Chapter 2

Segurança e Escalabilidade em Sharding Blockchain

Antonio A. de A. Rocha, Célio V. N. de Albuquerque
Eduardo B. Loivos, Bruno T. Gondim
Arthur A. Vianna, André O. Ferreia

Abstract

The circulation of the real money, on paper, is decreasing, migrating to the virtual. As a result, the need for a secure way to carry out transactions without relying on a trusted third-party increase. The cryptocurrencies that emerged in 2009 with the Nakamoto protocol represent an alternative to this demand, however the initial proposal has a scalability problem that makes it impossible to compete with the credit card network. The need to confirm transactions means that a cryptocurrency that uses the traditional blockchain like Ethereum only has 20 to 30 transactions per second, far away from VisaNet which has around 1700 transactions per second. To solve the scalability, the concept of sharding was created. This work proposes to present some sharding models highlighting their particularities and security solutions.
Keywords*: Blockchain, Sharding, scaling, decentralized ledger.*

Resumo

A circulação do dinheiro real, em papel, está cada vez menor, migrando para o virtual. Com isso, a necessidade de uma forma segura de realizar transações, sem depender de uma terceira parte confiável, aumenta. As criptomoedas, que surgiram em 2009 com o protocolo do Nakamoto, representam uma alternativa para essa demanda. Porém, a proposta inicial possuí um problema de escalabilidade que impossibilita competir com a rede do cartão de credito. A necessidade de confirmar as transações fazem com que uma criptomoeda, que usa a blockchain tradicional como Ethereum, permita de 20 a 30 transações por segundo. Essa taxa é bem distante da VisaNet que possui por volta de 1700 transações por segundo. Para superar a limitação de escalabilidade, foi criado o conceito de sharding. Este trabalho propõe apresentar alguns modelos de sharding destacando suas particularidades e soluções de segurança.
Palavras-chaves*: Blockchain, Sharding, escalabilidade, razão descentralizada.*

2.1. Introdução

Com o surgimento da Internet, um dos pontos que apresentaram deficiência é justamente a questão relacionada à implementação de segurança. Problema este que se agravou com a quantidade de transações que emergiram do uso da rede mundial de computadores, pois aos poucos ela foi se tornando uma grande ferramenta para comprar e fazer transações. Mas, antes de adentrarmos nesse assunto, vamos fazer uma breve análise do dinheiro que conhecemos e a real motivação da Tecnologia Blockchain para os dias atuais.

A história da civilização nos conta que o homem primitivo procurava defender-se do frio e da fome, abrigando-se em cavernas e alimentando-se de frutos silvestres, ou do que conseguia obter da caça e da pesca. Ao longo dos séculos, com o desenvolvimento da inteligência, passou a espécie humana a sentir a necessidade de maior conforto e a reparar no seu semelhante. Assim, como decorrência das necessidades individuais, surgiram as trocas, que foram fundamentais para a mudança do comportamento Humano.

Esse primeiro sistema de troca direta, que durou por vários séculos, deu origem ao surgimento de vocábulos como "salário", o pagamento feito através de certa quantidade de sal, que a partir das grandes navegações, possibilitaram uma expansão territorial de trocas. Daí surgiram as primeiras moedas, tal como conhecemos hoje, feitas geralmente em metal. Com isso, surgiram então a necessidade de guardar as moedas em segurança dando surgimento aos bancos. Assim os negociantes de ouro e prata, por terem cofres e guardas a seu serviço, passaram a aceitar a responsabilidade de cuidar do dinheiro de seus clientes e a dar recibos escritos das quantias guardadas. Esses recibos (então conhecidos como "goldsmith´s notes") passaram, com o tempo, a servir como acordos através de pagamento por seus possuidores, por serem mais seguros de portar do que o dinheiro vivo. Também surgiram as primeiras cédulas de "papel-moeda", ou cédulas de banco, ao mesmo tempo em que a guarda dos valores em espécie dava origem a instituições bancárias [CasaDaMoeda 2015].

A ideia de livro-razão surgiu algum tempo depois, em 1494, desenvolvida por um padre chamado Luca Pacioli. Na sua época, esse livro consistia em promover um balanço de ativos tangíveis e intangíveis, disponibilizando de forma ordenada e detalhada várias operações de crédito e débito e a composição de um balanço final. A tecnologia Blockchain é derivada dessa ideia, sendo por vezes chamada de livro-razão digital. No livro, Summa de Arithmetica, Geometria, Proportioni et Proportionalità, escrito em 1494, o frei franciscano Luca Pacioli (1445-1517) desenvolveu os primeiros estudos de matemática para serem utilizados em contabilidade. Nesses estudos, o religioso utiliza entre outras coisas, a observação da movimentação de feiras livres com o objetivo de compreender o "Método das Partidas Dobradas", que é o sistema padrão universal de débito e crédito utilizado até hoje pelas empresas, governos e mercados mundiais e, não obstante, é estudado ainda hoje como matéria básica nos cursos de Administração e Negócios.

Seguindo a escala de modernização da Economia e da digitalização da sociedade (WEB), com o surgimento da Internet, um dos pontos que apresentaram deficiência foi justamente a questão relacionada à implementação de segurança. Problema este que se agravou com a quantidade de transações que emergiram do uso da Internet, pois aos poucos ela foi se tornando uma grande ferramenta para comprar e fazer transações. Na tentativa de mitigar tal questão, em 1971 foi criado por Horst Feisel (IBM) um algoritmo

de criptografia, denominado Lucifer, baseado em um elevado nível de segurança e uma chave de codificar e descodificar. Já em 1974 um grupo de cientistas da IBM adequou melhor a ferramenta Lucifer e surge o Data Encryption Standard "DES" (um sistema de codificação simétrico por blocos de 64 bits, dos quais 8 bits ou um byte servem de teste de paridade para verificar a integridade da chave). A principal motivação do grupo foi justamente readequar a ferramenta criada e promover maiores índices de segurança para ela. Em 1981, o DES foi adotado com o nome Data Encryption Algorithm (DEA) pela American Standard Institution com o principal objetivo de promover padronização de cifragem e procedimentos para serem utilizados em instituições financeiras. Desta forma, o DES se tornou o principal algoritmo de chave única, mas, de qualquer forma, a criação do DES não foi um sucesso absoluto para evitar fraudes e vazamentos de informações.

Ainda na constante busca de uma solução única e eficaz para promover a segurança das transações eletrônicas, em meados de 1983, David Chaum, grande cientista e entusiasta em criptografia, fundou a DigiCash, uma empresa de moeda eletrônica criptografada. Seu produto foi chamado de E-cash e seu objetivo era que os usuários obtivessem moedas digitais de um banco e não pudessem ser rastreados nem pelo banco nem por qualquer outra pessoa. Tal invenção foi precursora do movimento Cypherpunk, que teve seu início no final da década de 1980, já propondo sistemas *peer-to-peer*. Naquela mesma ocasião, um dos sócios da empresa DigiCash, Nick Szabo, formado em Direito e criptógrafo, promoveu a pesquisa relativa a contratos digitais e moeda digital, denominada "Contratos Digitais". Em 2005, Szabo desenvolveu um mecanismo inovador para tratamento de uma moeda digital que recebeu o nome de Bit Gold, algumas bibliografias se referem a ela como a precursora do Bitcoin.

Finalmente, em 2008, foi lançada a moeda virtual Bitcoin, que se utiliza da plataforma Blockchain para suas transações. O lançamento da moeda foi feito por Nakamoto Satoshi, um pseudônimo, pois até os dias de hoje não se sabe precisamente quem foi o criador da moeda virtual Bitcoin. Não é sabido nem mesmo se trata-se de uma pessoa somente ou uma equipe de cientistas que desenvolveu a tecnologia [Fernando Wosniak Steler 2017].

2.1.1. Então, o que é Blockchain?

A expressão "*In Blockchain We Trust*" vem sendo utilizada para ressaltar a confiança na segurança dessa solução tecnológica. Isto porque, nela as transações entre indivíduos e a transferência de valores é garantida por meio de algorítimos matemáticos e criptográficos. A Blockchain é uma tecnologia de núcleo que possibilita que grandes grupos de pessoas cheguem a um acordo (também definido como, consenso) e registrem transações permanentes, sem uma autoridade central.

O termo (Bloco = block + chain = cadeia) tem origem no seu funcionamento, onde cada bloco validado é criptograficamente selado ao bloco anterior, formando uma cadeia de blocos cada vez maior, diretamente relacionado na construção de uma economia digital justa, inclusiva, segura e democrática. Por isso, a Blockchain é descrita como uma máquina de confiança, não é apenas um *ledger* distribuído em larga escala, mas também como uma trilha de auditoria imutável, onde cada bloco é incorporado em todos os seguintes, impossibilitando a alteração da história de seu conteúdo.

A tecnologia Blockchain também é, portanto, um banco de dados distribuídos,

sendo praticamente invulnerável a falhas e adulterações, e suas múltiplas utilidades descolam-se da tecnologia de Criptomoedas Bitcoin – para a qual foi criada. Antes do desenvolvimento da tecnologia Blockchain, os registros contábeis eram mantidos em bancos de dados centralizados e não públicos. As pessoas precisavam confiar na idoneidade do banco de dados para ter certeza de que não haveria nenhuma alteração nos registros (saldos e transações da conta). Com a Blockchain, os dados são distribuídos entre todos os participantes, com total transparência e descentralização. Logo, torna-se desnecessário confiar em uma terceira parte para que os dados contábeis sejam registrados corretamente e não haja perigo de fraudes. Podemos concluir que o modelo de solução Blockchain assemelha-se a um "livro-razão", ou seja, uma base de informações com entrada de diversos dados e transações. Todas as informações imputadas e contidas na ferramenta são compartilhadas entre vários usuários.

O processamento desta base de dados é feito em blocos, "de tempos em tempos", criando um código de verificação a cada bloco processado. Estes códigos de verificação são criados com base nos blocos processados anteriormente, fazendo com que a Blockchain seja uma solução de alta confiabilidade, pois, uma vez adulterado um bloco, isso impactará nos demais blocos processados.

Idealmente utiliza-se Blockchain em soluções que temos desconfiança mútua pelas partes envolvidas, são exemplos delas: Estabelecimento de contratos; Registro de propriedade intelectual; Estabelecimento de Identidade digital; Prontuário médico; Cartórios digitais; Operações cambiais imediatas; Sistema de voto digital; e, Auditabilidade [REVOREDO 2019].

2.2. Blockchain Tradicional

A *Blockchain* oferece propriedades que proveem benefícios às aplicações e sistemas baseados nesta tecnologia. Conforme definido em [Rebello et al. 2019], as principais propriedades da *Blockchain* são:

- **Descentralização.** A *Blockchain* é executada de maneira distribuída, sem a necessidade de um intermediário confiável para a troca de ativos, através do estabelecimento de consenso entre todos os participantes da rede;
- **Imutabilidade.** Os dados armazenados em uma corrente de blocos são imutáveis. Não é possível modificar ou recriar qualquer dado incluído na corrente de blocos. Toda atualização na corrente de blocos é realizada de forma incremental;
- **Irrefutabilidade.** Os dados são armazenados na corrente de blocos em forma de transações assinadas, que não podem ser alteradas devido à propriedade de imutabilidade da corrente de blocos. Portanto, o emissor de uma transação jamais pode negar sua existência;
- **Transparência.** Todos os dados armazenados na *Blockchain* são acessíveis por todos os participante da rede. Permitindo que todos os participantes possam verificar, auditar e rastrear os dados inseridos na corrente de blocos para encontrar possíveis erros ou comportamentos maliciosos;
- **Disponibilidade.** As correntes de blocos são estruturas replicadas em cada participante da rede e, portanto, a disponibilidade do sistema é garantida mesmo sob

falhas, devido à redundância de informações;

- **Anonimidade.** Os usuários e nós mineradores de uma *Blockchain* são identificados por chaves públicas ou identificadores únicos que preservam suas identidades. Ainda, é possível utilizar uma chave pública em cada transação, evitando a rastreabilidade do usuário e conferindo um grau a mais de anonimidade.

2.2.1. Função Hash

A função hash é uma função matemática que tem como entrada uma string de tamanho variável e a mapeia em uma string de tamanho fixo. A conversão tem como objetivo gerar uma identidade única, resistente a colisões. Uma função hash é resistente à colisão, se houver baixa probabilidade de encontrar dois valores de entrada distintos, que possuam um mesmo valor de saída (Figura 2.1). Dado que mapear uma grande quantidade de dados para um universo menor, quanto menor a probabilidade de encontrar hash iguais de forma proposital, melhor a função de hash.

Figure 2.1. Exemplo de colisão entre dois hashs

Uma característica importante do hash é que, uma pequena variação na entrada resulta em uma grande variação na saída. Como pode ser visto na Figura 2.2, mudar uma uma letra de maiúscula para minúscula gera uma saída completamente diferente. Não deve ser possível adicionar conteúdo “a” um texto sem modificar sua saída. Se uma função H(a) resulta a string fixa “y”, encontrar uma string “b” tal que concatenada com “a” resulte em H(a || b) = y, em um curto espaço de tempo, não deve ser possível. Da mesma forma, se uma entrada tem uma parte gerada de forma aleatória, do hash resultante não pode ser escolhido um hash específico. Outra propriedade da função hash é: Dada uma saída, não é possível determinar a entrada que a originou. Em um espectro pequeno ou repetido de entradas, uma vez vista a solução de para uma dessas entradas, passa a ser possível identificá-la através de seu hash. É possível ocultar a entrada usando um nonce, ou um valor secreto, concatenado a entrada original.

