *staking* são o único mecanismo que cunha DOTs. Assim, é conveniente introduzir nosso modelo de inflação nesta seção também.

Lembre-se da descrição do protocolo *NPoS* (Seção 4.1) que tanto os validadores quanto os nomeadores fazem *stake* de DOTs. Eles são pagos de forma aproximadamente proporcionalmente à sua participação, mas podem ser reduzidos em até 100% em caso de má conduta. Apesar de estarem ativamente engajados por apenas uma era[3] de cada vez, eles podem continuar envolvidos por um número ilimitado de eras. Durante este período o seu *stake* é bloqueado, o que significa que não pode ser gasto e permanece bloqueado por várias semanas após sua última era de atividade, para manter os *stakers* sujeitos a cortes mesmo se uma ofensa for detectada tardiamente.

Taxa de *staking*, taxa de juros, taxa de inflação: Seja a taxa de *staking* o valor total de DOTs atualmente em *stake* por validadores e nomeadores, divididos pelo fornecimento total atual de DOT. A taxa de juros média dos *stakers* será uma função da taxa de *staking*: se a taxa de *staking* cair abaixo de um determinado valor-alvo selecionado pela governança, a taxa média de juros é aumentada, incentivando mais participação em *NPoS* e vice-versa. Por exemplo, uma taxa de *staking* de 50% poderia ser selecionada como um meio termo entre segurança e liquidez. Se a taxa de juros anual média dos *stakers* é definida em 20% neste nível, podemos esperar que a taxa de inflação flutue cerca de 50% × 20% = 10%. Assim, ao estabelecer metas para a taxa de *staking* e a taxa de juros dos *stakers*, também controlamos a taxa de inflação. Seguindo este princípio, a cada era ajustamos nossa estimativa da taxa de *staking* e usamos para calcular a quantidade total de DOTs a serem pagos aos *stakers* para aquela era.

---

[3] Lembre-se de que uma era dura aproximadamente um dia. Consulte a Tabela 1 no Apêndice.

Recompensas por meio do suporte dos validadores: Uma vez que o pagamento total para a era atual é calculado, precisamos estabelecer como ele é distribuído. Lembre-se de que o protocolo de eleição do validador (Seção 4.1) divide o *stake* ativo em suportes dos validadores, onde cada suporte do validador é composto do *stake* total de um validador mais uma fração do *stake* de seus nomeadores de apoio, e essa partição é feita de modo a tornar o suporte do validador o mais alto e uniformemente distribuídos possível, garantindo assim a segurança e a descentralização. Um mecanismo de incentivo adicional implementado para garantir descentralização ao longo do tempo está pagando o suporte do validador igualmente por trabalho igual, independentemente do seu *stake*. Como consequência, se um validador popular tiver um alto suporte, seus nomeadores provavelmente serão pagos menos por DOT em stake do que os nomeadores que apoiam um validador menos popular. Por isso, os nomeados serão incentivados a mudar suas preferências ao longo do tempo em favor de validadores menos populares (com boa reputação, no entanto), ajudando o sistema a convergir para o caso ideal onde todo o suporte dos validadores têm participação igual.

Em particular, desenvolvemos um sistema de pontos no qual os validadores acumulam pontos para cada ação realizada, e ao final de cada era os *slots* do

validador são recompensados proporcionalmente aos seus pontos. Isso garante que os validadores sejam sempre incentivados a manter alto desempenho e capacidade de resposta. As ações a serem pagas em Polkadot incluem: a) validando um bloco de *parachain*, b) produzir um bloco de *relay chain* em BABE, c) adicionando a um bloco BABE uma referência a um bloco tio[4], não referenciado anteriormente e d) produzindo um bloco tio.

Recompensas dentro de um slot de validador: Como o *stake* de um nomeador é normalmente dividido entre vários apoios do validador, seu pagamento em uma era corresponde à soma de seus pagamentos relativos a cada um desses apoios. Com o apoio do validador, o pagamento é o seguinte: Primeiro, o validador recebe uma comissão, que é um valor destinado a cobrir seus custos operacionais. Então o restante é compartilhado entre todos os *stakers* – validadores e nomeadores – proporcionalmente ao seu *stake*. Assim, o validador recebe duas recompensas separadas: uma taxa para executar um nó e um pagamento por *staking*. Ressaltamos que a taxa de comissão cabe a cada validador definir, e deve ser anunciada publicamente com antecedência. Uma taxa mais alta se traduz em um pagamento total mais alto para o validador, e menor pagamento aos seus nomeadores, assim os nomeadores geralmente preferem apoiar os validadores para reduzir as taxas, e o mercado se regula nesse sentido. Validadores que construíram uma forte reputação de confiabilidade e desempenho, no entanto, poderão cobrar uma taxa de comissão mais alta, o que é justo.

---

[4]No protocolo BABE, às vezes dois produtores de blocos podem gerar blocos diferentes A e B na mesma altura, levando a um *fork* temporário na *relay chain*. O *fork* será resolvida rapidamente e um dos blocos selecionado, digamos A, como parte da cadeia principal, enquanto o bloco B se torna tio de todos os descendentes de A. Por razões de segurança, é conveniente gravar e marcar a data e hora de todos os blocos produzidos, mas como os blocos tio não podem ser acessados via relação de pai, incentivamos os produtores de bloco a adicionar explicitamente essas referências à cadeia principal.

Finalizamos a seção com algumas observações sobre os incentivos que se espera que nosso esquema de pagamento cause nos *stakers.* Primeiro, como os validadores são bem remunerados e seu número é limitado, eles têm um incentivo para garantir altos níveis de apoio dos nomeadores para garantir a eleição, e assim valorizarão sua reputação. Ao longo do tempo, esperamos que as eleições sejam altamente competitivas e para validadores eleitos terem fortes históricos de desempenho e confiabilidade e grande *stake* como apoio. Em segundo lugar, mesmo

que os pagamentos através do suporte de diferentes validadores sejam independentes de seu *stake*, dentro do suporte de um validador cada ator é pago proporcionalmente ao seu *stake*, então sempre há um incentivo individual para aumentar seu próprio *stake*. Finalmente, se um validador obtiver um valor particularmente alto nível de apoio, pode lucrar com isso aumentando sua taxa de comissão, o que tem o efeito de aumentar sua própria recompensa com o risco de perder algumas indicações ou lançar um novo nó como candidato a validador e dividindo seu suporte entre todos os seus nós. Sobre este último ponto, damos as boas vindas aos operadores com múltiplos nós validadores, que ainda visam simplificar sua logística.

4.5.2 Limites do bloco da *relay chain* e taxas de transação

Limites de uso de recursos: Nós limitamos a quantidade de transações que um bloco de *relay chain* pode processar, a fim de, a) garantir que cada bloco possa ser processado eficientemente mesmo em nós menos potentes e evitar atrasos na produção de blocos, e b) ter disponibilidade garantida para uma certa quantidade de transações operacionais de alta prioridade, tais como relatórios de má conduta, mesmo quando há alto tráfego de rede. Em particular, definimos restrições de bloco nos seguintes recursos: comprimento de *bytes on-chain* e tempo e memória necessários para processar as transações.

Classificamos as transações em vários tipos, de acordo com seu nível de prioridade e perfil de consumo de recursos. Para cada um destes tipos, executamos testes baseados nos piores cenários de estado e para diferentes argumentos de entrada. A partir desses testes, estabelecemos estimativas conservadoras sobre a utilização de recursos para cada transação, e usamos estas estimativas para garantir que todas as restrições sobre a utilização de recursos sejam observadas.

Também adicionamos uma restrição extra aos recursos: distinguimos entre transações regulares e de alta prioridade, e permitir que as transações regulares representem até 75% do limite de recursos de cada bloco. Isso é para garantir que cada bloco tenha um espaço garantido para transações de alta prioridade de pelo menos 25% dos recursos.

Taxas de transação: Usamos o modelo descrito acima para definir o nível de taxa de uma transação com base em três parâmetros: seu modelo, seu comprimento na cadeia e seu uso esperado de recursos. Esta taxa de diferenciação é usada para refletir os diferentes custos que uma transação incorre na rede e sobre o estado e incentivar o processamento de certos tipos de transações em detrimento de outras. Uma fração de cada taxa de transação é paga ao produtor do bloco, enquanto outra fração vai para financiar o Tesouro (Seção 4.6.4). Destacamos que, para um produtor de blocos, as recompensas que vêm das taxas de transação podem constituir apenas uma pequena fração de sua receita total, apenas o suficiente para incentivar a inclusão no bloco.

Também executamos um cronograma de taxas de transação adaptável que reage ao nível de tráfego e garante que os blocos estão normalmente longe de serem cheios, para que os picos de atividade possam ser tratados de forma eficaz e tempos de inclusão longos sejam raros. Em particular, a taxa de cada transação é multiplicada por um parâmetro que evolui ao longo do tempo, dependendo do tráfego de rede atual.

Fazemos com que as taxas evoluam lentamente, de modo que a taxa de qualquer transação possa ser prevista com precisão em um período de uma hora. Em particular, não pretendemos que as taxas de transação sejam a principal fonte de renda dos *stakers*.

4.6 Governança

Polkadot utiliza mecanismos sofisticados de Governança que lhe permitem evoluir graciosamente ao longo tempo a pedido final de suas partes interessadas reunidas. Uma chave e regra infalível é que todas as mudanças no protocolo devem ser acordadas por referendo ponderado pela participação - a maioria do *stake* sempre comanda a rede.

Para fazer qualquer alteração na rede, a ideia é reunir os detentores de DOT e administrar uma decisão de atualização da rede com a ajuda do Conselho (consulte a Seção 4.6.2). Não importa se a proposta é apresentada por um titular de DOT ou pelo Conselho, acabará por ter que passar por um referendo para deixar todos os detentores de DOT, ponderados por participação, tomarem a decisão.

Cada detentor de DOT em Polkadot tem o direito de: a) apresentar uma proposta, b) endossar uma proposta para priorizá-la no calendário do referendo, c) votar em todos os referendos ativos, d) tornar-se um candidato a um assento no Conselho, e) votar em candidatos ao Conselho. Além disso, qualquer detentor de DOT pode se tornar um nomeador ou candidato a validador para participar do *NPoS* (consulte a Seção 4.1).

4.6.1 Propostas e Referendos

O centro da lógica de Polkadot é armazenado *on-chain* em uma função de transição de estado amorfo e definido em uma linguagem de plataforma neutra: WebAssembly. Cada proposta assume a forma de uma chamada de função no tempo de execução, que é capaz de modificar o próprio código de tempo de execução, alcançando o que caso contrário, exigirá um "*hard fork*". Uma proposta é então apresentada e votada por meio de referendo.

As propostas podem ser iniciadas de várias maneiras:

- uma proposta pública, que é apresentada por qualquer detentor de DOT;
- uma proposta do Conselho, apresentada pelo Conselho;
- uma proposta apresentada automaticamente como parte da promulgação de um referendo prévio, e
- uma proposta de emergência apresentada pelo Comitê Técnico (Seção 4.6.2).

Cada proposta aprovada por referendo tem um atraso de promulgação associado, ou seja, um intervalo de tempo entre o final do referendo e as mudanças que estão sendo promulgadas. Para os dois primeiros tipos de propostas acima é fixado um intervalo, provisoriamente definido para 28 dias. Para o terceiro tipo, pode ser definido conforme desejado. As propostas de emergência tratam de grandes problemas com a rede, que precisam ser acelerados, e, portanto, terá um atraso de promulgação mais curto. Ter um atraso de promulgação garante um nível de estabilidade, uma vez que dá a todas as partes um aviso prévio suficiente para se adaptarem às novas mudanças. Após esse período, a chamada para a função privilegiada associada é feita automaticamente.

Qualquer parte interessada pode apresentar uma proposta pública depositando um valor mínimo fixo de DOTs, que fica bloqueado por um determinado período. Se alguém concordar com a proposta, poderá depositar a mesma quantidade de *tokens* para endossá-lo. As propostas públicas são armazenadas em uma fila prioritária e em intervalos regulares, a proposta com mais endossos é submetida a referendo. Os *tokens* bloqueados são liberados assim que a proposta é apresentada.

As *propostas do* Conselho são apresentadas pelo Conselho e são armazenadas em uma fila de prioridade separada onde as prioridades são definidas a critério do Conselho.

Um **referendo** é um esquema de votação simples, inclusivo e ponderado. Tem período de votação fixo, após o qual os votos são computados. Os referendos são sempre binários: as opções de votação são “sim”, “não”, ou abster-se completamente.

Prazos: A cada trinta dias, uma nova proposta será apresentada e um referendo surgirá para votação. A proposta a ser apresentada é a principal proposta da fila de propostas públicas ou a fila de propostas do Conselho, alternando entre as duas filas se ambas não estiverem vazias. Se ambas as filas estão vazias, o *slot* é ignorado no calendário de referendo. Referendos múltiplos não podem ser ativos em simultâneo, exceto nos referendos de emergência que seguem um calendário paralelo.

Contagem de votos: A votação em referendos está aberta a todos os detentores de DOT com poder de voto proporcional ao seu *stake*, até um eventual multiplicador de votos que é atribuído a alguns partidos em função do seu nível de comprometimento com o sistema, como explicamos agora. Uma parte geralmente deve bloquear seus *tokens* utilizados para votação até pelo menos o período de atraso da promulgação além do final do referendo. Isto é a fim de assegurar que é necessária alguma adesão económica mínima ao resultado e para dissuadir a venda de votos. É possível votar sem nenhum bloqueio, mas nesse caso o poder de voto é uma pequena fração de um voto normal para o *stake* dado. Por outro lado, Polkadot oferecerá voluntariamente um bloqueio estendido, que permite a qualquer partido aumentar seu poder de voto, estendendo o período de tempo em que eles estão dispostos a bloquear seus *tokens.* Isso garante que os eleitores comprometidos com

o sistema a longo prazo, que estão dispostos a aumentar sua exposição à decisão de um referendo, tenham uma maior participação no assunto.

Viés de participação: pode parecer restritivo forçar um processo completo baseado nas partes interessadas a fazer algo tão pequeno quanto, digamos, reduzir o tempo de bloqueio em 5%. No entanto, sem esta regra a rede provavelmente seria instável, pois colocar seu controle fora das mãos das partes interessadas criaria um desalinhamento que pode levar à inação ou até pior. No entanto, aproveitando o fato de que a participação raramente é de 100%, podemos obter resultados diferentes dependendo das circunstâncias, elaborando equilíbrio de poder entre os atores ativos e passivos. Por exemplo, sistemas de votação simples normalmente introduzem uma noção de quórum, segundo a qual uma quantidade mínima de participação deve ser alcançada antes de uma mudança ser aprovada.

Para propostas públicas, generalizamos essa noção em um “viés de participação positiva”, onde a participação sempre torna a mudança mais provável, assumindo a mesma proporção de sim para não. Mais especificamente, em caso de baixa participação, favorecemos o lado negativo, ou *status quo*, exigindo uma aprovação por supermaioria, e à medida que a participação se aproxima de 100%, o requisito diminui para a maioria. Isso funciona em dois princípios: Em primeiro lugar, que o status quo tende a ser mais seguro do que qualquer mudança e, portanto, deve ter algum preconceito em relação a isso. Em segundo lugar que, como todos os meios de medida empírica, haverá inevitavelmente algum grau de imprecisão e volatilidade ao longo do tempo, principalmente quando a participação é baixa – um resultado pode ser 51% − 49% em um mês e depois mudar para 49% − 51%, e dados os custos envolvidos na realização das alterações de uma proposta, é vantajoso garantir que um resultado não virará logo após a aprovação.

Por outro lado, para as propostas apresentadas pelo Conselho, os referendos não têm viés de participação e a maioria é observada. O raciocínio aqui é que as propostas pré-aprovadas pelo Conselho são consideradas mais seguras e menos passíveis de reversão, de modo que os problemas mencionados anteriormente são amenizados e podemos deixar que os titulares de DOTs decidam livremente sobre o assunto.

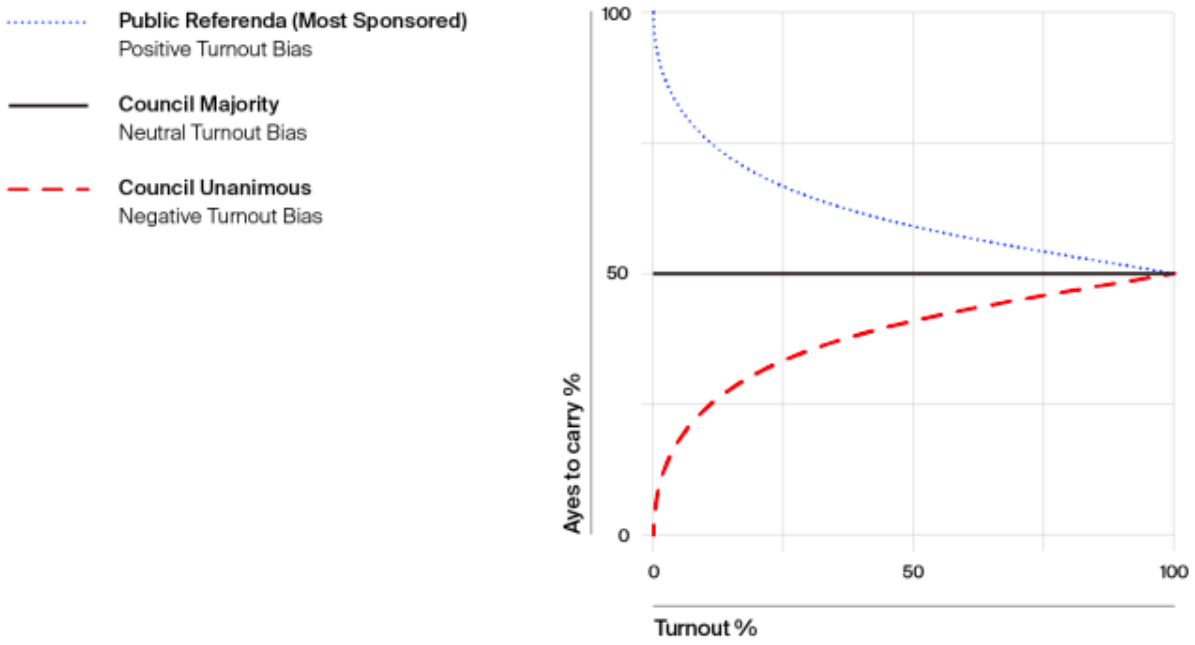

Figura 5: Viés de participação adaptável (crédito da imagem: Ignasi Albero).