Figure 2.2. Exemplo hash usando SHA256

2.2.2. Estrutura de bloco

A estrutura de bloco consiste principalmente em um conjunto de dados. No universo das criptomoedas, os dados de um bloco representam as transações que ocorrem em um

determinado período de tempo. Além dos dados, um bloco possui seu índice ou altura, e um campo de nonce. Uma vez formado um bloco, uma função de hash é usada para gerar uma assinatura para o mesmo, garantindo que os dados registrados nele não serão alterados. Uma função de hash comum para esta finalidade é a SHA de 256 bits usada pelo Bitcoin.

Ao criar um bloco e assiná-lo, os dados são verificados e apenas as transações válidas entram em um bloco. Para dificultar que um usuário mal-intencionado valide um bloco com dados forjados é necessário um esforço para assinar o bloco. As formas mais comuns de esforço usadas são *Proof of Work (PoW)* e *proof of stake (PoS)*, mais detalhadas à frente.

2.2.3. Consenso

Plataformas Blockchain usam consenso descentralizado para manter a consistência em uma máquina de estado distribuída. Esta pode ser usada para realizar pagamentos completamente descentralizados ou mesmo cálculos de Turing completos, tornando realidade a criação de aplicações descentralizadas. É papel do consenso em sistemas de Blockchain garantir que todos os nós confiáveis em uma rede Blockchain executem as mesmas atualizações de estado na mesma ordem [Muratov et al. 2018].

Há dois tipos bem comuns de algoritmo de consenso, os baseados em prova e os em voto. No conceito básico de algoritmo de consenso baseado em prova, entre os muitos nós que se juntam à rede, o nó que executa a prova terá o direito de acrescentar um novo bloco à corrente e receber a recompensa. Os principais algoritmos com base em prova são *Proof of Work* , *Proof of Stake* e alguma variação desses [Pahlajani et al. 2019]. Já os algoritmos de consenso baseado em votação, além de manter o livro-razão, todos os nós da rede teriam que verificar juntos as transações ou blocos. Eles se comunicam uns com os outros, antes de decidir anexar ou não, os blocos propostos à cadeia. Dentro de quase todas essas variantes, os nós necessitam que um número mínimo de nós aceitem o mesmo bloco proposto para anexá-lo à cadeia [Pahlajani et al. 2019].

Proof of Work (PoW)

O uso de um sistema como PoW impede que usuário utilize seu poder computacional para gerar um spam de determinada informação. No cenário da Blockchain, o PoW impede que diversos blocos sejam criados sequencialmente pelo mesmo usuário. O PoW envolve a busca de um valor de nonce que quando processado por um hash, como com SHA-256, o *output* começa com um determinado número de bits zero. O trabalho médio necessário é exponencial, de acordo com número de bits zero necessários, e pode ser verificado executando um único hash. [Nakamoto 2008]

Proof of Stake (PoS)

Normalmente pergunta-se, se nós devem manter o consumo de energia para ter uma criptomoeda descentralizada. Portanto, demonstrar que a segurança de criptomoedas ponto a ponto não precisa depender de consumo de energia, é um marco importante tanto teórico

quanto tecnologicamente [King and Nadal 2012]. O PoS aproveita o fato de que a posse de moeda é proporcional ao tempo que um nó participa da rede. Desta forma demonstra a sua confiabilidade e assegura a honestidade do mesmo.

Proof of Elapsed Time (PoET)

O consenso PoET atua como uma loteria, na qual, cada nó deve esperar um período de tempo randômico, e o primeiro a concluir este tempo de espera ganha o bloco. O próprio nó deve gerar um valor de tempo pelo qual ele irá ficar em *sleep mode*, o primeiro nó a despertar, ou seja, aquele que tiver o menor período de tempo de espera, irá fazer o commit do bloco para a Blockchain e realizar o brodcast das informações relevantes [Jake Frankenfield 2020]. Geralmente, alguma prova de que ele não adulterou o processo. Para garantir que o nó não gere valores de tempo pequenos para sempre ganhar o bloco, os nós que participam deste consenso precisam executar esta geração de valores em um *TEE(Trusted Execution Environment)*. TEEs são ambientes de execução que têm uma superfície de ataque extremamente pequena e são equipados com procedimentos de verificação criptográfica que podem fornecer atestados verificáveis externamente e evidências de adulteração [Corso 2019].

Como o PoET foi desenvolvido pela Intel em 2016, ele foi pensado para ser usado juntamente com a tecnologia TEE da Intel, o SGX [IntelSGX 2021], tecnologia essa que permite ter áreas seguras no processador, para proteger código e dados selecionados contra modificações. Desta forma, o tempo de espera será gerado por um código que esteja rodando numa área segura do processador. Depois de esperar o tempo necessário, o nó receberá um certificado confirmando que ele esperou o tempo que havia sido sortead. Este certificado pode ser verificado, pois TEEs fornecem atestados verificáveis. Sendo, então, possível verificar que o nó realmente esperou o tempo proposto. Por fim, como mencionado anteriormente, o nó que ganhar o bloco fará o broadcast juntamente com informações relevantes, neste caso, o certificado [Centieiro 2021]. Portanto, este consenso destaca-se pela justiça, uma vez que funciona como uma loteria, e pelo baixo consumo de recursos e energia, uma vez que não é feito nenhuma espécie de processamento pesado e os nós ficam em *sleep mode* enquanto esperam.

Consenso baseado em voto

Nesse tipo de consenso, todos os nós mineradores devem votar pela aprovação ou rejeição de um bloco, e atingir um consenso antes de adicionar o novo bloco à corrente. Como consequência, todos os nós mineradores devem ser conhecidos e identificados. O protocolo de consenso garante que a adição de um bloco à corrente ocorre de forma sincronizada para todas as réplicas .Isto é, as réplicas adicionam a mesma sequência de blocos na corrente.

Dentre os modelos baseados em voto, os tolerantes a falhas bizantinas (Byzantine Fault Tolerant - BFT) são particularmente interessantes. A ideia principal dos protocolos BFT é eleger um líder por uma rodada, que determina e propõe um novo bloco aos participantes do consenso. Os protocolos BFT promovem uma camada de confiança a mais

em comparação a protocolos tolerantes apenas a falhas por parada, pois toleram comportamento malicioso na rede. No entanto, geralmente necessitam de mais réplicas, quando comparado com os protocolos que toleram falhas apenas por parada, para tolerar a mesma quantidade de falhas [Rebello et al. 2019].

2.2.4. Cadeia de blocos

A função hash garante que qualquer alteração em um bloco seja facilmente detectada. Para garantir a integridade dos dados é necessário que um bloco esteja ligado ao próximo através do hash. Assim, para alterar um bloco deve-se alterar todos os blocos posteriores. Os blocos são conectados através de um campo adicional contendo o hash do bloco anterior. O primeiro bloco, conhecido como bloco gênesis, possui esse campo com o valor 0 em todos os caracteres.

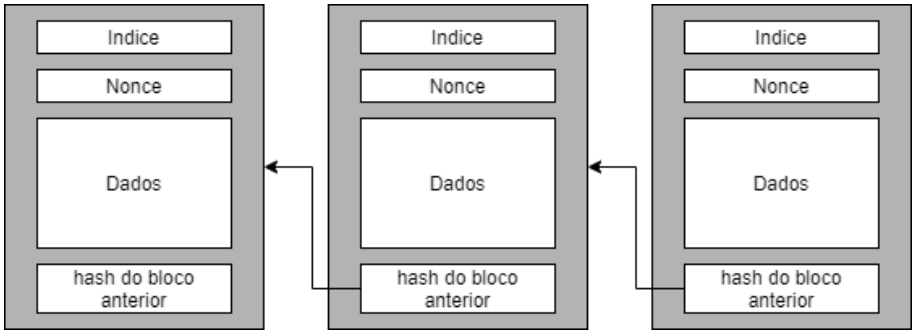

Figure 2.3. Modelo *Blockchain*

Todos os usuários possuem uma cópia da *Blockchain*, e sempre que um minerador confirma um bloco ele transmite sua solução na rede. Devido a latência ou a ação de um nó mal intencionado, duas soluções podem ser aceitas em partes diferentes da rede gerando um *fork* na *Blockchain*. Uma vez identificado o fork os nós consideram a cadeia mais longa como a verdadeira. Logo para falsificar os dados de um bloco da cadeia, um usuário malicioso precisa, no caso de prova de trabalho, de poder computacional suficiente para tornar seu fork mais longo que a *Blockchain* verdadeira.

2.2.5. Bitcoin x Ethereum

Como o protocolo Bitcoin é "Open Source", qualquer um poderia pegar seu protocolo, bifurcar (modificar o código) e iniciar sua própria versão de dinheiro eletrônico. Essa qualidade da Blockchain Bitcoin possuir um código aberto que contribuiu para que, ao longo dos anos, o protocolo Bitcoin fosse modificado centenas de vezes para criar versões alternativas do Bitcoin que são mais rápidas ou mais anônimas. Percebeu-se que o protocolo Blockchain subjacente possibilitava que pessoas desconhecidas, ou que não confiavam entre si, realizassem qualquer tipo de transação de valor, e não apenas dinheiro, sem quaisquer intermediários.

Começam a surgir, então, projetos que buscavam usar a tecnologia Blockchain para transferências, sem intermediários, de outros tipos de valor. Também ganhou corpo a ideia de se afastar da blockchains de propósito único, para criar um protocolo onde qualquer tipo de transação, sem validadores tradicionais de confiança, fosse possível. En-

tão, ao perceber que as adaptações da Blockchain Bitcoin não eram satisfatoriamente eficientes nem flexíveis, Vitalik Buterin introduziu a ideia de dissociar as funcionalidades blockchain e iniciou o projeto da Blockchain Ethereum [Wood et al. 2014] em 2014.

Ao contrário da Blockchain Bitcoin, que é uma Blockchain de propósito único com um único contrato inteligente, a Blockchain Ethereum é projetada como uma rede de computadores descentralizada na qual qualquer tipo de contrato inteligente pode ser programado, permitindo qualquer tipo de troca direta de valor. Outras Blockchains surgiram como solução, pois o surgimento da Ethereum inspirou projetos de Blockchain mais recente (como NEO, EOS, CARDANO, CHAINLINK, QTUM, STELLAR, GOCHAIN, entre outros. Além do aspecto tecnológico, fatores técnicos, econômicos e legais também serão relevantes para avaliar a viabilidade de uma Blockchain [REVOREDO 2019].

2.2.6. Casos de uso e aplicações da Blockchain

De acordo com o site (http://medium.com) entre 2020 e 2025, nosso mundo será povoado por uma nova espécie de 50 bilhões de dispositivos conectados, 100 milhões de robôs e 20 milhões de veículos elétricos. Isso também é confirmado pelo fórum econômico mundial, onde eles assumem que 30% da Contabilidade, Aquisições e Liquidação serão feitas por blockchain e AI. 10% do PIB será baseado em tecnologias de blockchain até 2025 e a sociedade estará cercada por uma nova espécie de máquinas autônomas, carros autônomos, IAs para tomada de decisão e robôs de edição de DNA CRISPR.

Acredita-se que as máquinas serão os clientes do futuro presumindo que mais cedo ou mais tarde, terão carteiras integradas, as primeiras tentativas nesse sentido já foram feitas e com o Ethereum, é possível implantar carteiras com contratos inteligentes mas embora a aplicação como moeda e sistema financeiro sem intermediários seja a aplicação mais famosa da tecnologia de blockchain, várias outras possibilidades e aplicações podem ser vislumbradas., ainda que tenha algumas desvantagens tecnológicas, a tecnologia traz várias possibilidades. [Eichmann 2018]

2.2.6.1. O que é DAO?

Uma Organização Autônoma Descentralizada, ou DAO, é uma organização ou empresa teórica operada por código em vez de pessoas. Os DAOs criam uma maneira de as organizações ou empresas serem estruturadas de forma menos hierárquica, argumentam os defensores, com os investidores direcionando diretamente a direção das empresas, em oposição aos líderes designados.

Os defensores do DAO acreditam que o Ethereum pode dar vida a essa ideia futurista. Ethereum é a segunda maior criptomoeda por capitalização de mercado e é a maior plataforma para usar a tecnologia por trás da criptomoeda - blockchain - para usos além do dinheiro. A ideia é que, se o bitcoin pode eliminar os intermediários nos pagamentos online, a mesma tecnologia ou uma tecnologia comparável pode fazer o mesmo pelos intermediários nas empresas? E se organizações inteiras pudessem existir sem um líder central ou CEO comandando o show?

Com isso erá possível criar múltiplas cópias de segurança: Redundância de infor-

mações é algo fundamental para evitar perdas de dados e nada tem mais backups do que algo controlado por um blockchain, afinal, cada um dos mineradores e até mesmos outros membros da rede (como o caso dos fullnodes do bitcoin) possuem uma cópia completa, atualizada e integral de todos os blocos e, portanto, todas as informações registradas por ele como:

- Registros virtualmente inalteráveis: o hash que valida um bloco é formado pelas informações registradas em um bloco e hash do bloco anterior, que foi gerado pelos dados dele e seu antecessor, e por aí vai. Assim sendo, as informações em um blockchain são incrementais e acumulativas, armazenadas de forma a não ser alteradas ou apagadas.

- A autenticidade de documentos a garantir a autenticidade de documentos é algo custoso, especialmente no Brasil, e ser caro não garante que o processo é à prova de falhas ou fraudes, pois sempre que houver um fator humano que possa ser corrompido e subverter o sistema, a fraude será possível. A startup brasileira OriginalMy surgiu com uma proposta: registrar documentos em um Blockchain, beneficiando-se de, pelo menos, duas características importantes: o fato de que o registro do documento não possa ser alterado ou removido e a transparência que um blockchain público provê, permitindo a verificação deste registro sem burocracias.

Também será possível uma maior Proteção dos direitos autorais, caso um sistema notarial de algum país fosse implementado integralmente em um único blockchain, além de mitigar fraudes por adulteração ou remoção de informações, todos os documentos poderiam ser verificados e confrontados, evitando a fraude do "vidente que registra sua previsão em cartório": algumas pessoas fazem dezenas de previsões aleatórias como "prever" celebridades mortas em acidentes aéreos, por exemplo. Ao registrar tais previsões em dezenas de cartórios diferentes (cada previsão em um cartório diferente), quando o previsto se torna um fato, basta apresentar a previsão que "acertou" o acontecimento, desprezando todas as demais.

Algo semelhante aconteceu na Copa do Mundo de 2014, quando uma conta de Twitter chamada "@fraudefifa" quis provar que a FIFA manipulava os resultados da competição. Curiosa, no entanto, foi a maneira pela qual a conta decidiu "provar" sua tese, realizando postagens com todas as possibilidades de confrontos e vencedores, dias antes das partidas. Quando dois times realmente jogavam e um destes ganhava, bastava remover do Twitter todas as postagens desfavoráveis.

Surgiram outras iniciativas de autenticação de documentos na Internet, como é o caso do site LexisNexis, que também registra os documentos de maneira a comprovar sua autoria futura. Nesse contexto, a gigante Kodak decidiu não investir na tecnologia em razão da grande receita que a empresa adquiria na revelação de filmes fotográficos tradicionais.

A ideia é criar uma plataforma de gerenciamento de direitos de imagem da qual os fotógrafos possam registrar seus trabalhos em um blockchain, que impedirá que tal registro seja alterado ou apagado. Desta maneira, o registro da foto em um blockchain

poderia comprovar um eventual plágio fotográfico, permitindo identificar o autor da obra com facilidade.

Em 2018, pela primeira vez, na França, o Carrefour usou a tecnologia blockchain com uma de suas linhas icônicas de produtos animais: o frango Carrefour Quality Line Auvergne, do qual um milhão é vendido todos os anos , um marco importante para seu plano de transformação chamado Carrefour 2022. A tecnologia blockchain já é utilizada para frangos de linha livre do Carrefour Quality Line Auvergne e será utilizado em mais oito linhas de produtos de origem animal e vegetal, como ovos, queijo, leite, laranja, tomate, salmão e bife moído. Um sistema inovador projetado garante aos consumidores uma completa rastreabilidade do produto.

Existem estudos de Implementação do Smart Governance, votação 2.0, pois além de garantir a privacidade, o contrato traz transparência ao processo de apuração pois, ao verificar as condições para as quais os votos foram apurados, é possível garantir que 1 pessoa = 1 voto. Ao utilizar um blockchain como infraestrutura para que o sistema de votação seja realizado, reduzem - se drasticamente quaisquer fraudes que possam hoje ser realizadas em sistemas eleitorais tradicionais.

A criação de contratos inteligentes trouxe consigo uma nova proposta de arquitetura para aplicações, chamada de DAPP. DAPP significa decentralized application, ou aplicações descentralizadas. Em DAPP, utilizada em aplicativos de smartphones para as mais diferentes necessidades. as aplicações possuem parte de sua programação em frontend (instalados nos smartphones, como geralmente o são), mas a programação mais pesada, conhecida como backend, roda em um blockchain em uma rede descentralizada P2P. Desta forma, as DAPPs se tornam verdadeiras aplicações compartilhadas não sendo possível, nem aos próprios desenvolvedores, modificar condições e regras já estabelecidas.

As blockchains e smart contracts são tecnologias em ascensão. Enquanto os ativos digitais são usados por uma parcela muito pequena da população mundial, a adoção da blockchain é inicial em outros campos e ainda faltam exemplos de grandes cases de uso da tecnologia. No entanto, trata-se de algo que sequer completou dez anos de existência. A cada dia que se passa, novas iniciativas são anunciadas e muitas delas prometem revolucionar os mercados nos quais estão inseridas. Não deixe de acompanhar esta tecnologia que será uma das forças mais transformadoras da próxima década. [Eichmann 2018]

2.3. Escalabilidade na Blockchain

Escalabilidade é um termo utilizado em sistemas, que diz respeito à capacidade de um sistema crescer, tendo como intenção atender mais usuários ou adicionar mais funcionalidades. Sendo assim, um sistema é dito escalável quando o seu desempenho aumenta proporcionalmente com o seu poder computacional.

Podemos dividir escalabilidade em duas vertentes, escalabilidade horizontal e escalabilidade vertical. Na escalabilidade horizontal aumentamos o poder computacional do sistema adicionando nós ao sistema, ou seja, adicionando uma nova máquina. Já na escalabilidade vertical aumentamos o poder computacional do sistema melhorando um nó existente, como por exemplo, adicionando mais memória RAM. Porém há sistemas

em que o aumento no poder computacional não acarreta um aumento no desempenho do sistema, nestes casos é necessário rever a arquitetura.

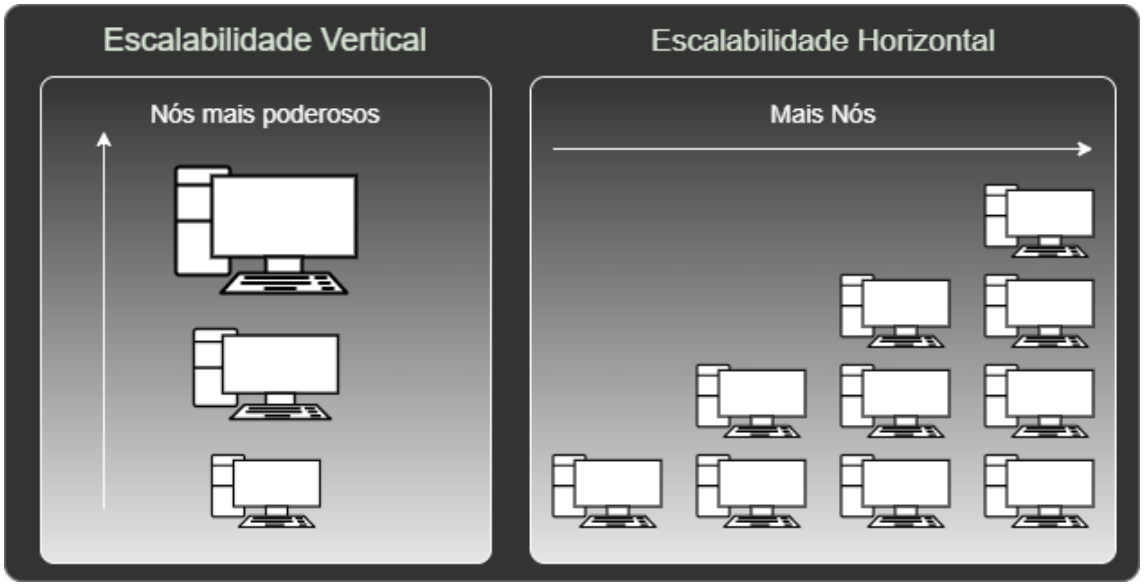

Figure 2.4. Tipos de Escalabilidade

A classe que foi chamada de *Blockchain* Tradicional, apresentada na seção anterior, apresenta problema de escalabilidade, no que diz respeito ao tempo para confirmar uma transação e vazão (quantas transações são confirmadas por segundo) quando comparado a sistemas de cartões de crédito. Que por sua vez são capazes de processar mais de duas mil transações por segundo. Nos sistemas de cartões de crédito, a escalabilidade pode ser tanto vertical, quanto horizontal, pois trata-se de um sistema centralizado. Portanto, a entidade central pode analisar e investir recursos para melhorar o desempenho da forma que julgar conveniente.

Como a *Blockchain* é uma rede P2P, a sua escalabilidade vai se dar principalmente da forma horizontal, no qual novos validadores(mineradores), vão se juntar à rede e o poder computacional do sistema irá aumentar. Se tratando de prova de trabalho, o modelo que é diretamente afetado pelo poder computacional, aumentar a capacidade de um nó não gera ganho para a rede, apenas uma vantagem para este nó em detrimento dos demais. Para os demais modelos essa desigualdade é irrelevante para a escalabilidade, uma vez que um nó possui apenas um voto ou uma maior probabilidade de ser sorteado.

Nas *Blockchains* Tradicionais, apesar de termos esse aumento de poder computacional, nós não temos um aumento proporcional no desempenho do sistema. No caso do Bitcoin, nem mesmo há um aumento, o desempenho da rede se mantém constante, independentemente do poder computacional, na Figura 2.5, temos um comparativo do Bitcoin, que é um sistema que não escala, com uma *Blockchain* ideal, que escala.

2.3.1. Por que a *Blockchain* Tradicional não escala?

A partir da figura 2.5 acima levanta-se um questionamento sobre o porque a *Blockchain* Tradicional não escala. Para responder esta pergunta devemos observar as decisões de

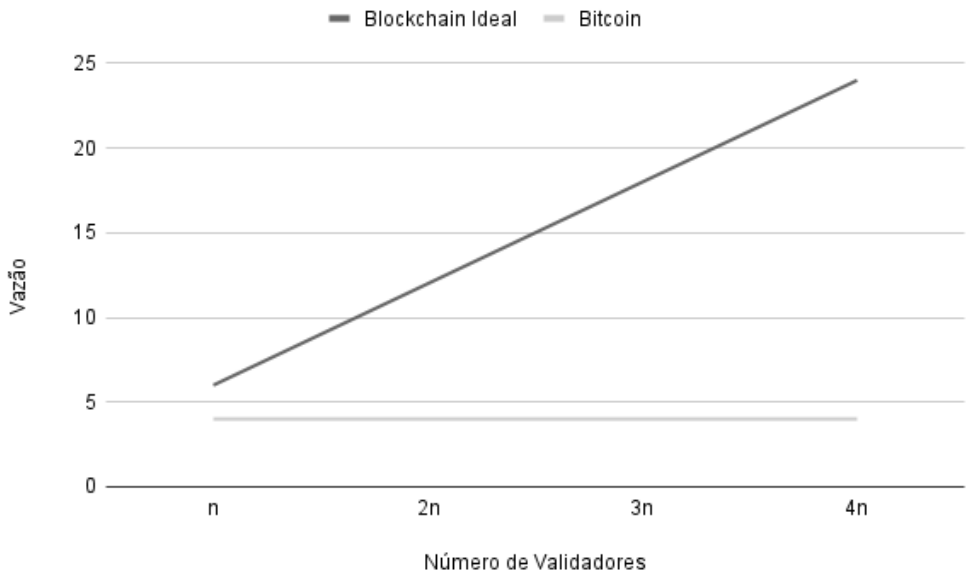

Figure 2.5. Comparativo Bitcoin

arquitetura desta classe de *Blockchain*. Decisões na arquitetura de sistemas são feitas com base em tradeoff. Tratando-se de *Blockchain* e performance, devemos olhar basicamente três pontos principais: a quantidade de blocos que cada nó armazena, o algoritmo de consenso e o tamanho dos blocos da cadeia. Para a análise dos pontos mencionados teremos como caso de estudo a Bitcoin, dado que, outras *Blockchains* que se encaixam na categoria de *Blockchain* Tradicional foram muito influenciadas por ela.

O primeiro ponto diz respeito à quantidade de blocos que cada nó armazena, ou seja, o quanto da cadeia de blocos cada nó guarda. No caso da Bitcoin cada nó armazena toda a cadeia de blocos, isto trará como benefícios, uma maior disponibilidade das informações, uma vez que ela está replicada em todos nós da rede. E desvantagens como uma maior dificuldade de manter a coerência destes dados e um desafio de performance, pois toda vez que for feita uma consulta estaremos consultando a base de dados completa.

Como segundo ponto temos o algoritmo de consenso. Como mencionado anteriormente, o algoritmo de consenso é responsável por manter a consistência dos dados armazenados pelos nós da rede. Essa tarefa se torna consideravelmente mais lenta uma vez que cada nó armazena a cadeia completa. No caso da Bitcoin o algoritmo de consenso utilizado é o Proof of Work, que é executado por máquinas que são chamadas de mineradores.

Ainda sobre o segundo ponto, o próprio algoritmo de consenso Proof of Work é considerado um problema. Neste caso, ele funciona como um puzzle computacional no qual não há atalho. Deve-se simplesmente buscar por um valor que faça parte do conjunto de soluções do puzzle, e esta ideia que faz com que o algoritmo de consenso funcione mas também é o seu maior problema, pois acabamos tendo um desperdício de poder computacional.