Por último, no caso excepcional de uma proposta do Conselho receber o apoio unânime de todos os membros do Conselho, observará um “viés de participação negativo”. Este é o oposto simétrico do primeiro caso, onde a participação adicional sempre torna a mudança menos provável, favorecemos o lado positivo em caso de baixa participação por exigir uma supermaioria de nãos para rejeitar a proposta, e como participação se aproxima de 100%, o requisito é reduzido para majoritários. Veja a Figura 5.

4.6.2 O Conselho e o Comitê Técnico

O Conselho é uma entidade composta por vários participantes, cada um representado por uma conta *on-chain*. Seus objetivos são representar as partes interessadas passivas, apresentar propostas sensatas e importantes e cancelar propostas incontestavelmente perigosas ou maliciosas.

O Conselho revisará constantemente as propostas de candidatos para lidar com questões emergenciais no sistema. Uma proposta candidata é oficialmente apoiada pelo Conselho – e entra na fila de propostas do Conselho – somente após ser aprovada por uma maioria estrita dos membros do Conselho, sem nenhum

membro exercendo um veto. Uma proposta candidata pode ser vetada apenas uma vez; se, após um período de espera, ela for mais uma vez aprovada pela maioria dos membros do Conselho, não poderá ser vetada uma segunda vez.

Como mencionado anteriormente, no caso de todos os membros votarem a favor, uma proposta do Conselho é considerada incontroversa e possui um viés de participação especial que a torna mais provável de ser aprovada.

Finalmente, os membros do Conselho podem votar pelo cancelamento de qualquer proposta, independentemente de quem a apresentou, mas seu voto deve ser unânime. Dado que a unanimidade é um requisito elevado, espera-se que esta medida só seja usada quando for um movimento totalmente incontroverso. Isso pode funcionar como um último recurso se houver um problema encontrado no final do dia com a proposta de um referendo, tal como um erro no código tempo de execução ou uma vulnerabilidade que a proposta instituiria. Se o cancelamento for controverso o suficiente para que haja pelo menos um dissidente, então será deixado para todos os detentores de DOT em grandes quantidades para determinar o destino da proposta, com os votos de cancelamento registrados do Conselho servindo como bandeiras vermelhas para que os eleitores prestem atenção especial.

Eleição de membros do Conselho: Haverá 23 assentos no Conselho de Polkadot. Uma eleição geral para todos os assentos ocorrerá uma vez por mês. Todos os detentores de DOT são livres para registrar suas candidaturas para o Conselho, e livres para votar em qualquer número de candidatos, com direito a voto proporcional ao seu *stake*. Muito parecido com o problema da eleição do validador no *NPoS*, este é um problema de eleição com vários vencedores, baseado em cédulas de aprovação. Podemos, assim, resolvê-lo usando o mesmo algoritmo que usamos para *NPoS*, que em particular oferece a propriedade de representação proporcional justificada; consulte a Seção 4.1 para obter mais detalhes. Esta propriedade garante que o Conselho eleito representará o maior número possível das minorias, garantindo assim que a Governança permaneça descentralizada e resistente à captura. Os membros do Conselho podem ser reeleitos indefinidamente, desde que a aprovação permaneça alta o suficiente.

O Comitê Técnico é composto por um voto único para cada equipe que

implementou com sucesso e independentemente ou especificou formalmente o protocolo em Polkadot, ou em sua rede canário Kusama[5]. As equipes podem ser adicionadas ou removidas por maioria simples do Conselho.

O Comitê Técnico é a última linha de defesa do sistema. Seu único objetivo é detectar problemas presentes ou iminentes no sistema, como erros no código ou vulnerabilidades de segurança, e propor e acelerar referendos de emergência. Uma proposta de emergência precisa de uma aprovação de pelo menos três quartos dos membros do Conselho e pelo menos dois terços dos membros do Comitê Técnico a fim de serem apresentados. Uma vez apresentado, o referendo é acelerado e corre em paralelo com o calendário dos referendos regulares, com um período de votação muito mais curto, e um período de promulgação próximo de zero. A mecânica de aprovação neste caso não muda em relação ao que seria de outra forma, ou seja, uma maioria simples ou, no caso de uma aprovação unânime do Conselho, um viés baseado no comparecimento para aprovação.

Ressaltamos que por questões práticas o Comitê Técnico não é eleito democraticamente, mas, ao contrário, tem um escopo de ação extremamente reduzido e nenhum poder para agir unilateralmente, como explicado nas linhas acima. Espera-se que este mecanismo seja suficiente para correções de erros não contenciosos e atualizações técnicas, mas atendendo aos requisitos impostos, pode não ser eficaz no caso de emergências que tenham um toque de sensibilidade política ou importância estratégica para elas.

---

[5]http://kusama.network

4.6.3 Alocação de *slots* de *parachain*

Nós utilizamos leilões para ter um procedimento de alocação de *parachain* justo, transparente e não permissionado. De um modo geral, os interessados em receber um *slo*t de *parachain* participam de um leilão com lances feitos em DOT. A parte com o lance mais alto é declarada vencedora e recebe um *slot* por um período de tempo específico, com seu lance se tornando um depósito bloqueado que é liberado no final do referido período. O custo de locação do *slot* corresponde assim ao custo de

oportunidade de ter este depósito bloqueado. Este depósito realizado em DOTs também estabelece o poder de voto da *parachain* na governança de Polkadot.

Como a implementação de leilões *seal-bid* é difícil e para evitar o *sniping* de lances, adotamos um mecanismo de Leilão de velas [16] com fechamento determinado retroativamente. Entrando em detalhes, planejamos ter um leilão a cada poucas semanas, onde em cada leilão quatro *slots* consecutivos de seis meses são oferecidos para locação. Um lance pode ser feito para qualquer combinação de um, dois, três ou quatro *slots*, num total de dez períodos possíveis de 6, 12, 18 ou 24 meses. Uma vez que o leilão começa, as partes podem lançar ofertas como transações para qualquer um desses dez períodos, dentro de uma janela fixa de tempo com duração de várias horas. Uma parte tem permissão para enviar várias ofertas, onde uma oferta é registrada somente se: a) vencer o lance mais alto atual para o período correspondente, e b) a parte não se tornar o vencedor provisório de dois ou mais períodos com intervalos entre eles. Por exemplo, o vencedor do período (1, 2) – constituído pelas duas primeiras vagas – não pode licitar no período (4) – o quarto *slot* – até que alguém ultrapasse o período anterior.

Os objetivos declarados deste projeto são incentivar as partes a licitar antecipadamente e evitar o *sniping* de lances, para dar aos projetos menos financiados a chance de ganhar uma vaga, garantindo assim a natureza descentralizada de Polkadot, e para desencorajar ataques de luto das partes que aumentam o valor do lance vencedor com nenhuma intenção de eles mesmos vencerem.

4.6.4 Tesouro

O Tesouro levanta fundos continuamente. Esses fundos são usados para pagar desenvolvedores que fornecem atualizações de *software*, aplicar quaisquer alterações decididas por referendos, ajustar parâmetros e geralmente manter o sistema funcionando sem problemas. Os fundos também podem ser usados para outros objetivos, como atividades de *marketing*, eventos comunitários e divulgação. Em última análise, isso é controlado por todos os detentores de DOTs via Governança e será a comunidade e sua imaginação coletiva e julgamento que realmente determinará o curso do Tesouro.

Os fundos para o Tesouro são levantados de duas maneiras:

1. canalizando algumas das recompensas do validador que vêm da cunhagem de novos *tokens* e
2. canalizando uma fração das taxas de transação e dos cortes.

O primeiro método permite-nos manter uma taxa de inflação fixa ao mesmo tempo que as recompensas do validador dependem do nível de *staking* (consulte a Seção 4.5.1): a diferença entre o *tokens* cunhados programados e as recompensas do validador são atribuídas ao Tesouro em cada era. Nós também argumentamos que é conveniente que uma fração de todos os cortes sejam redirecionados para o Tesouro: seguindo um evento que produziu forte corte de *stake*, o sistema provavelmente precisará de fundos adicionais para desenvolver atualizações de *software* ou novas infraestruturas que lidam com um problema existente, ou pode ser decidido por Governança para reembolsar parte de *stake* cortado. Assim, faz sentido ter os DOTs cortados disponíveis no Tesouro, em vez de queimá-los e ter que cunhar mais DOTs logo em seguida.

4.7 Criptografia

Em Polkadot, necessariamente distinguimos entre diferentes permissões e funcionalidades com diferentes chaves e tipos de chave, respectivamente. Nós os categorizamos aproximadamente em chaves de conta com as quais usuários interagem e chaves de sessão que os nós gerenciam sem intervenção do operador além de um processo de certificação.

4.7.1 Chaves de conta

As chaves de conta têm um saldo associado do qual porções podem ser bloqueadas para desempenhar funções no *staking*, locação de recursos e governança, incluindo a espera de algum tipo de período de desbloqueio. Permitimos vários bloqueios de

duração variável, tanto porque esses papéis impõem restrições diferentes quanto para vários períodos de desbloqueio executados simultaneamente.

Incentivamos a participação ativa em todas essas funções, mas todas elas exigem assinaturas ocasionais das contas. Ao mesmo tempo, as chaves de conta têm melhor segurança física quando guardadas em locais inconvenientes, como cofres, o que torna árdua a assinatura. Evitamos esse atrito para os usuários da seguinte forma.