Quando um minerador na Bitcoin descobre a solução para uma prova, e ele envia uma mensagem *Broadcast* para avisar aos outros nós da rede que aquela prova foi resolvida e o bloco pode ser fechado. Entretanto existe um tempo de propagação dessa mensagem, e enquanto isso um outro nó pode encontrar uma outra solução para a prova. A partir daí podemos acabar gerando duas cadeias de blocos diferentes, dado que os mineradores podem pegar um conjunto de transações diferentes para tentar formar um bloco, este cenário é chamado de *fork*. É importante ressaltar que, o fork pode ser enxergado como uma anomalia, pois a existência de nós com cadeias de blocos diferentes vai contra a ideia de consenso. Esta situação é resolvida simplesmente adotando a maior cadeia como a correta. Assim temos novamente um desperdício computacional, pois, apenas um *fork* é escolhido para representar a cadeia, desperdiçando todo o poder computacional investido na construção dos outros *forks*.

A respeito do terceiro ponto temos que, na Bitcoin foi definido um tamanho máximo de 1MB para os blocos, este tamanho para o bloco parecia viável quando a Bitcoin foi concebida. Com o passar dos anos a criptomoeda se popularizou muito e por conta disso a quantidade de transações aumentou muito, fazendo com que haja uma demora ainda maior ao confirmar as transações. É necessário que um bloco seja fechado contendo todas as transações, e como os blocos são muito pequenos se comparados ao volume de transações disponíveis, acaba formando-se um grande volume de transações à espera de um bloco para incorporá-las.

Com base nos pontos discutidos, é possível entender porque a Bitcoin não escala. Nesta temos um atraso de confirmação das transações, que se dá justamente por conta do tempo necessário para os mineradores solucionarem o puzzle do Proof of Work. Também por conta do tamanho dos blocos. Esse tempo de confirmação é maior que 10 minutos. Além da demora para a primeira confirmação da transação, temos a situação dos *forks*, que podem fazer com que o *fork* que já tinha processado determinada transação seja descartado em detrimento de um que ainda não a processou.

No fim das contas a Bitcoin só é capaz de processar apenas aproximadamente 4 transações por segundo[L. 2019]. Isso destoa muito se comparado a um sistema de cartão de crédito que atualmente é capaz de processar aproximadamente 1700 transações por segundo[L. 2019], este nível de vazão de sistema é também chamado de padrão Visa. Além de uma vazão muito maior sistemas de cartão de crédito também possuem um tempo de confirmação de transação menor, apresentando então, alta vazão e baixa latência. Tratando-se de criptomoedas e *Blockchain* tem-se como meta um desempenho tão bom quanto o dos sistemas de cartões de crédito.

2.3.2. *Sharding*

À medida que a popularidade da Blockchain cresce, o mesmo acontece com o volume transacional gerenciado pela rede e a carga de trabalho é ajustada para manter a dificuldade. Se pensarmos em uma Blockchain como um banco de dados compartilhado, à medida que mais e mais dados são adicionados à rede, precisa-se encontrar novas maneiras de processar todos esses dados com eficiência e rapidez. É aí que o sharding pode ajudar.[Mearian 2019]

O *sharding* não é uma técnina nova, ele foi originalmente desenvolvido para mel-

horar a performance de bancos de dados muito grandes, e somente, recentemente, passou a ser explorado como solução para escalabilidade na Blockchain. A técnica consiste na fragmentação ou divisão horizontal de bancos de banco de dados, permitindo que processem mais transações por segundo. O *sharding* divide o sistema em partições menores, conhecidas como "*shards*". Cada fragmento é composto por seus próprios dados, tornando-o distinto e independente quando comparado a outros fragmentos, permitindo um melhor manuseio dos mesmos, tornando-os menos pesados e mais fáceis de operar.

O *sharding* pode ajudar a reduzir a latência ou lentidão de uma rede, pois divide uma rede *Blockchain* em fragmentos independentes. No entanto, existem algumas questões de segurança em torno do *sharding* que podem ser exploradas por adversários. Ao subdividir os nós em *shards*, o poder de *hashing* de cada grupo diminuirá consideravelmente. Isso gera um problema de segurança ao permitir que um agente mal-intencionado execute um ataque com mais facilidade. Uma situação que coloca em risco a segurança e a integridade das informações.[bit2me 2020]

Figure 2.6. Tradeoff Sistemas Distribuídos

Devemos ter em mente o tradeoff que estamos fazendo ao implementar uma *Blockchain* que faz uso de *sharding*, pois como ilustrado na Figura 2.6, é impossível atingir os três extremos. O *sharding* por sua vez, tende à escalabilidade e descentralização, introduzindo assim, desafios de segurança à *Blockchain*. Temos então, dois principais desafios introduzidos pelo *sharding* [Wang et al. 2019]:

1. Distribuir os nós de maneira uniforme nos *shards*, de forma a garantir que a maioria deles seja honesta com alta probabilidade;

2. Como garantir que um adversário não obtenha vantagem significativa, enviesando as operações ou criando identidades Sybil, que é quando um único nó consegue burlar a política de identidade e se passar por vários simultaneamente [Douceur 2002].

Outro problema inserido na *Blockchain* tradicional com a implementação de partições ocorre porque os nós, quando atribuídos a um subgrupo, não têm acesso ao saldo dos nós que pertencem aos outros subgrupos. Tornando necessário criar um protocolo para resolver as transações entre os subgrupos.

Portanto, para uma *Blockchain* se manter segura e funcional, enquanto faz *sharding*, ela deve, em geral, ser composta por cinco componentes [Wang et al. 2019], que são:

1. **Identity establishment and committee formation**: Para se juntar ao protocolo, cada nó precisa estabelecer uma identidade, como uma chave pública, um endereço IP e uma solução de prova de trabalho (PoW). Cada nó é então atribuído a um comitê correspondente à sua identidade estabelecida. Nesse processo, o sistema precisa prevenir a identidade Sybil. No entanto, uma blockchain permissionada não requer este processo;

2. **Overlay setup for committees**: Uma vez que os comitês são formados, cada nó se comunica para descobrir as identidades de outros nós em seu comitê. Para uma blockchain, uma sobreposição de um comitê é um subgrafo totalmente conectado contendo todos os membros do comitê. Normalmente, este processo pode ser feito com um protocolo de fofoca(*gossip protocol*);

3. **Intra-committee consensus**: Cada nó dentro de um comitê executa um protocolo de consenso padrão para concordar com um único conjunto de transações. Nesse processo, todos os membros honestos devem concordar com o bloco proposto em seu comitê;

4. **Cross-shard transaction processing**: A transação deve ser confirmada atomicamente em todo o sistema. Para transações cross-shard, os shards relacionados precisam obter consistência. Normalmente, esse processo requer um tipo de transação de "retransmissão" para sincronizar entre os fragmentos relacionados;

5. **Epoch reconfiguration**: Para garantir a segurança dos shards, os shards precisam ser reconfigurados, a cada período de tempo chamado epoch, exigindo uma aleatoriedade. Essa aleatoriedade será usada na próxima epoch.

Com base nestes cinco componentes, iremos analisar diferentes propostas de *sharding*, que serão divididas em três categorias, partial sharding, complete sharding e outras soluções. No modelo de partição parcial, os grupos trabalham em conjunto para gerar uma cadeia única partilhada por todos. Enquanto no completo, cada grupo possui uma cadeia de blocos distinta. Por fim, na categoria outras soluções, iremos analisar propostas que destoam dos padrões vistos nas outras duas. A figura 2.7 ilustra a diferença entre partial sharding e complete sharding.

2.4. Partial Sharding

As primeiras propostas de *sharding* tentam resolver o problema da escalabilidade dividindo o poder computacional da rede, criando grupos capazes de minerar blocos de forma independente. Porém mantêm a estrutura base da *Blockchain* onde todos os nó da rede possuem uma cópia exata da *Blockchain*. Uma vez que cada nó recebe e armazena informações completas do sistema, não existem transações *cross-shard* no sistema, mas os nós ainda sofrem de armazenamento pesado e sobrecarga de largura de banda [Hong et al. 2021]. Uma vez que a fragmentação ocorra em apenas uma das três dimensões processamento,armazenamento e comunicação esse modelo é classificado com partial *sharding*.

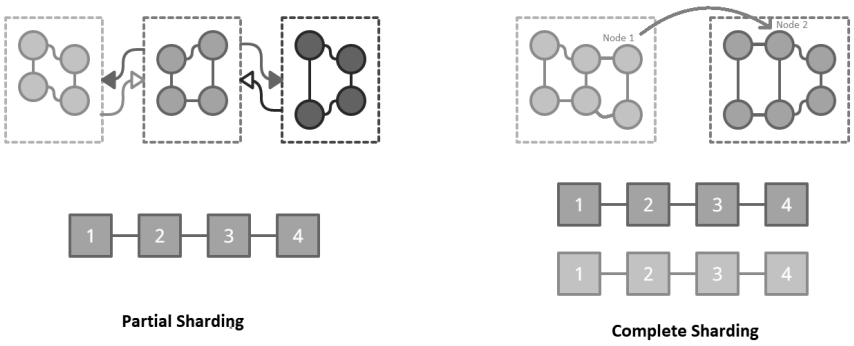

Figure 2.7. Estrutura dos modelos de *sharding*

2.4.1. Elastico

O ELASTICO [Luu et al. 2016] foi uma das primeiras *Blockchains* a fazer uso da técnica de *Sharding*. Em uma *Blockchain*, o sistema depende do poder majoritário para superar os invasores. No entanto, quando o poder de mineração é distribuído para zonas diferentes, um invasor pode reunir seus esforços em direção a uma única zona e pode facilmente exceder o limite de 51% dentro dessa zona[Wang and Wang 2019]. Para evitar que um comitê seja dominado, o ELASTICO utiliza epochs, períodos de tempo no qual os nós são redistribuídos em comitês de forma semi-aleatória de acordo com a identidade gerada, tal estratégia de redistribuição se torna essencial para preservar a segurança da blockchain quando fazemos uso de sharding, por conta disso, veremos variações desta mesma estratégia em todas as propostas.

2.4.1.1. Identity establishment and committee formation

No Elastico existem dois tipos de comitê, final e comum. O comitê final é único e responsável por unificar as soluções de cada comitê, checar as transações entre usuários de grupos diferentes e gerar as possíveis identidades para o próximo epoch. Os membros do comitê final são designados aleatoriamente junto dos demais.

Para estabelecer as identidades, cada nó gera uma **string** identidade usando um grupo de **strings** aleatórias válidas para o epoch. A geração dessas strings será discutida no etapa de epoch reconfiguration. Para gerar sua identidade, um nó realiza uma operação XOR bit a bit das **strings**. Uma vez que cada usuário tenha sua identidade definida, o comitê é gerado a partir dos últimos bits da identidade como mostrado na Figura 2.8. Normalmente o comitê final é composto pelos endereços com 0 nos bits da parte que determina o comitê.

Durante a etapa de formação dos comitês o autor busca boa aleatoriedade. Isso é:

1. Cada usuário tem uma string publicamente aleatória de r bits, gerada de forma verificável no epoch anterior;
2. Nenhum usuário tem acesso a string aleatória verificável mais do que δt antes do início do epoch;

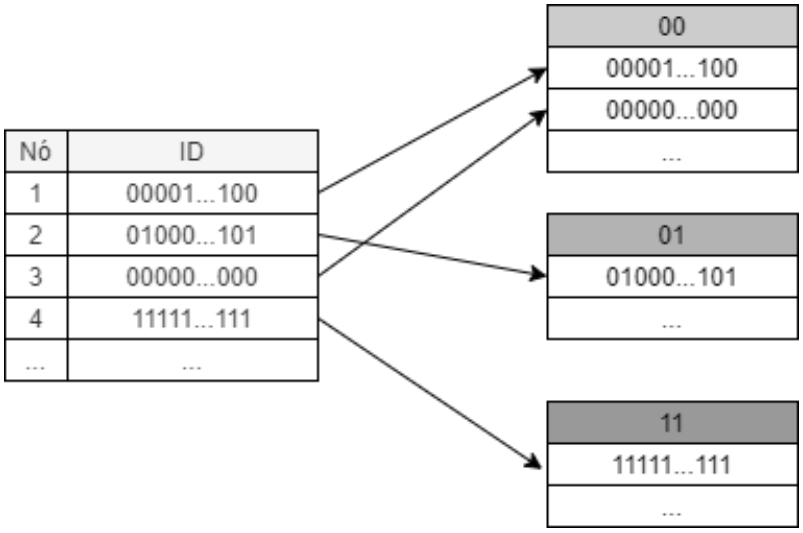

Figure 2.8. Formação de Comitê

3. Usuários mal-intencionados podem influenciar a aleatoriedade com probabilidade insignificante.

Para isso é necessário definir n_0 tal que das n' primeiras identidades criadas, no máximo $n'/3-1$ sejam maliciosas, para todo $n' \geq n_0$. Seja X_i uma variável que assume o valor 1 se a i-ésima identidade for gerada por um processador honesto e $X = \sum_{i=1}^{n'} X_i$. Então X segue a seguinte distribuição binomial:

$$Pr[X \geq 2n'/3] = \sum_{k=0}^{\lceil 2n'/3 \rceil} Pr[X = k] = \sum_{k=0}^{\lceil 2n'/3 \rceil} \binom{n'}{k} f^{n'-k}(1-f)^k \quad (1)$$

Desta forma a probabilidade diminui exponencialmente em n_0. Dado um parâmetro de segurança λ, podemos encontrar n_0 tal que $Pr[X \geq 2n'/3] \leq 2^{-\lambda}.\forall n' \geq n_0$

2.4.1.2. Overlay setup for committees

Uma vez definida a partição de cada usuário da rede, é necessário estabelecer a comunicação entre os membros de um mesmo grupo. É possível perguntar todos os nós da rede por aqueles que pertencem ao comitê, porém seria uma solução não escalável e mesmo que o ELASTICO não busque separar a rede para diminuir a quantidade de mensagens não é interessante aumentar o volume de comunicação da rede. A solução proposta foi que os primeiros nós formam um diretório de identidades, um novo membro envia sua identidade para o diretório e recebe um set de endereços do grupo. Desta forma reduz a complexidade de comunicação de $O(n^3)$ para $O(cn)$ onde c é a quantidade de nós no diretório.

Para diminuir a carga de mensagens na rede foi necessário limitar as visões dos nós da rede, mas isso pode introduzir falhas ao sistema. Essa propriedade busca garantir

que as inconsistências entres as visões sejam limitadas diminuindo o impacto gerado por elas, ou seja:

1. Cada membro tem sua própria opinião sobre quem faz parte do comitê. As visões de dois membros honestos diferem em no máximo 1/3 do tamanho do comitê;
2. Todos os membros honestos têm identidades de outros membros honestos em suas visões;
3. O número total de identidades únicas em todas as visualizações é no máximo $3c/2$ dos quais menos de $1/3$ são maliciosos.

Uma vez que os diretórios honestos aceitam todos os PoWs válidos na última rodada (antes que o comitê seja preenchido por pelo menos c membros), todos os membros honestos do comitê terão outras identidades honestas em sua visão, tornando comportamentos maliciosos a principal fonte de discrepância. Através da propriedade 1 esses comportamentos estão limitados a $1/3$ do tamanho do comitê.

2.4.1.3. Intra-committee consensus

Esta solução para a comunicação faz com que membros de um grupo tenham visões diferentes das identidades do comitê. Os nomes em comum entre as visões tornam o acordo possível. Cada membro tenta entrar em acordo com os membros de sua própria lista através de um algoritmo BFT tratando toda informação externa como falsa. O acordo se propaga entre as listas através dos nós comuns até que só exista uma solução e esta é divulgada para os membros do comitê final. De posse das soluções de cada comitê, o comitê final repete o processo, resolve o bloco e o adiciona na *Blockchain*.

Uma vez que o tamanho máximo das visões é 3c/2 contendo os membros honestos, qualquer protocolo consenso byzantino que aceite comportamentos maliciosos limitados a 1/3 consegue garantir o acordo. Assim, uma vez assinado por 2c+1 nós, ao menos 1 nó honesto executou o algoritmo de consenso e aprovou o valor escolhido.

2.4.1.4. Cross-shard transaction processing

O comitê final tem acesso aos dados de cada comitê. Logo este consegue resolver as transações cross-shard sem a necessidade de envolver os comitês. O comitê final resolve essas transações com o protocolo BFT escolhido.

2.4.1.5. Epoch reconfiguration

Cada membro do comitê final cria uma **string** de tamanho fixo e predeterminado. O comitê final discute e escolhe um set dessas **strings**. Todos os membros transmitem esse set e sua própria **string** junto com o bloco final. Cada nó da rede recebe parte dessa strings devido a latência ou manipulação dos invasores. Cada nó gera sua própria identidade através de um XOR de $2c+1$ dessas strings. O set divulgado e a quantidade de assinaturas

escolhidas garantem que existe pelo menos uma string gerada por um nó honesto entre as escolhas, preservando a aleatoriedade da identidade resultante.

De acordo com a distribuição feita na formação dos comitês, há no mínimo 2c/3 nós honestos no comitê final. Uma vez que cada nó recebe um conjunto de 2c/3 a 3c/2 strings. Como o número máximo de identidades geradas de forma maliciosa é c/2. Ao realizar o XOR em $2c + 1$ strings ao menos uma dessas strings foi gerada por um nó honesto. Como um usuário desonesto não conhece essa identidade antes do comitê final definir o set, a identidade gerada é perfeitamente randômica pois não é possível controlar o resultado para escolher o comitê do próximo epoch.

2.5. Complete Sharding

Sharding completo são aqueles que atingem a fragmentação nas bases computação, armazenamento e comunicação. A divisão do poder computacional permite, assim como no modelo parcial, aumentar o número de blocos minerados ao permitir que as partições minerem simultaneamente. Para o armazenamento o objetivo é diminuir o tamanho da cadeia armazenada em cada nó pois a mesma cresce indefinidamente. O consenso baseado em BFT é muito presente nos modelos que visam a escalabilidade pois as provas demandam tempo e esforço, impedindo que muitos blocos sejam minerados. Esses modelos de consenso requerem um grande volume de comunicação e começam a perder sua eficiência em redes maiores. A fragmentação da comunicação busca resolver esse problema.

Muitas pesquisas foram feitas com o objetivo de atingir o *sharding* completo de forma segura. O ChainSpace, o Omniledger e o RapidChain foram os modelos escolhidos para este estudo. ChainSpace é uma das primeiras pesquisas a atingir o *sharding* completo que usa um sistema a base de cliente que suporta *sharding* de contratos inteligentes genericos. Outro modelo a base de cliente é o Omniledger que usa os clientes para comitar transações entre partições. Já RapidChain propõe um mecanismo de transferência para o output de transações não gastas, baseados em UTXO, a estrutura de moeda que recebeu grande destaque com o crescimento do bitcoin.

2.5.1. Chainspace

Chainspace[Al-Bassam et al. 2017] é uma infraestrutura descentralizada, conhecida como razão distribuída, que suporta contratos inteligentes definidos pelo usuário e executa transações fornecidas pelo usuário em seus objetos. A execução correta de transações de contrato inteligente é verificável por todos.

Chainspace é seguro contra subconjuntos de nós que tentam comprometer sua integridade ou propriedades de disponibilidade através de *Byzantine Fault Tolerânce* (BFT) e técnicas de auditabilidade, e o não repúdio à *Blockchain*. Mesmo quando o BFT falha, mecanismos de auditoria estão disponíveis para rastrear participantes mal-intencionados.

2.5.1.1. Identity establishment and committee formation

A plataforma é agnóstica quanto à linguagem do contrato inteligente, ou infraestrutura de identidade. o ChainSpace suporta recursos de privacidade por meio de técnicas zero-

knowledge modernas como [Bootle et al. 2016] e [Danezis et al. 2014]. [Al-Bassam et al. 2017]

2.5.1.2. Overlay setup for committees

A Figura 2.9 ilustra o design do sistema do Chainspace que é composto por uma rede de nós de infraestrutura que gerenciam objetos válidos e garantem que, apenas as transações válidas nesses objetos sejam confirmadas.

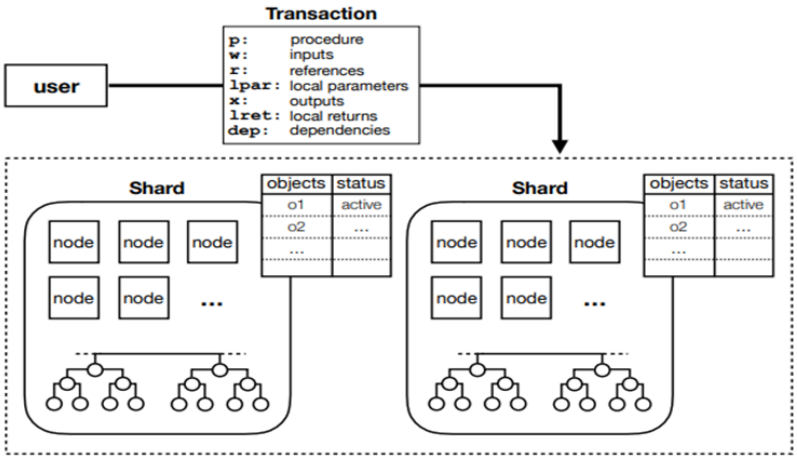

Figure 2.9. Design do Chainspace [Al-Bassam et al. 2017]

2.5.1.3. Intra-committee consensus

Um objeto representa uma unidade de dados no sistema Chainspace, como uma conta bancária, e está em um dos seguintes três estados:

- **ativo:** pode ser usado por uma transação;
- **bloqueado:** está sendo processado por uma transação existente;
- **inativo:** foi usado por uma transação anterior.

Objetos também têm um tipo que determina o identificador exclusivo do contrato inteligente que os define. Os procedimentos de contratos inteligentes podem operar apenas em objetos ativos, enquanto os objetos inativos são retidos apenas para fins de auditoria.

O Chainspace permite a composição de contratos inteligentes de diferentes autores para fornecer recursos do ecossistema. Cada contrato inteligente é associado a um verificador para permitir o processamento privado de transações em nós de infraestrutura, pois os verificadores não usam nenhum parâmetro local secreto. Os verificadores são funções puras, determinísticas e sem efeitos colaterais, que retornam um valor booleano.

Uma transação válida aceita objetos de entradas ativas junto com outras informações auxiliares e gera objetos de saída. Para obter alto rendimento de transação e baixa latência, o Chainspace organiza os nós em fragmentos que gerenciam o estado dos objetos, controlam sua validade e registram as transações canceladas ou confirmadas. Isso foi Implementado pelos autores usando o *Sharded Byzantine Atomic Commit* (S-BAC) - um protocolo que compõe o acordo BFT existente e primitivas de commit atômico de uma maneira nova.

Sharded Byzantine Atomic Commit

O protocolo tem sua origem combinando propriedades de outros dois protocolos primitivos: Byzantine agreement que permite que os nós entrem em consenso se a partição tiver até 1/3 nós maliciosos e Atomic commit uma partição apenas confirma sua transação após esta ser aceita pelas demais partições interessadas.

A Figura 2.10 mostra os passos para aceitar uma transação, uma vez que a partição que propôs uma transação T ele a transmite na rede AllPrepared(commit,T). A transação se torna ativa até que o primeira partição interessada a aceite através do LocalPrepared(commit,T) que trava a transação reservando os recursos. Nesse estado, se alguma partição negar com LocalPrepared(abort,T) os recursos são liberados e a transação retorna ao estado ativo. Uma vez que todos as partições confirmem a transação. O AllPrepared(commit,T) é confirmado e a mesma é registrada no bloco.

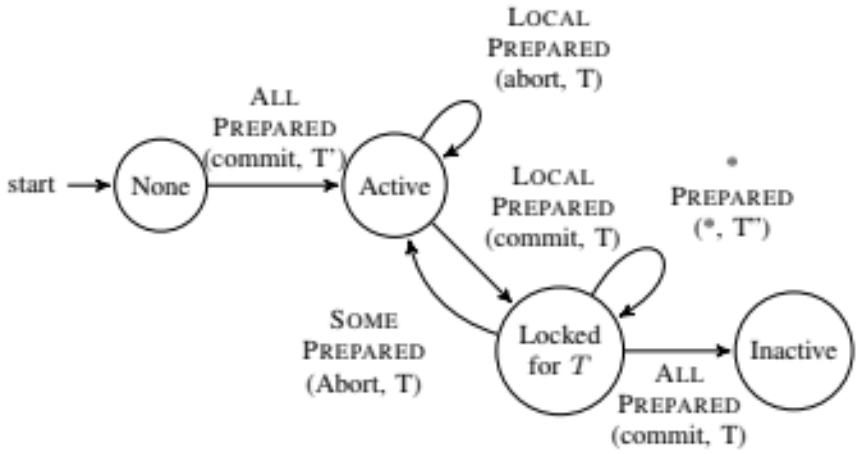

Figure 2.10. Estados de uma transação [Al-Bassam et al. 2017]

O protocolo S-BAC garante três propriedades-chave, que refletem a segurança do Chainspace: vivacidade, consistência e validade.

1. **Vivacidade:** Uma transação T que é proposta a pelo menos um nó honesto, eventualmente resultará em uma decisão única, ou seja, todos os nós decidindo entre accept(commit, T) ou accept(abort, T).

Baseado na propriedade de vivacidade do acordo byzantino. Assumindo que "prepare(T)" foi dado a um nó honesto, será enviado "prepared(commit, T)" ou "prepared(abort, T)" dos nós da partição BFT para os os nós da partição interessada. Ao receber a mensagem, a nova partição agendará um "prepare(T)" para seus nós eventualmente gerando a decisão adequada.

2. **Consistência:** Não serão confirmadas duas transações conflitantes, ou seja, transações que compartilham a mesma entrada. Além disso, existe uma execução sequencial para todas as transações.

   Uma transação só é confirmada se alguns nós concluírem "ALLPREPARED(commit, T)", e para isso esses nós devem receber "LOCALPREPARED(commit, T)" dos nós envolvidos. Duas transações concorrentes possuem um subgrupo de nós envolvidos em comum e uma vez que uma delas receba o accept local a outra será bloqueada, recebendo local abort, até que a primeira seja confirmada ou rejeitada.

3. **Validade:** Uma transação só pode ser confirmada se for válida de acordo com os verificadores de contrato inteligentes combinando as características dos procedimentos que executa.

   Uma transação só é confirmada se todos os nós interessados garantiram a regularidade com seu "local accept" assim uma vez que a transação receber "AllPrepared(commit,T)" ela é válida.

2.5.1.4. Cross-shard transaction processing

Os nós comunicam aos shards envolvidos para decidirem se aceitam ou rejeitam uma transação via consenso cross-shard. Em vez de uma abordagem orientada ao cliente, o ChainSpace executa o protocolo S-BAC colaborativamente entre todos os comitês envolvidos. Isso é alcançado ao fazer com que todos os comitês atuem como um gerente de recursos para as transações que eles gerenciam. [Wang et al. 2019]

2.5.1.5. Epoch reconfiguration

Embora a reconfiguração não seja definida uma vez que a formação do shard fica em aberto, o Chainspace oferece uma ferramenta de auditabilidade para identificar nós maliciosos nesta etapa.