As contas que bloqueiam fundos para *staking* são chamadas de contas *stash*. Todas as contas *stash* registram um certificado *on-chain* que delega todos os poderes de operação e nomeação do validador a alguma conta controladora, e também designa alguma chave *proxy* para votos de governança. Neste estado, o controlador e contas *proxy* podem assinar para a conta *stash* nas funções de *staking* e governança, respectivamente, mas não transferir fundos.

Atualmente, oferecemos suporte a ambos, ed25519 [6] e Schnorrkel/sr25519 [9] para chaves de conta. Essas são ambas assinaturas do tipo Schnorr implementadas usando a curva Ed25519, então ambas oferecem segurança semelhante. Recomendamos chaves ed25519 para usuários que precisam do Módulo de Segurança de *Hardware* (HSM) ou outra solução externa de gerenciamento de chaves, enquanto Schnorrkel/sr25519 fornece funcionalidade mais amigável para *blockchain*, como Derivação de Chave Determinística Hierárquica (HDKD) e multi-assinaturas.

Em particular, Schnorrkel/sr25519 usa a implementação Ristretto [21] Decaf de Mike Hamburg [18, §7], que fornece os pontos livres de 2 torção da curva Ed25519 como uma ordem principal do grupo. Evitar o co-fator como este significa que o Ristretto torna a implementação de protocolos mais complexos significativamente mais segura. Empregamos Blake2b para o *hashing* mais convencional em Polkadot, mas O próprio Schnorrkel/sr25519 usa STROBE128 [19], que é baseado em Keccak-f (1600) e fornece uma interface de *hash* adequada para assinaturas e provas de conhecimento zero não interativas (NIZKs).

4.7.2 Chaves de sessão

Cada chave de sessão preenche aproximadamente uma função específica em consenso ou segurança. Como regra, as chaves de sessão ganham autoridade apenas de um certificado de sessão, assinado por alguma chave do controlador, que delega *stake.*

A qualquer momento, a chave do controlador pode pausar ou revogar este certificado de sessão e/ou substituir por novas chaves de sessão. Todas as novas chaves de sessão podem ser registradas antecipadamente, e a maioria deve ser, para que os validadores possam fazer uma transição limpa para o novo *hardware* emitindo certificados de sessão que só se tornarão válidos após alguma sessão futura. Sugerimos o uso de mecanismo de pausa para manutenção de emergência e usando a revogação se uma chave de sessão puder ser comprometida.

Preferimos que as chaves de sessão permaneçam vinculadas a uma máquina física porque isso minimiza o risco de equívoco acidental. Pedimos aos operadores do validador que emitam certificados de sessão usando um protocolo RPC, não para manipular as próprias chaves secretas de sessão.

Quase todas as primeiras redes de *proof-of-work* têm uma infraestrutura de chave pública negligente que incentiva a duplicação de chaves secretas de sessão entre máquinas e, portanto, reduz a segurança e leva a corte inútil.

Não impomos restrições prévias à criptografia empregada por componentes específicos ou seus tipos de chaves de sessão associadas[6]

---

[6] Sempre implementamos criptografia para Polkadot em código nativo, não apenas porque o tempo de execução sofre as penalidades de desempenho do WASM, mas porque todos os protocolos de consenso de Polkadot são parcialmente implementados, fora o tempo de execução nos módulos Substrate.

No BABE 4.3.1, os validadores usam chaves Schnorrkel/sr25519 tanto para assinaturas Schnorr regulares, bem como para uma função aleatória verificável (VRF) baseada em NSEC5 [28].

Um VRF é o análogo de chave pública de uma função pseudoaleatória (PRF), também conhecida como *hash* criptográfico, que funciona com uma chave distinta, como muitos MACs. Nós concedemos *slots* de produção em bloco quando o produtor de bloco obtém uma saída VRF suficientemente baixo VRFsk(*re*||*slot number),* para que qualquer pessoa com as chaves públicas VRF possa verificar se os blocos foram produzidos no *slot* correto, mas apenas os produtores de bloco conhecem seus *slots* com antecedência por meio de sua chave secreta VRF.

Como em [15], nós fornecemos uma fonte de aleatoriedade para as entradas VRF juntando todas as saídas VRF que formam a sessão anterior, que requer que as chaves BABE sejam registradas pelo menos duas épocas completas antes de serem usadas.

Reduzimos a maleabilidade da saída do VRF fazendo *hash* da chave pública do signatário junto com a entrada, que melhora drasticamente a segurança quando usado com HDKD. Também fazemos *hash* da entrada VRF e saída em conjunto ao fornecer saída usada em outro lugar, o que melhora a composição quando usado como um oráculo aleatório em provas de segurança. Veja a construção do 2Hash-DH do Teorema 2 na página 32 no apêndice C de [15].

No GRANDPA 4.3.2, os validadores devem votar usando assinaturas BLS, que suportam agregação convenientes de assinatura e seleciona a curva BLS12-381 do ZCash para desempenho. Existe o risco de BLS12-381 pode cair significativamente abaixo de 128 *bits* de segurança, devido aos avanços da peneira de campo numérico. Se e quando isso acontecer, esperamos que a atualização do GRANDPA para outra curva seja simples.

Nós tratamos as chaves de transporte libp2p aproximadamente como chaves de sessão também, mas elas incluem chaves de transporte para nós *sentry*, não apenas para o próprio validador. Como tal, o operador interage um pouco mais com estas.

4.8 Rede

Nas seções anteriores, falamos sobre nós enviando dados para outro nó ou outro conjunto de nós, sem ser muito específico sobre como isso é alcançado. Fazemos isso para simplificar o modelo e para delinear uma separação de preocupações entre diferentes camadas.

Obviamente, em um sistema descentralizado do mundo real, a parte de rede também deve ser descentralizada - não adianta se toda a comunicação passar por alguns servidores centrais, mesmo que o protocolo de alto nível executado em cima dele seja descentralizado em relação às suas entidades. A título de exemplo concreto: em certos modelos de segurança, incluindo a configuração tradicional de tolerância a falhas Bizantina, os nós são modelados como possivelmente maliciosos,

mas nenhuma consideração é dada a bordas maliciosas. Um requisito de segurança como “> 1/3 dos nós são honestos” no modelo, na verdade se traduz em “> 1/3 dos nós são honestos e todos podem se comunicar de forma perfeitamente confiável entre si o tempo todo” na realidade. Por outro lado, se uma borda na realidade, é controlada por um ISP malicioso é o(s) nó(s) correspondente(s) que deve(m) ser tratado como malicioso em qualquer análise sob o modelo. Mais significativamente, se a rede de comunicações subjacente é centralizada, isso pode dar às partes centrais a capacidade de “corromper” > 1/3 dos nós dentro do modelo quebrando assim suas suposições de segurança, mesmo que não na verdade, têm direitos de execução arbitrários em muitos nós.

Nesta seção, esboçamos e enumeramos as primitivas de comunicação que precisamos em Polkadot, e traçar um projeto de alto nível sobre como alcançá-los de forma descentralizada, com os específicos a serem refinados à medida que avançamos com um sistema de produção.

4.8.1 Visão geral da rede

Conforme discutido acima, Polkadot consiste em uma única *relay chain* interagindo com muitas *parachains* e fornecendo-lhes serviços de segurança. Estes requerem o seguinte nível de funcionalidade de rede, geralmente para distribuição e disponibilidade:

1. Como em todos os protocolos do tipo *blockchain*, a *relay chain* requer:

    (a) aceitar transações de usuários e outros dados externos (conhecidos coletivamente como dados *extrínsecos* ou *extrínsecos*) e distribuí-los
    (b) distribuir produtos do subprotocolo de agrupamento 4.2
    (c) distribuição de produtos do subprotocolo de finalização 4.3.2
    (d) sincronizar o estado previamente finalizado

Como um caso especial importante, as *parachains* podem optar por se implementar de acordo com a estrutura acima, talvez até reutilizando os mesmos subprotocolos.

Parte da implementação de Polkadot é arquitetada como uma biblioteca separada chamada `substrate` para que eles façam exatamente isso.

2. Para interação entre a *relay chain* e as *parachains*, exigimos:

    (a) aceitar blocos de *parachain* de coletores de *parachain*
    (b) distribuição de metadados de bloco de *parachain*, incluindo atestados de validade
    (c) distribuir os dados do bloco *parachain* e disponibilizá-los por um tempo 4.4.2, para fins de auditoria

3. Para interação entre *parachains*, exigimos:

    (a) distribuição de mensagens entre *parachains* 4.4.3, especificamente dos remetentes relevantes aos destinatários relevantes

Para cada um dos requisitos de funcionalidade acima, nós os satisfazemos com o seguinte:

- 1(b), 1(c), 2(b) - os produtos são transmitidos como estão (ou seja, sem codificação adicional) via *gossip*.
- 1(a), 1(d), 2(a) - efetivamente, um conjunto de nós fornece o mesmo serviço distribuído aos clientes. Para aceitar extrínsecos ou blocos, os clientes os enviam diretamente ao nó servidor; para sincronização, os clientes recebem dados verificáveis diretamente do nó servidor.
- 2(c) - caso especial, abaixo. Resumidamente, os dados são codificados para eliminação para que diferentes destinatários recebam uma pequena parte; as peças são enviadas diretamente via QUIC.
- 3(a) - caso especial, abaixo. Resumidamente, as mensagens são enviadas diretamente via QUIC; no processo, as caixas de saída são reagrupadas em caixas de entrada e estas podem ser transferidas como um lote para os destinatários.