Uma partição maliciosa (com mais de f nós defeituosos) que tentou adicionar uma transação ou objeto inválido no estado de outra partição, pode ser detectado por um auditor realizando uma auditoria completa do sistema Chainspace. Um par hash-chains de partições distintas são válidos se:

- A reexecução das transações levam ao mesmo estado.
- todas as mensagens prepared(Transaction,*) são compatíveis.

Se duas hash-chains se provarem incompatíveis é possível determinar qual é a desonesta isolando as assinaturas.

2.5.2. Ominiledger

O OmniLedger[Kokoris-Kogias et al. 2018] parte de um modelo mais simples de *sharding*, denominado SimpleLedger, e estuda os problemas para encontrar a solução ótima. O SimpleLedger evolui em epochs, e utiliza um coordenador para gerar um valor aleatório que cada nó usa para determinar sua partição. As melhorias necessárias para chegar ao OmniLedger desse modelo são:

- Performance:
  1. ByzCoinX: consenso Byzantino robusto;
  2. Podar a *Blockchain* das partições;
  3. Validação Trust-but-Verify;
- Segurança
  1. *Sharding* através de aleatoriedade distribuida;
  2. Transição de epoch suave;
  3. Atomix: transação entre partições atômica;

2.5.2.1. Identity establishment and committee formation

Para participar de um epoch *e*, o nó deve registrar-se em uma blockchain identidade no epoch *e - 1*, para geração da identidade pode-se utilizar qualquer mecanismo resistente a um ataque sybil, como por exemplo, Proof of Work, Proof of Stake, entre outros.

Um passo importante para a descentralização é definir como associar os validators(nós) às partições. Para isso é necessário um protocolo de geração de distribuição aleatória que seja Imparcial, Imprevisível, Verificável por terceiros e Escalável. O protocolo escolhido foi o RandHound [Syta et al. 2016].

RandHound assume um líder honesto que é responsável por coordenar a execução do protocolo e para fazer a aleatoriedade produzida disponível para os outros. No entanto, nós nem sempre podemos garantir que um líder honesto será selecionado. Cada vez que um líder controlado pelo adversário é eleito e executa RandHound, o adversário pode escolher aceitar a saída aleatória e a atribuição de partição produzida por ela, ou desistir dessa e tentar novamente na esperança de uma mais favorável porém ainda aleatória. Ao atingir o número máximo de tentativas o líder falha, e por padrão este líder não poderá participar da próxima eleição.

No início de cada epoch, cada validator gera um ticket usando uma função aleatória e verificável, e compartilha com os outros nós, o ticket válido com menor valor é escolhido como novo líder.

2.5.2.2. Overlay setup for committees

Como veremos mais adiante, o OmniLedger faz uma reconfiguração gradual dos shards, ou seja, os nós vão entrando aos poucos nos shards para os quais foram designados. Uma

vez que um nó esteja pronto, ele envia uma mensagem ao líder do shard informando-o, o líder então permite a entrada do nó no shard.

2.5.2.3. Intra-committee consensus

Para o consenso intra-shard utiliza-se o ByzCoinX, uma variante do ByzCoin, que utiliza padrões de comunicação mais robustos para processar transações de forma eficiente dentro dos shards, mesmo se alguns dos validadores falharem, e que resolve dependências no nível da transação para obter uma melhor paralelização de blocos.

O OmniLedger pode, opcionalmente, utilizar uma arquitetura diferente nos shards para permitir o processamento em tempo real de transações. Ela consiste no uso de validadores otimistas, que formam um grupo menor e portanto mais rápido para chegar a um consenso. Os blocos otimistas produzidos por tais validadores, são posteriormente verificados pelo núcleo de validadores, que é um grupo muito maior de validadores, este grupo portanto é mais lento para processar as transações, porém mais seguro. Esta abordagem foi chamada de Trust-But-Verify, seu esquemático pode ser visto na figura 2.11.

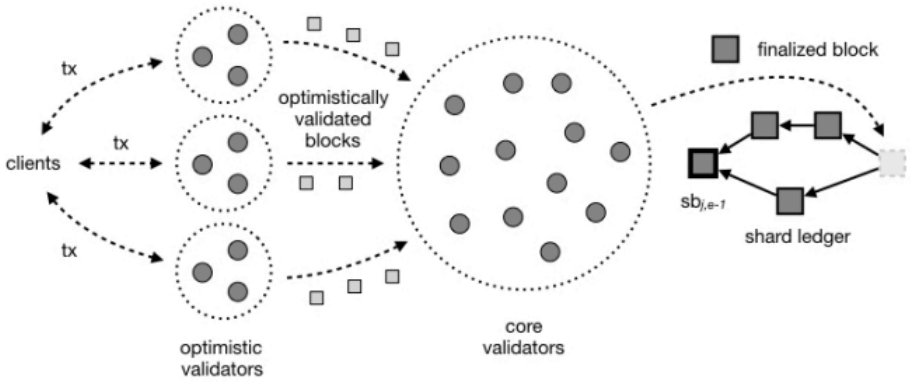

Figure 2.11. Trust But Verify [Kokoris-Kogias et al. 2018]

2.5.2.4. Cross-shard transaction processing

Para processar transações cross-shard, garantindo sua atomicidade, foi desenvolvido o Atomix, que usa o modelo de estado UTXO, onde o balanço de um usuário é armazenado na forma de tokens de valor não gasto (disponível). O protocolo tem três etapas:

1. **Initialização.** Um cliente cria uma transação entre partições que gasta o UTXO de algumas partições de entrada (IS) e cria novos em algumas partições de saída (OS);

2. **Lock.** Cada IS verifica se o UTXO pode ser gasto. Se possível o recurso é travado e a partição divulga proof-of-acceptance, caso contrário a partição divulga proof-of-rejection;

3. **Unlock.** Há duas opções nesta etapa, Unlock to Commit que consome os recursos travados para criar novos no destino da transação e Unlock to Abort, se um IS rejeitou a transação o cliente comunica aos demais para liberar os recursos.

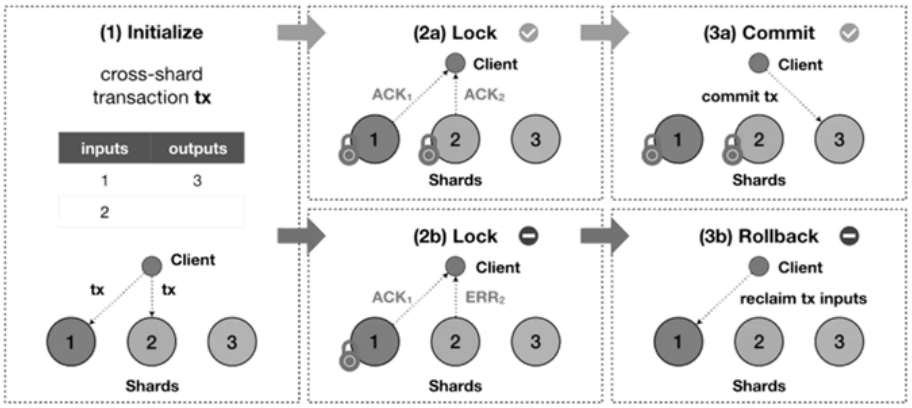

Figure 2.12. Protocolo Atomix [Kokoris-Kogias et al. 2018]

2.5.2.5. Epoch reconfiguration

Para manter a operabilidade durante as fases de transição, o OmniLedger troca os validadores de cada shard gradualmente por epoch. Isso permite que os operadores restantes continuem disponíveis (no cenário honesto) aos clientes enquanto os validadores recém-associados estão inicializando. A fim de alcançar esta operação contínua, podemos trocar no máximo $1/3$ do tamanho da partição, porém quanto maior for o lote, maior é o risco de que o número de validadores honestos remanescente seja insuficiente para chegar a um consenso e mais estresse é gerado pela inicialização de novos validadores a rede.

2.5.3. RapidChain

O RapidChain[Zamani et al. 2018] consiste em três componentes principais: Bootstrap, Consenso e Reconfiguração. Assim como muitos métodos de *sharding*, o RapidChain é executado em períodos de tempo fixos chamados epochs. No primeiro epoch, um protocolo de bootstrapping único é executado, permitindo que os participantes concordem em um comitê. Esse comitê, que chamamos de comitê de referência, é responsável por conduzir eventos de reconfiguração periódicos entre os epochs. Cada epoch consiste em várias iterações de consenso seguidas por uma fase de reconfiguração.

2.5.3.1. Identity establishment and committee formation

As identidades são geradas usando um mecanismo gerador de identidades Sybil-resistant como [Andrychowicz and Dziembowski 2015]. Este mecanismo requer que cada nó resolva um puzzle computacionalmente difícil com as identidades geradas localmente.

O conjunto inicial de participantes inicia o RapidChain executando uma eleição de comitê protocolo, onde todos os nós concordam com um grupo de $O(\sqrt{n})$ nós, aos

quais nos referimos como grupo raiz. O grupo é responsável por gerar e distribuir uma sequência de bits aleatórios que são usados para estabelecer um comitê de referência de tamanho O (log n). Em seguida, o comitê de referência cria k comitês C1, ..., Ck cada um de tamanho O (log n). A fase de bootstrap é executada apenas uma vez no início do RapidChain.

Estabilidade do Bootstrapping

Suponha que haja n nós e uma fração constante, 1/3 desses nós estão corrompidos. No fim do protocolo de bootstrapping do RapidChain, quase todos (percentil 99) os nós não corrompidos concordam com a string aleatória r_0 e, consequentemente, com todos os comitês do *sharding* com probabilidade constante maior que 0.

2.5.3.2. Overlay setup for committees

Na etapa de bootstrapping, os nós do mesmo comitê descobrem um ao outro através de algoritmo peer-discovery. Os nós que são realocados pela regra de cuckoo ou que entram na rede tardiamente baixam fazem download dos dados do seu novo comitê.

Cada transação tx é submetida por um usuário do sistema a um grupo arbitrário de nós do RapidChain. Esses nós roteiam tx, por meio de um protocolo de roteamento entre comitês, a um comitê responsável pelo armazenamento de tx. O protocolo de roteamento inspirado no Kademlia [Maymounkov and Mazieres 2002]. O protocolo Kademlia foi aplicado no RapidChain a nível de comitê. Especificamente cada comitê mantém endereço dos *logn* comitês mais próximos. Assim, o responsável por identificar o comitê que deve armazenar tx, só comunica com um número logaritmo de comitês para encontrá-lo.

2.5.3.3. Intra-committee consensus

Assim que os membros de cada comitê concluem a reconfiguração da epoch atual, eles aguardam para que usuários externos enviem suas transações. Cada usuário envia suas transações para um subconjunto de nós (encontrado através de um protocolo de descoberta P2P) que agrupa e encaminha as transações à comissão responsável pelo seu tratamento. O comitê executa um protocolo de consenso Bizantino dentro do comitê para aprovar a transação e adicioná-la à sua razão.

Estabilidade do consenso intra-comitê

- **Safety:** Com fração de nós corrompidos $f = 1/2$. Provamos safety para um cabeçalho de bloco específico proposto pelo líder na iteração i. Suponha que o nó P é o primeiro nó honesto a aceitar um cabeçalho Hi para i. Em todas as rodadas da iteração i ou todas as iterações após i, nenhum líder pode construir uma proposta segura para um cabeçalho diferente de Hi uma vez que ele não pode obter votos suficientes de nós honestos em um valor que não é seguro. É impossível finalizar uma raiz Merkle de um bloco que está associada a dois blocos diferentes.

- **Liveness:** Em primeiro lugar, observe que todas as mensagens no sistema são mantidas por meio de assinaturas digitais. Todos os nós honestos aceitarão todos os blocos com o valor seguro pendentes, ou eles já os aceitaram, assim que o líder para a altura do bloco atual for honesto. Como líderes são escolhidos aleatoriamente e a aleatoriedade é imparcial, cada comitê terá um líder honesto a cada duas rodadas na expectativa. O líder honesto enviará propostas válidas para todos os blocos pendentes para todos os nós que é seguro propor e tem uma prova válida. Assim, todas as réplicas honestas já aceitaram o esse valor ou irão aceitá-lo uma vez que é seguro.

2.5.3.4. Cross-shard transaction processing

A transação cross-shard no RapidChain baseia-se amplamente no esquema de roteamento inter-comitê. O esquema permite que os usuários e líderes de comitês identifiquem rapidamente a quais comitês eles devem enviar sua transação. Em transações cross-shard em RapidChain, cada transação cria três transações diferentes para criar o mesmo resultado. Todas as três sub-transações são single-shard. Em caso de falha em algumas das sub-transações o RapidChain usa um mecanismo de rollback atômico.

2.5.3.5. Epoch reconfiguration

A reconfiguração permite que novos nós estabeleçam identidades e se juntem aos comitês existentes, e garante que todos os comitês mantenham sua resiliência de 1/2. A reconfiguração do RapidChain se baseia na regra Cuckoo, quando um novo nó se junta a rede ele é designado a um comitê aleatório e alguns nós desse comitê são realocados para os demais. Além disso, o RapidChan separa seus comitês em dois grupos pelo tamanho, sempre que um novo nó se junta, ele é alocado a um comitê do grupo dos maiores e a realocação é feita para o grupo dos comitês menores mantendo o tamanho dos comitês mais estáveis.

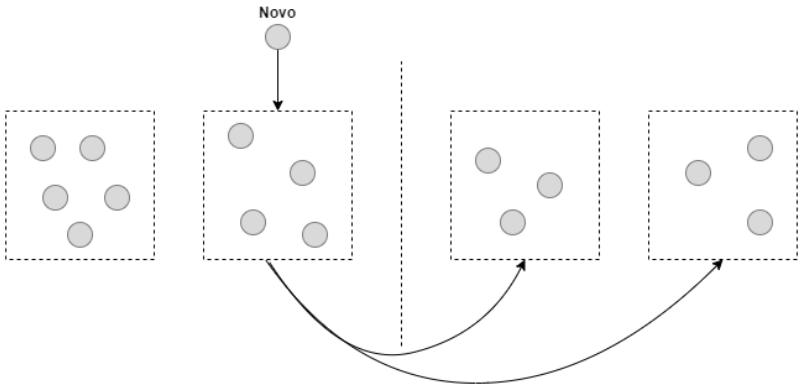

Figure 2.13. Regra de Cuckoo

Estabilidade do Epoch

A função que permite calcular a probabilidade de falha (ter mais nós corrompidos que um dado limite) de um epoch:

$$Pr[X \geq [m/2]] < \sum_{x=[m/2]}^{m} \frac{txn - tm - x}{nm} \tag{2}$$

Onde m é o tamanho do comitê, n é o número de nós da rede e t é o limite superior de nós corrompidos na rede.

Ao contrário da distribuição binomial, a distribuição hipergeométrica depende diretamente do total tamanho da população. Uma vez que n pode mudar ao longo do tempo em uma rede de adesão aberta, a probabilidade de falha pode ser afetada. Para manter a probabilidade de falha desejada, cada comitê em RapidChain executa um consenso em intervalos pré-determinados para concordar com um novo tamanho do comitê.

Para limitar a probabilidade de falha de cada epoch, calculamos o total de comites k = n/m, onde cada um pode falhar com a probabilidade calculada anteriormente. Portanto, a probabilidade de fracasso de cada epoch é:

$$Pepoch < Pbootstrap + k \cdot Pcommittee \tag{3}$$

Estabilidade do Reconfiguration

Assumimos que a qualquer momento durante o protocolo, a fração de nós corrompidos para nós honestos é ε. Nós também assumimos que o protocolo começa a partir de um estado estável com n nós particionados em m comitês que satisfaz as condições de equilíbrio e honestidade. Definimos o conjunto de comitês ativos como os comitês de m/2 com maior número de nós.

Para provar o teorema, é suficiente provar as propriedades de balancing e honesty para qualquer comitê.

- **Balanceamento:** O número máximo de nós em cada comitê é $c\log n + c/2(1+\delta)(3-\frac{t}{n}+\frac{t}{n-t}k)\log n$ e o mínimo é $c/2(1-\delta)\log n$;
- **Honestidade:** Escolhendo k tal que $\frac{t}{n-t} < 1-1/k$ qualquer comitê tem $(1-t/n)(1-\delta)c\log nk/2$ nós honesto e $\frac{t}{n-t}(1+\delta)c\log nk/2$ nós corrompidos com alta probabilidade. Observe que esses valores são calculados para o pior cenário, quando o adversário mirou no comitê de tamanho $(c\log n)k/n$.

2.5.4. Cycledger

O CycLedger [Zhang et al. 2020] é uma proposta mais recente de Sharding, por conta disso, ele pôde fazer uma análise crítica sobre as propostas anteriores. Nesta análise destacou-se três pontos que são considerados deficiências destas propostas, são eles: todos os nós sem falhas devem se conectar bem um com o outro; há uma grande perda de eficiência quando a honestidade dos líderes dos shards é questionável; e não há um incentivo explícito para os nós participarem ativamente do protocolo.

Procurando contornar tais deficiências, o ClycLedger se apresenta como uma solução que: é escalável, pois faz sharding completo; se mantêm robusta mesmo quando os líderes dos shards são desonestos, dado que, cada shard possui um conjunto de nós que supervisionam o líder do shard e podem atuar como suplentes se necessário; e que pretende fornecer incentivos o suficiente para que os nós participem de forma honesta do protocolo, fazendo uso de um sistema de reputação que associa a reputação ao poder computacional do nó, fazendo com que os nós que contribuem mais para a rede sejam devidamente recompensados para se manterem honestos. A proposta assume que não há mais que 1/3 de nós maliciosos, e que mais de 1/2 dos nós de um shard são honestos.

O CycLedger funciona em rounds. Sabendo que cada nó possui um par (chave pública, chave privada) e uma reputação w_r. A cada novo round r é escolhido em $r-1$ um comitê árbitro, que irá gerenciar a identidade dos nós, produzir a próxima aleatoriedade e propor o bloco do round r. Ao mesmo tempo, os outros nós são designados para os m shards, idealmente formando shards de tamanho $c = O(log^2 n)$, dessa forma temos que $n = m * c$. Cada shard possui um líder, λ líderes em potencial(conjunto de sub líderes) e $c - \lambda - 1$ membros. A figura 2.14 demonstra a estrutura discutida. É importante ressaltar que o bloco gerado no round $r-1$ contém também os endereços dos membros do comitê árbitro, dos líderes e sub líderes de cada shard para o round r.

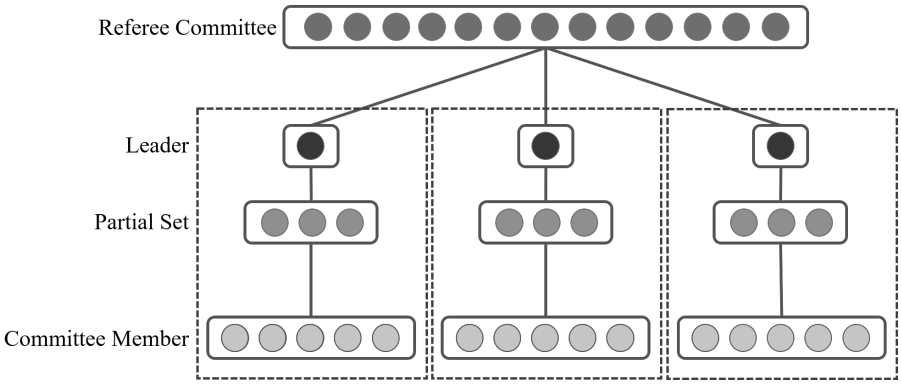

Figure 2.14. Comitês existentes no CycLedger [Zhang et al. 2020]

2.5.4.1. Identity establishment and committee formation

A identidade do nó é estabelecida utilizando um PKI(Public Key Infrastructure) para dar a cada nó um par (chave pública, chave privada). Para que um nó saiba para qual shard será designado é utilizado uma VRF(Verifiable Random Function), uma função que retorna um hash e uma prova, que junto com a chave pública do nó comprova que ele gerou aquela hash.

Papel dos Membros dos Shards

- Determina o id do seu shard usando a VRF;
- Envia sua chave pública, endereço, a hash e a prova pi gerada pela VRF, para o líder e sub líderes do shard;

- Quando o nó i recebe a lista de membros do shard, do líder ou de um sub líder, ele envia os mesmos itens enviados no passo anterior para todos os nós da lista;
- Quando i recebe a chave pública j de um nó j, ele verifica se j está no mesmo shard que ele utilizando a chave e a prova pi.

Papel dos Líderes e Sub Líderes dos Shards

- Mantém uma lista <Chave Pública, Endereço>. Inicialmente ela contém apenas o líder e os sub líderes;
- Quando recebe a chave pública do nó j, verifica se j pertence ao shard utilizando a chave e a prova pi, se confirmado, j é adicionado à lista de nós daquele shard.

2.5.4.2. Overlay setup for committees

No caso do CycLedger, esta etapa é feita em paralelo com a formação dos shards, pois, como vimos, os líderes e sub líderes de cada shard mantém uma lista de membros do shard, que é atualizada conforme novos membros se identificam para os líderes do shard. A partir desta lista os membros estabelecem a comunicação entre si.

Assume-se que há uma boa conexão enquanto os líderes e seus conjuntos são linkados. Supõe-se também que cada líder ou membro do conjunto de sub líderes está conectado com todo o comitê árbitro daquele round, enquanto as soluções anteriores supunham uma boa conexão entre todos os nós honestos. Por último supõe-se que a comunicação dentro do shard é síncrona considerando um delay delta, o que é plausível, dado que os shards possuem apenas centenas de nós. Os líderes e sub líderes de todos os shards também possuem uma comunicação síncrona entre si, porém considera-se um delay maior.

2.5.4.3. Intra-committee consensus

Para realizar o consenso, o CycLedger faz uso de algoritmo que ele chama de semi-commitment exchange, que funciona da seguinte forma: Dada uma transação, primeiramente o líder do shard faz o multicast de <r, M, H(M), sn> com a tag PROPOSE, onde *r* é o número do round, *M* é a mensagem original, *H(M)* é o resumo da mensagem e *sn* é o número de sequência da mensagem, que é único. Quando um nó *i* recebe *M* e *H(M)*, ele verifica a corretude do resumo, o número do round e verifica também se sn é único e então faz o broadcast <r, sn, H(M), i> com a tag ECHO e retransmite o PROPOSE para os outros membros. Caso o nó i recebe de mais da metade dos membros , um ECHO e um PROPOSE idênticos aos que ele produziu e recebeu respectivamente, ele irá enviar um <r, sn, H(M), i> com a tag CONFIRM para o líder do shard. Caso algum nó honesto perceba que o líder enviou mensagens diferentes, então ele informa aos outros membros do shard e o consenso é interrompido, o líder é então expulso e um sub-líder tomará o seu lugar.

O consenso intra-shard é dividido em 5 passos:

1. Após receber transações de usuários externos, cada líder cria uma lista de transações internas;
2. O líder faz o broadcast da lista para todos os membros do shard;
3. Após receber a lista de transações, para cada transação o nó vota, Yes, No ou Unknow, onde Unknow é quando o nó não consegue avaliar a transação. Feito isso ele envia a lista de votos para o líder;
4. O líder então pega as transações que obtiveram maioria de votos Yes, e executa o algoritmo descrito na subseção anterior;
5. Por último, com aprovação de ao menos metade dos membros do shard, as transações são aceitas.

2.5.4.4. Cross-shard transaction processing

O consenso Inter-Shard funciona de maneira análoga ao intra-shard, porém, a lista de transações criada pelo líder possui apenas transações cross-shard. Chamemos então esta lista de $TXList_{i,j}$, nesta lista temos transações entre os shards i e j. Após o shard i chegar a um consenso a respeito das transações em $TXList_{i,j}$, o líder do shard i envia o consenso e a lista de membros do shard i para o líder e sub líderes do shard j. O líder de j então chegará a um consenso sobre a lista recebida e enviará o consenso e sua lista de membros ao líder e sub líderes de i.

Após a conclusão do consenso, intra-shard ou cross-shard, o líder garante a cada membro votante uma pontuação de acordo com a proximidade dos votos daquele membro com o resultado do consenso. Dessa forma cria-se uma relação entre a reputação e o poder computacional. O líder então organiza essa pontuação em uma lista e faz o broadcast dela para os membros do shard, para que a reputação dos membros seja atualizada.

2.5.4.5. Epoch reconfiguration

Na proposta em questão, a Epoch reconfiguration é dividida em duas etapas: A seleção do comitê árbitro, do líder e sub líderes do shard; e da geração e propagação do bloco. Isto acontece pois, o endereço dos nós escolhidos para atuar como parte do comitê árbitro ou como líder/sub líder de um shard no próximo round, deve constar no bloco gerado no round atual, permitindo que esta informação esteja disponível durante a formação dos shards no próximo round.

Seleção do Comitê Árbitro, Líderes e Sub Líderes

Usando uma geração de aleatoriedade distribuída, os membros do comitê árbitro do round r, chamaremos este comitê de C_r, eles irão gerar a seed que será usada no round r + 1. Os

nós que desejam participar do próximo round devem resolver um puzzle PoW, ao resolver, eles devem enviar a sua solução para o C_r, o comitê então irá registrar a identidade do nó. Desta forma, C_r está ciente de todos os participantes do próximo round, ele então escolhe como líderes dos shards os m nós de maior reputação. Fazendo uso da aleatoriedade do round atual, C_r irá definir o comitê árbitro e os sub líderes, de cada shard, do próximo round.

Geração e Propagação do Bloco

O comitê árbitro após receber todas as transações, verifica e encapsula todas as legítimas junto com a aleatoriedade gerada para o próximo round, os participantes do próximo round, a reputação atualizada dos nós, o comitê árbitro escolhido para o próximo round, bem como os líderes e sub líderes. Uma vez que este bloco foi gerado ele é liberado para consulta para todos os nós. Cada nó então chega a um consenso sobre as transações das quais participa após ver o bloco gerado, e o líder envia estas transações para o comitê árbitro do próximo round.