Abordaremos isso com mais detalhes nas próximas seções. Finalmente, falamos sobre as camadas inferiores sustentando todos esses subprotocolos, ou seja, autenticação, transporte e descoberta.

4.8.2 *Gossiping*

Este subprotocolo é usado para a maioria dos produtos da *relay chain*, onde todos precisam ver mais ou menos a mesma informação pública. Parte de sua estrutura também é usada para quando um nó fica *offline* por muito tempo e precisa sincronizar quaisquer dados mais recentes que não tenha visto antes.

A relay chain de Polkadot forma uma rede de sobreposição de *gossip* em cima da rede de comunicações físicas, como uma maneira eficiente de fornecer um meio de transmissão descentralizado. A rede consiste em um número conhecido de nós confiáveis (validadores) que foram autorizados por meio de *staking*, e um número desconhecido de nós não confiáveis (nós completos que não realizam validação) da Internet aberta não permissionados. (Como um aviso, lembre-se de que alguns dos nós não confiáveis podem ter outras funções conforme definidas anteriormente, por exemplo coletor de *parachains*, *fishermen*, etc.)

Uma abordagem simples baseada em empurrar é implementada atualmente, com cachês de rastreadores baseados em *hash* para evitar enviar duplicatas para *peers* e algumas restrições para evitar os ataques mais comuns de *spam*.

- Dados técnicos só podem ser recebidos em ordem de dependência; *peers* não têm permissão para enviá-los fora de ordem. Embora isso diminua a eficiência no nível da rede, é simples de implementar e fornece um nível saudável de segurança.

- Para comunicar com eficiência aos pares remetentes o que eles têm permissão para enviar em ordem de dependência, os pares periodicamente atualizam uns aos outros com a visão das últimas pontas da cadeia.

Existem também regras de restrição mais específicas aplicadas a dados técnicos pertencentes aos vários subprotocolos de nível superior usando o protocolo de

gossip, para evitar transmissões obsoletas ou dados técnicos desnecessários. Por exemplo, para o GRANDPA só permitimos que sejam recebidos dois votos para cada tipo de voto, número da rodada e eleitor; quaisquer outros votos serão ignorados. Para produção de blocos válidos apenas produtores de blocos podem produzir um bloco por rodada; quaisquer outros blocos serão ignorados.

Há suporte básico para nós *sentry*, servidores *proxy* que são essencialmente os únicos vizinhos de um servidor privado, executando operações mais críticas de segurança, como a função de validador.

A topologia de rede é um ponto fraco atualmente; nós conectam-se uns aos outros em um *ad-hoc* base realizando pesquisas aleatórias no catálogo de endereços. O trabalho adicional prosseguirá ao longo de duas frentes:

1. Os nós confiáveis reservarão uma parte de sua largura de banda e recursos de conexão, para formar uma estruturada de sobreposição com uma topologia determinista, mas imprevisível, que muda a cada era. Para nós executando atrás de *sentry*, isso significa efetivamente que seus nós *sentry* participam desta topologia.

2. Para o restante da capacidade de recursos dos nós confiáveis e para a capacidade de recursos do conjunto de nós não confiáveis, eles selecionarão vizinhos por meio de um esquema baseado em medições de latência, com os detalhes a serem decididos. Notavelmente, para boas propriedades de segurança, queremos um esquema que não simplesmente escolha o "mais próximo primeiro”, mas também alguns links distantes.

Em certo sentido, isso pode ser visto como os nós confiáveis formam um núcleo com os não confiáveis. nós ao seu redor - mas observe que os nós confiáveis devem usar alguns de seus recursos para servir nós não confiáveis também. Ambas as topologias são escolhidas para mitigar ataques de eclipse, assim como ataques Sybil, no caso não confiável não permissionado.

O trabalho adicional também incluirá algum tipo de protocolo de reconciliação definido, para reduzir ainda mais a redundância quando muitos remetentes tentarem enviar o mesmo objeto para o mesmo destinatário de uma só vez; e potencialmente

olhar para o levantamento da restrição de ordem de dependência, mantendo a segurança.

4.8.3 Serviço distribuído

Este subprotocolo é usado quando alguma parte de Polkadot está prestando um serviço para alguma entidade externa, nomeadamente, 1(a) aceitar transações da *relay chain*, 1(d) sincronizar o estado da *relay chain* e 2(a) aceitando blocos agrupados na lista acima.

Em nossa implementação inicial, isso envolve simplesmente procurar um determinado destino definido no catálogo de endereços, selecionando alguns nós deste conjunto e conectando-se a eles. Para 1(a) e 1(d) o conjunto alvo é todo o conjunto de validadores, e para 2(a) o conjunto alvo é o conjunto de validadores de *parachain* para a *parachain* do compilador do cliente. Ambos podem ser recuperados diretamente do estado da cadeia, e de fato para 1(a) este é simplesmente o mesmo processo de entrar na rede *gossip*.

Trabalhos futuros considerarão a questão do balanceamento de carga das conexões da camada de transporte sobre todo o conjunto de metas, além de garantir a disponibilidade. Isso pode exigir sofisticação adicional no livro de endereços.

4.8.4 Armazenamento e disponibilidade

Este subprotocolo aborda a rede usada para o subprotocolo de Disponibilidade e Validade descrito na Seção 4.4.2.

Lembre-se de que, para escalabilidade, Polkadot não exige que todos armazenem o estado de todo o sistema, ou seja, todo o estado apontado por todos os blocos. Em vez disso, cada bloco de *parachain* é dividido em pedaços por codificação de apagamento, de modo que haja 1 pedaço para cada validador para um total de N peças, o limite de eliminação sendo blocos $ceil(N/3)$ por motivos de segurança explicados em outro lugar. Todas as peças estão inicialmente disponíveis nos validadores de *parachai*n relevantes, tendo sido submetidas por alguns dos coletores. (Neste papel, os validadores de *parachain* também são chamados de

verificadores (checkers). As peças são então distribuídas seletivamente nas seguintes fases:

1. Distribuição - cada validador inicialmente quer 1 dessas peças, e os validadores de *parachain* deve distribuí-las como tal.
2. Recuperação - os *verificadores de aprovação* precisam estar convencidos de que os validadores $ceil(N/3)$ têm seus pedaços, e alguns deles tentarão realmente recuperá-los.
3. Recuperação adicional - ainda mais tarde, e opcionalmente, outras partes não validadoras também podem querer realizar verificações adicionais, por exemplo em resposta aos alertas dos *fishermen*, e novamente vai querer qualquer $ceil(N/3)$ das peças.

Este subprotocolo visa, portanto, garantir que o limite esteja disponível e possa ser recuperado dos validadores relevantes por um período de tempo razoável, até pelo menos as últimas fases estarem completas. Seguiremos um protocolo tipo *bittorrent* com as seguintes diferenças:

- Tanto na distribuição quanto na recuperação, o conjunto de destinatários é conhecido. Assim, as peças podem ser empurradas preventivamente de validadores que já possuem a peça, além de puxar semântica *bittorren*t.
- Os validadores por trás dos nós *sentry* os usarão como *proxies*, em vez de enviar diretamente.
- Em vez de um rastreador centralizado, informações semelhantes a rastreadores, como quem tem qual peça, são transmitidas através da rede de *gossip* da *relay chain*.

Espera-se que os verificadores preliminares estejam totalmente ou principalmente conectados; isso é preexistente requisito para o protocolo de agrupamento também. Os verificadores de aprovação também devem ser totais ou parcialmente conectados, para ajudar o processo de recuperação a ser concluído mais rapidamente.

Além disso, os nós podem se comunicar com quaisquer outros nós conforme o protocolo achar adequado, semelhante ao *bittorrent*. Para proteger contra ataques

DoS, eles devem implementar restrições de recursos como em *bittorrent* e, além disso, os nós devem se autenticar e se comunicar apenas com outros validadores, incluindo os verificadores preliminares e de aprovação. Partes não validadoras nesta última fase opcional será fornecida com um *token* de autenticação para esta finalidade. Em separado mais documento detalhado, propomos um esquema para balanceamento de carga, de modo que os nós não sobrecarreguem outros nós em suas escolhas aleatórias.

Existem várias razões pelas quais não consideramos muito adequado, usar uma camada de topologia estruturada para este componente:

1. Cada peça é enviada para pessoas específicas, e não para todos.
    (a) As pessoas que desejam um dado específico, sabem onde obtê-lo - exemplo, validadores, para suas próprias partes, ou seja, os verificadores preliminares.
    (b) Outras pessoas querem partes não específicas - exemplo, verificadores de aprovação, querem qualquer 1/3 de todas as partes para poder reconstruir.

As topologias de sobreposição geralmente são mais úteis para exatamente o oposto dos requisitos de uso acima:

1. Cada parte de dados é enviada para quase todos, ou
2. As pessoas querem um dado específico, mas não sabem de onde obtê-lo.

Por exemplo, o *bittorrent* tem requisitos semelhantes e não usa uma sobreposição estruturada - os pares ali se conectam a outros pares de acordo com a necessidade.

4.8.5 Mensagens de cadeia cruzada

Este subprotocolo endereça o esquema de rede usado para o subprotocolo de mensagens XCMP descrito na Seção 4.4.3.

Para recapitular essa seção, as *parachains* podem enviar mensagens umas para as outras. O conteúdo das mensagens de saída faz parte dos blocos de

parachains PoV, enviados pelos coletores de *parachain* que enviam para seus validadores. Isso é distribuído para outros validadores como parte do protocolo de disponibilidade. Blocos da relay chain contêm metadados que descrevem as mensagens de saída relevantes correspondentes às mensagens de entrada para cada *parachain*. O trabalho da rede XCMP, portanto, é para cada *parachain do* destinatário para obter suas mensagens recebidas das caixas de saída.

Observe que, sem qualquer sofisticação adicional, isso pode sempre ser feito recuperando as partes codificadas apagadas do protocolo A&V, por exemplo, através da rede *gossip*, e decodificando todas as caixas de saída de todos os potenciais remetentes. Isso, obviamente, é muito ineficiente - ambos utilizando um meio de transmissão de dados em que apenas a *parachain* receptora está interessada nos dados sendo recuperados, que inclui mensagens enviadas para a *parachains* que não seja o destinatário. No entanto, isso pode servir como ingênua implementação inicial para os estágios iniciais da rede, onde o tráfego deve ser baixo.

Os trabalhos futuros seguirão as seguintes linhas. Uma visão do problema é como converter as caixas de saída de todos os remetentes, em caixas de entrada de todos os destinatários. Uma vez feito isso, qualquer destinatário pode simplesmente recuperar a caixa de entrada diretamente, de quem fez a conversão. Nós notamos que a estrutura de nossa rede A&V tem um requisito de comunicação muito semelhante - lá, as peças de cada bloco de *parachain* devem ser distribuídas para todos os outros validadores e, inversamente, cada validador tem que receber um pedaço de cada bloco de *parachain*. Portanto, nossos principais esforços serão direcionados para estender o protocolo de rede A&V, para auxiliar a conversão de caixas de saída XCMP em caixas de entrada.

Uma diferença importante que devemos abordar, é que as peças em A&V têm algumas redundâncias, enquanto as mensagens XCMP não têm redundância embutida e devem ser distribuídas de forma confiável. Aplicar a codificação de apagamento também é uma solução direta e óbvia, mas também exploraremos alternativas.

4.8.6 Nós *Sentry*

Às vezes, os operadores de rede desejam organizar aspectos de sua rede física por razões de segurança operacional. Alguns desses arranjos são independentes e compatíveis com o projeto de qualquer protocolo descentralizado, que normalmente funciona na camada acima. Entanto alguns outros arranjos precisam de tratamento especial pelo protocolo descentralizado, em particular arranjos afetando a acessibilidade dos nós.

Para esses casos de uso, Polkadot suporta a execução de nós completos como nós *sentry* de outro nó completo que só podem ser alcançados por esses nós *sentry*. Isso funciona melhor quando se executa vários nós *sentry* para um único nó completo privado. Em termos de protocolo, brevemente, nós *sentry* são vizinhos regulares de seu nó privado, com alguns metadados adicionais para dizer aos outros para comunicar este nó privado através de seus nós *sentry*. No modo de envio direto, eles agem de forma semelhante aos servidores *TURN*, sem qualquer restrição de gargalo de recursos, pois cada nó *sentry* está atendendo apenas a um nó privado. Essas adições são bastante diretas e mais detalhes estão disponíveis em outros lugares.

Não é necessário executar nós *sentry*, por exemplo, se você acredita que os benefícios da segurança supramencionada não valem o custo adicional de latência.

Segue uma breve discussão sobre as compensações de segurança dessa abordagem. Um benefício de uma topologia física restrita, é oferecer suporte ao balanceamento de carga e proteção *DoS* em vários *proxies* de *gateway*. A indireção também pode ajudar a proteger o nó privado se houver *exploits* no *software* - embora note que isso não cobre as explorações mais graves que dão direitos de execução no nó *sentry*, que pode então ser usado como plataforma de lançamento para ataques ao nó privado. Portanto, embora não acreditemos que um endereço público seja, em si mesmo, uma responsabilidade de segurança quando o código de serviço é bem escrito, os nós *sentry* podem ajudar a mitigar esses outros cenários.

(Uma possibilidade alternativa é o operador de rede executar *proxies* de nível inferior, como IP ou *proxies* TCP, para seu nó privado. Isso certamente pode ser feito sem qualquer suporte de protocolo de Polkadot Uma vantagem dos nós *sentry* em relação a este cenário é que o tráfego vindo de um nó *sentry* passou por algum nível de verificação e higienização como parte do protocolo de Polkadot, que estaria faltando para um *proxy* de nível inferior. Claro que pode haver *exploits* contra isso,

mas estes são tratados com alta prioridade, uma vez que são vetores de amplificação utilizáveis contra toda a rede.)

4.8.7 Autenticação, transporte e descoberta

Nos protocolos seguros em geral, assim como em Polkadot, as entidades se referem umas às outras por suas chaves criptográficas públicas. Não há forte associação de segurança com referências fracas, como endereços IP, uma vez que normalmente são controlados não pelas próprias entidades, mas por seus provedores de comunicações.

No entanto, para comunicar, precisamos de algum tipo de associação entre entidades e seus endereços. Em Polkadot, usamos um esquema semelhante ao de muitas outras *blockchains*, que está usando a tabela de *hash* distribuído (DHT), Kademlia [22]. Kademlia é um DHT que usa a métrica de distanciamento XOR e é frequentemente usada para redes com alto *churn*. Usamos a implementação libp2p Kademlia da Protocol Labs, com algumas mudanças para este fim. Para evitar ataques de Eclipse [20], permitimos tabelas de roteamento grandes o suficiente para conter pelo menos alguns nós honestos, e estão em processo de implementação da abordagem S-Kademia para roteamento *multipath*.

Atualmente, este serviço de *catálogo de endereços* também é usado como o principal mecanismo de descoberta - os nós realizam pesquisas aleatórias no espaço das chaves quando elas ingressam na rede e conectam a qualquer conjunto de endereços que sejam desenvolvidos. Da mesma forma, os nós aceitam qualquer conexão de entrada. Isso facilita o suporte a clientes leves e outros usuários sem privilégios, mas também facilita a execução de ataques DoS.

Trabalhos adicionais irão desacoplar o mecanismo de descoberta do catálogo de endereços, conforme descrito em *Gossiping* (4.8.2), resultando em uma topologia de rede mais segura. Parte disso exigirá que alguma fração das conexões de nível de transporte sejam autenticadas em relação ao conjunto de validadores atualmente confiável. Entretanto, também exigimos manter a capacidade de aceitar conexões de entrada de entidades não autenticadas, e isso precisa ser restrito com base em recursos, sem privar as entidades autenticadas.

Trabalhos futuros também irão desacoplar a implementação do catálogo de endereços de sua interface, por exemplo: podemos colocar partes *on-chain*. Isso tem diferentes compensações de segurança de um catálogo de endereços baseado em Kademlia, alguns dos quais estão fora do escopo atual de Polkadot, como localização da privacidade. Ao oferecer diferentes possibilidades, esperamos satisfazer um conjunto diversificado de nós com diferentes requisitos de segurança.

5 Trabalho Futuro

Para trabalhos futuros, planejamos nos concentrar em uma série de extensões para Polkadot. Queremos adicionar um modelo de incentivo para os *fishermen*, para garantir que eles sejam incentivados a denunciar comportamentos mal intencionados. Para habilitar mensagens sem confiança, estamos trabalhando no SPREE A.1. Também estamos interessados em aumentar ainda mais a escalabilidade de Polkadot, por exemplo, investigando a ideia de ter *relay chains* aninhadas. Além disso, estamos trabalhando para incorporar os protocolos de ponte A.2 a outras cadeias como Bitcoin, Ethereum e Zcash. Além disso, para aumentar a usabilidade, estamos planejando habilitar *Parathreads*, que têm a mesma utilidade que *parachains*, mas são temporárias e têm um modelo de taxa diferente.

Reconhecimento
Gostaríamos de agradecer a Bill Laboon da Web3 Foundation por seu feedback e aos desenvolvedores da Parity Technologies por suas contribuições úteis e boas discussões.

Referências

[1] Availability and validity scheme. https://research.web3.foundation/en/latest/polkadot/Availability_and_Validity/.
[2] Blind assignment for blockchain extension (babe). https://research.web3.foundation/ en/latest/polkadot/BABE/Babe/.

[3] Visa inc. at a glance, Dec 2015. Accessed: 2020-02-25.

[4] Mustafa Al-Bassam, Alberto Sonnino, and Vitalik Buterin. Fraud and data availability proofs: Maximising light client security and scaling blockchains with dishonest majorities. arXiv preprint arXiv:1809.09044, 2018.

[5] Handan Kılın¸c Alper. Consensus on clock in universally composable timing model. Cryptology ePrint Archive, Report 2019/1348, 2019. https://eprint.iacr.org/2019/1348. [6] Daniel Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. Journal of Cryptographic Engineering volume, 2012(2):77–89, 2012.

[7] Markus Brill, Rupert Freeman, Svante Janson, and Martin Lackner. Phragm´en's voting methods and justified representation. In Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[8] Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on bft consensus. arXiv preprint arXiv:1807.04938, 2018.

[9] Jeffrey Burdges. schnorrkel: Schnorr vrfs and signatures on the ristretto group. https: //github.com/w3f/schnorrkel, 2019.

[10] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform, 2014. Accessed: 2016-08-22.

[11] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. arXiv preprint arXiv:1710.09437, 2017.

[12] Alfonso Cevallos and Alistair Stewart. Validator election in nominated proof-of-stake. arXiv preprint arXiv:2004.12990, 2020.

[13] Tarun Chitra. Competitive equilibria between staking and on-chain lending. arXiv preprint arXiv:2001.00919, 2019.

[14] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains. volume 9604, pages 106–125, 02 2016.

[15] Bernardo David, Peter Ga˘zi, Aggelos Kiayias, and Alexander Russell. Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 66–98. Springer, 2018. https://eprint.iacr.org/2017/573

[16] Sascha F¨ullbrunn and Abdolkarim Sadrieh. Sudden Termination Auctions—An Experimental Study. Journal of Economics & Management Strategy, 21(2):519–540, June 2012.

[17] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 281–310. Springer, 2015.

[18] Mike Hamburg. Decaf: Eliminating cofactors through point compression. In Rosario Gennaro and Matthew Robshaw, editors, Advances in Cryptology – CRYPTO 2015, pages 705–723, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. https://eprint.iacr.org/2015/673.

[19] Mike Hamburg. The STROBE protocol framework. IACR ePrint 2017/003, 2017. https: //eprint.iacr.org/2017/003 and https://strobe.sourceforge.io.

[20] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In 24th USENIX Security Symposium (USENIX Security 15), pages 129–144, Washington, D.C., August 2015. USENIX Association. [21] Isis Lovecruft and Henry de Valence. Ristretto. https://ristretto.group. Accessed: 2019.

[22] Petar Maymounkov and David Mazi`eres. Kademlia: A peer-to-peer information system based on the xor metric. In Revised Papers from the First International Workshop on Peer-to-Peer Systems, IPTPS '01, pages 53–65, London, UK, UK, 2002. Springer-Verlag.

[23] Silvio Micali. ALGORAND: the efficient and democratic ledger. CoRR, abs/1607.01341, 2016.

[24] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In 40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039), pages 120–130. IEEE, 1999.

[25] David Mills et al. Network time protocol. Technical report, RFC 958, M/A-COM Linkabit, 1985.

[26] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, Dec 2008. Accessed: 2015-07-01.

[27] Ryuya Nakamura, Takayuki Jimba, and Dominik Harz. Refinement and verification of cbc casper. Cryptology ePrint Archive, Report 2019/415, 2019. https://eprint.iacr.org/ 2019/415.

[28] Dimitrios Papadopoulos, Duane Wessels, Shumon Huque, Moni Naor, Jan V˘cel´ak, Leonid Reyzin, and Sharon Goldberg. Making NSEC5 practical for dnssec. IACR ePrint Report 2017/099, 2017. https://eprint.iacr.org/2017/099.

[29] Luis S´anchez-Fern´andez, Edith Elkind, Martin Lackner, Norberto Fern´andez, Jes´us A Fisteus, Pablo Basanta Val, and Piotr Skowron. Proportional justified representation. In Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[30] Luis S´anchez-Fern´andez, Norberto Fern´andez, Jes´us A Fisteus, and Markus Brill. The maximin support method: An extension of the d'hondt method to approval-based multiwinner elections. arXiv preprint arXiv:1609.05370, 2016.

[31] Elaine Shi. Analysis of deterministic longest-chain protocols. In 2019 IEEE 32nd Computer Security Foundations Symposium (CSF), pages 122–12213. IEEE, 2019.

[32] Alistair Stewart. Byzantine finality gadgets. Technical Report, 2018. https://github.com/ w3f/consensus/blob/master/pdf/grandpa.pdf.

[33] Gavin Wood. Polkadot: Vision for a heterogeneous multi-chain framework. White Paper, 2016.

[34] Vlad Zamfir. Casper the friendly ghost: A "correct-by-construction" blockchain consensus protocol. 2017.

[35] Alexei Zamyatin, Dominik Harz, Joshua Lind, Panayiotis Panayiotou, Arthur Gervais, and William J. Knottenbelt. XCLAIM: trustless, interoperable, cryptocurrency-backed assets. In 2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019, pages 193–210. IEEE, 2019.

A Apêndice

A.1 SREE

SPREE (Shared Protected Runtime Execution Enclaves) é uma maneira das *parachains* compartilharem código e, além disso, para que a execução e o estado deste código sejam colocados em *sandbox*. A partir do ponto de vista da *parachain* A, o quanto ela pode confiar na *parachain* B? Garantias de segurança compartilhadas de Polkadot, a execução correta do código de B com tanta segurança quanto o código de A. No entanto, se não fizermos conhecer o próprio código de B e mesmo que conheçamos o código agora, talvez o mecanismo de governança de B pode alterar o código e não confiarmos nisso. Isso muda se

soubermos um pouco do código de B, que sua governança não tinha controle, e que poderiam ser enviada mensagens por A. Então saberíamos como o código de B agiria nessas mensagens se fosse executado corretamente e, portanto, a segurança compartilhada fornece as garantias de que precisamos.

Um módulo SPREE é um pedaço de código colocado na *relay chain*, na qual as *parachains* podem optar. Este código faz parte da função de validação de transição de estado das cadeias (STVF). A execução e o estado deste módulo SPREE é isolado do resto da execução do STVF. Módulos SPREE em uma cadeia remota podem ser endereçados pelo XCMP. A distribuição de mensagens recebidas por uma *parachain* seria ela própria controlada por um módulo SPREE (que seria obrigatório para cadeias que desejam usar qualquer módulo SPREE).

Esperamos que a maioria das mensagens enviadas pelo XCMP sejam de um módulo SPREE em uma cadeia para o mesmo módulo SPREE em outra cadeia. Quando os módulos SPREE são atualizados, o que envolve colocar o código atualizado na *relay chain* e agendar um número de bloco de atualização, ele é atualizado em todas as *parachains* em seus próximos blocos. Isso é feito de forma a garantir que as mensagens enviadas por uma versão do módulo SPREE de uma cadeia para o mesmo módulo em outra nunca sejam recebidas por versões anteriores. Assim, os formatos de mensagem para tais mensagens não precisam ser compatíveis e não precisamos de padrões para esses formatos.

Para um exemplo das garantias de segurança que recebemos do SPREE, se A tiver um *token* nativo, o *token* A, gostaríamos de ter certeza de que a *parachain* B não poderia cunhar esse *token.* Poderíamos impor isso por A mantendo uma conta para B no estado de A. No entanto, se uma conta em B quiser enviar algum *token* A para uma terceira *parachain* C, então ele precisaria informar A. Um módulo SPREE para *tokens* permitiriam esse tipo de transferência de *tokens* sem essa contabilização. O módulo em A seria enviaria apenas uma mensagem para o mesmo módulo em B, enviando os *tokens* para alguma conta. B poderia então enviá-los para C e C para A de maneira semelhante. O próprio módulo responderia pelos *tokens* nas contas na cadeia B, e a segurança compartilhada de Polkadot e o código do módulo imporiam isso em B, que nunca poderia cunhar tokens A. XCMP's garantem que as mensagens serão entregues e SPREE'S garantem que elas serão interpretadas corretamente, significa que isso pode ser feito enviando apenas uma mensagem por transferência e é livre de confiança. Isso tem aplicações muito além

da transferência de *token* e significa que os protocolos de minimização de confiança são muito mais fáceis de projetar.

Partes do projeto e implementação de SPREEs ainda precisam ser totalmente projetadas. O crédito vai para o usuário do reddit u/Tawaren pela ideia inicial por trás do SPREE.

A.2 Interoperabilidade com Cadeias Externas

Polkadot hospedará vários componentes de ponte para outras cadeias. Esta seção será focada na ponte para BTC e ETH (1.x) e, portanto, revisará principalmente pontes entre cadeias *proof-of-work*. Nosso projeto de ponte é inspirado no *XClaim* [35]. A lógica da ponte terá duas partes importantes: um retransmissor (*relay*) de ponte, que entende tanto quanto possível o consenso da cadeia em ponte e um banco (possível mudança de nome por razões de relações públicas), que envolve atores que possuam *tokens* em *stake* em nome da Polkadot. O retransmissor da ponte precisa ser capaz de transportar a verificação de consenso da cadeia da ponte e verificar as provas de inclusão de transação lá. Por um lado, o banco pode ser usado por usuários na cadeia da ponte para bloquear *tokens* como suporte para o ativo correspondente que desejam receber em Polkadot, por exemplo, PolkaETH ou PolkaBTC. Por outro lado, os usuários podem usar o banco para resgatar esses ativos para *tokens* na cadeia. O retransmissor da ponte visa colocar o máximo da lógica de um cliente leve/fino de uma cadeia em ponte em uma ponte *parachain* quanto possível – pense no BTC-Relay. No entanto, criptografia e armazenamento são muito mais baratos em uma *parachain* do que em um contrato inteligente ETH. Nosso objetivo é colocar todos os cabeçalhos de bloco e provas de inclusão de certas transações da cadeia em ponte nos blocos da cadeia de pontes. Isto é suficiente para decidir se uma transação está em uma cadeia que provavelmente é final. A ideia para o retransmissor de ponte para o Bitcoin e o ETH1.0, é ter uma cadeia de ponte mais longa, onde os conflitos são resolvidos com um esquema de votação/certificação.

A.3 Comparação com outros sistemas multi-cadeia

A.3.1 ETH2.0

Ethereum 2.0 promete uma transição parcial para prova de participação e implantação de fragmentação para melhorar a velocidade e o rendimento. Existem extensas semelhanças entre os projetos de Polkadot e Ethereum 2.0, incluindo similar produção de blocos e *gadgets* de finalidade.

Todos os *shards* no Ethereum 2.0 operam como cadeias homogêneas baseadas em contratos inteligentes, enquanto *parachains* em Polkadot são *blockchains* heterogêneos independentes, apenas alguns dos quais suportam diferentes linguagens de contrato inteligente. À primeira vista, isso simplifica a implantação no Ethereum 2.0, mas “arrancar” contratos entre *shards* complica drasticamente o projeto do Ethereum 2.0. Nós temos uma linguagem de contrato inteligente existente! que existe para que o código de contrato inteligente possa ser migrado mais facilmente para ser um código *parachain*. Afirmamos que o foco inerente das *parachains* em sua própria infraestrutura deve suportar um desempenho mais alto com muito mais facilidade do que os contratos inteligentes.

O Ethereum 2.0 pede que os validadores façam *stake* de exatamente 32 ETHs, enquanto Polkadot fixa um número alvo de validadores e tenta maximizar a o *stake* de apoio com *NPoS* (consulte a Seção 4.1). Em um nível teórico, acreditamos que a aplicação de 32 ETHs resulta em validadores menos “independentes” do que no *NPoS*, o que enfraquece as suposições de segurança em todo o protocolo. Reconhecemos, no entanto, que o coeficiente de Gini importa aqui, o que dá ao Ethereum 2.0 uma vantagem inicial em “independência”. Esperamos que o *NPoS* também permita mais participação de detentores de Dots com saldos abaixo de 32 ETH.

O Ethereum 2.0 não possui um análogo exato do protocolo de disponibilidade e validade de Polkadot (consulte a Seção 4.4.2). No entanto, tivemos a ideia de usar códigos de apagamento da proposta do Ethereum [4], que visa convencer clientes leves. Validadores no Ethereum 2.0 são atribuídos a cada *shard* para atestar bloco de *shards,* como validadores de *parachain* em Polkadot constituem assim o comitê do *shards*. Os membros do comitê recebem uma prova de Merkle de um pedaço de código escolhido aleatoriamente de um nó completo do *shard* e verificam-nos. Se

todos os pedaços forem verificados e nenhuma prova de fraude for anunciada, então o bloco é considerado válido. A segurança deste esquema baseia-se em ter uma maioria honesta no comitê enquanto a segurança do esquema de Polkadot se baseia em ter pelo menos um validador honesto entre validadores de *parachain* ou verificadores secundários (consulte a Seção 4.4.2). Portanto, o tamanho do comitê no Ethereum 2.0 é consideravelmente grande em comparação com o tamanho dos validadores de *parachain* em Polkadot.

A cadeia beacon no Ethereum 2.0 é um protocolo a *proof-of-stake* como a *relay chain* de Polkadot. Da mesma forma, tem um *gadget* de finalidade chamado Casper [11, 34] como GRANDPA em Polkadot. Casper também combina eventual finalidade e acordo Bizantino como o GRANDPA, mas o GRANDPA dá melhor propriedade de vivacidade do que Casper [32].

A.3.2 *Sidechains*

Uma maneira alternativa de dimensionar as tecnologias *blockchain* é utilizando *side-chains*[7]. Estas soluções também abordam a interoperabilidade, na medida em que permitem fazer a ponte entre as *sidechains* e a cadeia principal. Por exemplo, para o Eth1.0 foram introduzidas muitas sidechains que contribuíram para a escalabilidade, como Plasma Cash e Tear[8]. Uma solução proeminente que se baseia exclusivamente na criação de ponte de cadeias independentes entre si é a Cosmos[9] que é revista e comparada a Polkadot a seguir.

A.3.3 Cosmos

Cosmos é um sistema projetado para resolver o problema de interoperabilidade da *blockchain* que é fundamental para melhorar a escalabilidade para a *web* descentralizada. Nesse sentido, há semelhanças superficiais entre os dois sistemas. Assim, a Cosmos consiste em componentes que desempenham papéis semelhantes e assemelham-se aos subcomponentes de Polkadot. Por exemplo, o Cosmos *Hub* é usado para transferir mensagens entre as zonas de Cosmos de forma semelhante à

forma como a *Relay Chain de* Polkadot supervisiona a passagem de mensagens entre as *parachains* Polkadot.

No entanto, existem diferenças significativas entre os dois sistemas. Mais importante, enquanto o sistema Polkadot como um todo é uma máquina de estado fragmentada (consulte a Seção 4.2), a Cosmos não tenta unificar o estado entre as zonas e assim o estado das zonas individuais não se reflete no estado do *Hub*. Como resultado, diferentemente de Polkadot, a Cosmos não oferece segurança compartilhada entre as zonas. Consequentemente, as mensagens de cadeia cruzada de Cosmos não são mais confiáveis. Quer dizer, que uma zona receptora precisa confiar totalmente na zona remetente para agir sobre as mensagens que recebe do remetente. Se considerarmos o sistema Cosmos como um todo, incluindo todas as zonas de maneira semelhante, analisando o sistema Polkadot, a segurança de tal sistema é igual à segurança da menor zona segura. Da mesma forma, a promessa de segurança de Polkadot garante que os dados de *parachain* validados estão disponíveis posteriormente para recuperação e auditoria (consulte a Seção 4.4.2). No caso da Cosmos, os usuários devem confiar nos operadores de zona para manter o histórico do estado da cadeia.

É notável que usando os módulos SPREE, Polkadot oferece segurança ainda mais forte do que a segurança compartilhada. Quando uma *parachain* se inscreve em um módulo SPREE, Polkadot garante que certas mensagens XCMP recebidas por essa *parachain* estão sendo processadas pelo pelo conjunto de código pré-definido do módulo SPREE. Nenhuma estrutura de confiança entre zonas semelhantes é oferecida pelo sistema Cosmos.

Outra diferença significativa entre Cosmos e Polkadot consiste na forma como os blocos são produzidos e finalizados. Em Polkadot, porque todos os estados da *parachain* estão fortemente conectados ao estado da *relay chain*, a *parachain* pode se realizar um *fork* temporariamente ao lado da *relay chain*.

Isso permite que a produção de blocos se desvincule da lógica de finalidade. Nesse sentido, os blocos de Polkadot podem ser produzidos sobre blocos não finalizados e vários blocos podem ser finalizados de uma só vez. Por outro lado, as zonas da Cosmos dependem da finalidade instantânea do estado do *Hub* para realizar uma operação de cadeia cruzada e, portanto, uma finalização atrasada interrompe as operações entre zonas.

[7]que permitem que *tokens* de uma blockchain sejam considerados válidos em uma *blockchain* independente e sejam usados lá
[8]https://loomx.io
[9]https://cosmos.network

B Glossário

Nome	Descrição	Símbolo (plural)	Definição
BABE	Um mecanismo para designar validadores eleitos aleatoriamente para produção em bloco para um determinado slot.		4.3.1
BABE Slot	Um período para o qual um bloco relay chain pode ser produzido. É de cerca de 5 segundos.	sl	4.3.1
Collator	Auxiliar os validadores na produção de blocos. Um conjunto de coladores é definido como C.	$\mathrm{c}(C)$	3.1
Dot	O token nativo de Polkadot		4.5
Validadores eleitos	Um conjunto de validadores eleitos		
Época	Um período para o qual a aleatoriedade é gerada pelo BABE. São cerca de 4 horas.	e	
Era	Um período para o qual um novo conjunto de validadores é decidido. É o cerca de 1 dia.		
Extrínseco	Dados de entrada fornecidos para a *relay chain* para transição de estados		4.2

Fishermen	Monitora a rede quanto a mau comportamento		3.1
Gossiping	Transmitir cada nova mensagem recebida para os colegas.		4.8.2
GRANDPA	Mecanismo para finalizar blocos		4.8.2
GRANDPA Round	Um estado do algoritmo do GRANDPA que leva a finalização de bloco.		4.8.2
Nomeador	Parte interessada que nomeia validadores para serem eleito. Um conjunto de nomeadores é definido como N	$n\ (N)$	3.1
NpoS	*Nominated Proof-of-Stake* - versão *PoS* de Polkadot, onde os validadores nomeados são eleitos para poder produzir blocos.		4.1
Parachain	Cadeia independente heterogênea.	P	
PJR	*Proportional-Justified-Representation* - Assegura que validadores representam tantas minorias nomeadas quanto possível.		4.1
PoV	*Proof-of-Validity* - Mecanismo onde um validador pode verificar um bloco sem ter seu estado completo.		4.4.1
Relay chain	Garante o consenso global entre as *parachains*		4.2
Runtime	O *blob* Wasm que contém a transição de estado de funções, incluindo outras operações centrais requeridas por Polkadot		4.2
Sentry nodes	Servidor *proxy* especializado que encaminha o tráfego de/para o validador.		
Session	Uma sessão é um período de tempo que possui um conjunto constante de validadores. Os validadores só podem		

	ingressar ou sair do conjunto de validadores em uma mudança de sessão.		
STVF	*State-Transition-Validation-Function* - Uma função do *Runtime* para verificar o *PoV*.		4.4.1
Validador	O partido eleito e o mais alto no comando que tem uma possibilidade de ser selecionado pelo BABE para produzir um bloco. Um conjunto de candidatos a validadores é definido como C. O número de validadores a serem eleitos é definido como n_{val}.	$\upsilon(V)$	3.1
VRF	*Verifiable-Random-Function* - Função criptográfica para determinar os validadores eleitos para a produção em bloco.		4.3.1
XCMP	Um protocolo que as *parachains* usam para enviar mensagens umas para as outras.		4.4.3