2.5.5. Sharper

SharPer é um sistema de blockchain permissionada. Esse sistema agrupa os nós em clusters e fragmenta os dados e a razão da blockchain. Com isso permite o processamento paralelo de transações. Cada bloco contém uma única transação e é armazenado em cada nó dos clusters envolvidos. Com as transações entre shards a razão é formada como grafo acíclico dirigido. A figura 2.15 mostra um exemplo do grafo formado com as razões do SharPer a partir de um bloco genesis λ

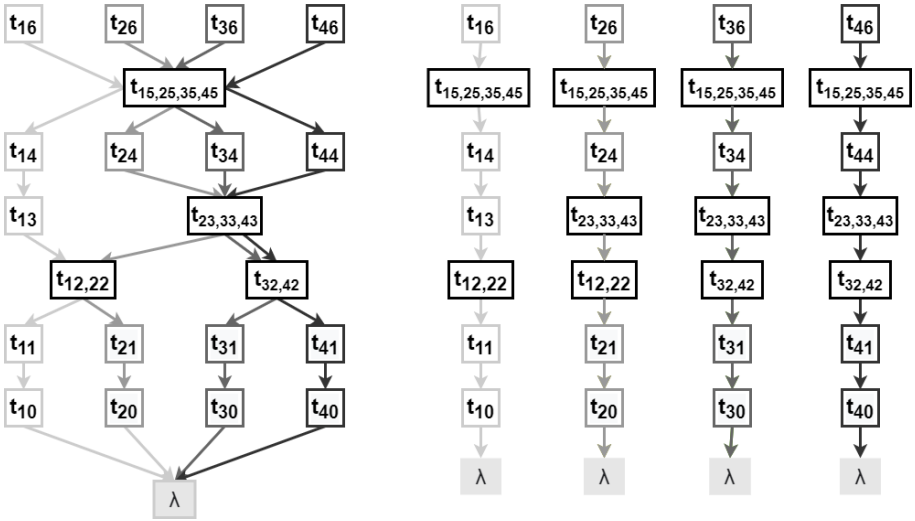

Figure 2.15. Blockchain do SharPer com quatro clusters [Amiri et al. 2019]

2.5.5.1. Identity establishment and committee formation

Em Blockchains não permissionadas, cada cluster consiste em centenas de nós. Este tamanho é necessário para atingir, com boa probabilidade, o valor de 1/3 de nós com

falha. Em outros modelos permissionados o tamanho necessário do cluster é bastante reduzido. O AHL por exemplo garante a segurança com 80 nós. O sharPer fornece um modelo determinístico que fornece segurança com valores próximos ao mínimo para superar a falha. Em um sistema de 16 nós e com falhas bizantinas onde f = 1 e , o tamanho do cluster é 16/4 (3f+1). O último cluster será um pouco maior se a divisão não for exata.

Os protocolos de consenso tolerante a falhas demandam um grande volume de mensagens. Com isso o tamanho reduzido do cluster diminui a latência no sistema. Para diminuir ainda mais a latência, os clusters são distribuídos de acordo com a sua localização geográfica. Nós próximos são agrupados no mesmo cluster.

2.5.5.2. Overlay setup for committees

O SharPer usa comunicação ponta a ponta bidirecional e autenticada. Com isso, um nó malicioso não pode forjar uma mensagem de um nó correto. As mensagens ainda podem conter assinaturas de chave pública e digests para reforçar a integridade. Os nós no SharPer seguem o modelo de falha de parada ou falha bizantina. No modelo de falha de parada, os nós não podem conspirar, mentir ou tentar subverter o protocolo. Com apenas falhas por parada são necessários 2f+1 nós para garantir a segurança com falha de f nós. Enquanto que o modelo resistente a falha bizantina, requer 3f+1 para superar o mesmo número de nós defeituosos.

2.5.5.3. Intra-committee consensus

Crash-Only Nodes

O SharPer usa uma variação do algoritmo Paxos [Lamport et al. 2001] denominada de multi-Paxos. Neste algoritmo um líder estável é pré-escolhido para coordenar o protocolo. Os clientes enviam suas requisições assinadas para o líder. O líder atribui o número de sequência da transação e propaga uma mensagem de propostas para os nós do cluster. O líder aguarda por f nós aceitarem a proposta. Com o líder, f+1 nós aceitaram a proposta garantindo que ao menos 1 nó sem falha aceitou a proposta.

Byzantine Nodes

Para o modelo resistente à falha byzantina, o SharPer usa PBFT [Castro et al. 1999] para garantir a segurança na presença de até f nós maliciosos. Para isso, cada participante do protocolo se comporta como um líder e tenta aprovar a proposta. O protocolo inicia com um cliente solicitando uma transação a um líder que repassa para todos os nós do cluster (pre-prepare). De posse de um proposta válida, cada nó envia uma mensagem de accept aos demais. Os nós do cluster esperam por 2f dessas mensagens, 2f+1 com o próprio, para propagar sua mensagem de commit aos demais. Mais uma vez, se nó receber 2f propostas de commit, ele responde ao cliente. A proposta é aprovada com f+1 respostas ao cliente.

Se um tempo após enviar a proposta ao primeiro líder, o cliente não receber nenhuma resposta ele envia a propostas a todos os nós do protocolo. Neste cenário é provável que o primeiro líder tenha falhado.

2.5.5.4. Cross-shard transaction processing

Crash-Only Nodes

As transações cross shards são processadas de forma similar. O líder envia a proposta para todos os nós dos clusters envolvidos e a aguarda por f+1 nós de cada cluster aceitarem a proposta. Devido a latência na rede, as mensagens para os demais clusters podem ser conflitantes. Um nó pode não ter recebido a última mensagem de seu líder ou mais de um líder divulgar uma proposta no mesmo instante. Para lidar com esse problema o líder que propõe a transação cross shard pode se comunicar com o líder de cada cluster. Então cada cluster processa a mensagem como um transação interna, resolvendo o conflito.

Byzantine Nodes

O protocolo tolerante a falhas bizantinas é semelhante ao caso com apenas falha por parada. acordo de todos os envolvidos. Ainda é necessário que todos os clusters concordem com a proposta, embora o tamanho do quorum passa a ser 2f + 1. Além disso, devido ao possível comportamento malicioso do nó primário, todos os nós, de cada cluster envolvido, enviam as mensagens de accept e commit para cada outro nó.

2.5.5.5. Epoch reconfiguration

O Sharper usa uma distribuição determinística baseada na localização geográfica. Além disso, tem como base uma Blockchain permissionada, onde cada nó possui sua identidade única. Desta forma o modelo descarta a necessidade da reconfiguração e da delimitação de tempo, epochs.

2.6. Outras Soluções

O Pyramid e LightChain buscam atingir objetivos específicos ao modificar a estrutura do *sharding*. O pyramid sobrepõe as partições para otimizar o processamento de transações entre partições. Uma vez que um nó participe de 2 partições ele é capaz de conferir transações entre nós de suas partições. Mesmo que eles estejam, cada um em uma partição.

O LightChain tem por objetivo reduzir a carga de memória gerada pelo crescimento constante da *Blockchain*. A ideia é dividir a cadeia entre as instâncias de execução com auxílio de uma tabela hash distribuída. Esse modelo acaba por fragmentar a comunicação pois as instâncias só enxergam os nós em sua tabela.

2.6.1. Pyramid

O Pyramid [Hong et al. 2021] propõe um novo modelo de *sharding*, o *Layerd sharding*, no qual as partições passam as se sobrepor ao invés de serem completamente isoladas. Desta forma, alguns nós participaram de mais de uma partição. Os nós que fazem parte da sobreposição podem verificar e processar as transações entre partições, envolvendo as

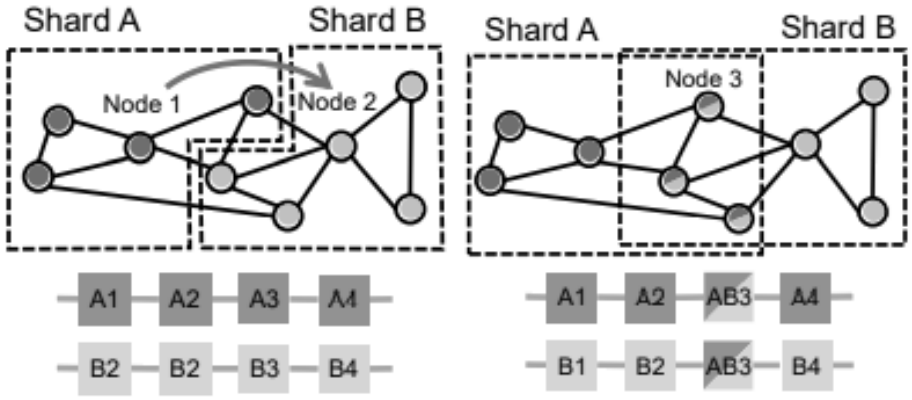

Figure 2.16. Modelo Layered *sharding* [Hong et al. 2021]

diferentes partições que eles participam, de forma direta e eficiente, ao invés de dividir a transação em várias sub-transações, como é feito no *sharding* completo.

As partições do Pyramid podem ser classificadas de duas formas de acordo com seu tipo, i-*shard* ou b-*shard*. Os i-*shards* possuem apenas os nós responsáveis por lidar com transações internas, enquanto os b-*shards* incluem nós que fazem uma ponte entre múltiplos i-*shards*, permitindo que resolvam as transações entre esses i-*shards*. Além desse papel, os nós pertencentes aos b-*shards* também são capazes de lidar com transações internas. No exemplo abaixo o b-*shard* C é responsável por processar as transações entre os i-*shards* A e B.

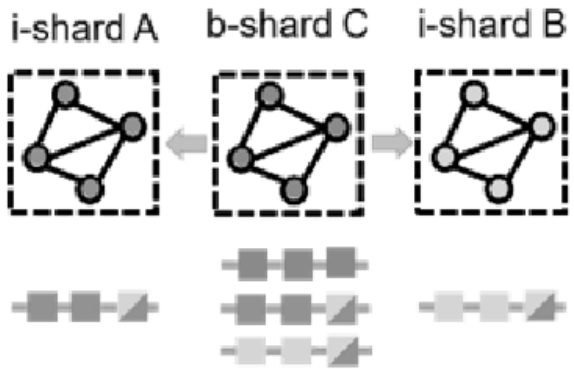

Figure 2.17. Funcionamento dos b-*shards* [Hong et al. 2021]

Embora esse modelo melhore o processamento de transações entre partições ele traz uma complexidade maior para estrutura, pelo fato dos nós terem funções diferentes, o que gera alguns desafios. O primeiro desafio é a construção das partições. Quantas partições são necessárias e quais nós devem ser designados a cada partição.

O segundo desafio é o protocolo de consenso, como gerar e verificar blocos. Os Sistemas que utilizam *sharding* geralmente adotam PoW ou um protocolo BFT. Por conta do *Layered sharding* algumas partições no pyramid são responsáveis por criar blocos para transações entre partições e os blocos gerados precisam ser enviados às partições correspondentes para confirmação, o que torna o design do protocolo de consenso mais desafiador.

2.6.1.1. Identity establishment and committee formation

A formação das partições é realizada em 3 etapas:

1. **Geração de aleatoriedade:** O funcionamento da Pyramid acontece em períodos de tempo fixos chamados de epoch, no começo de cada epoch a aleatoriedade é gerada atravez de um protocolo externo como verifiable random function [Micali et al. 1999] ou verifable delay function [Boneh et al. 2018];

2. **Participação:** Para cada nó, antes de se juntar a um epoch, um novo puzzle PoW é gerado baseado na sua chave pública e na aleatoriedade do epoch. Para participar de um epoch, o nó precisa resolver seu puzzle;

3. **Atribuição:** A cada nó admitido é designado uma ID de partição aleatoriamente, baseado na identidade do nó e na aleatoriedade gerada no epoch. A ID da partição pode representar um i-*shard* ou um b-*shard*, sendo que, os b-*shards* correspondem a um número de i-*shards*;

Na segunda etapa, após resolver o puzzle, o nó precisa colocar a sua solução numa *Blockchain* identidade para registrar sua identidade. Essa *Blockchain* é uma *Blockchain* baseada em PoW cuja função é registrar a identidade dos nós. A dificuldade do puzzle gerado para um nó que quer se juntar a um epoch é ajustada de acordo com a quantidade de nós registrados no epoch anterior.

Uma característica interessante do Pyramid é que podemos definir um parâmetro de escalabilidade w, onde, $0 < w <= 1$, e quanto maior w for, maior será a quantidade de b-*shards* criada. Dessa forma, aumentamos o throughput e diminuímos a latência das transações, porém, aumentamos o uso de armazenamento, dado que o b-*shard* armazena a sua blockchain e todas as quais ele faz a ponte.

2.6.1.2. Overlay setup for committees

O Pyramid considera que todos os nós da rede estão conectados como uma rede ponto-a-ponto parcialmente síncrona. E como os resultados da etapa de atribuição, visto na subseção anterior, são públicos e podem ser verificados usando a aleatoriedade e a Blockchain Identidade, os nós podem trocar mensagens com os outros membros do shard para o qual foi designado.

2.6.1.3. Intra-committee consensus

No pyramid, cada epoch consiste de turnos de consenso. A cada turno de consenso, um líder para a partição será aleatoriamente eleito. Se a partição for uma i-*shard* o bloco proposto inclui as transações internas e um consenso aos moldes das *Blockchain complete sharding* será executado.

2.6.1.4. Cross-shard transaction processing

Uma transação é um pagamento entre duas contas, chamadas remetente e receptor. Se o remetente e o receptor de uma transação estão localizados em diferentes partições, essa transação se trata de uma transação entre partições. A validade de cada transação pode ser dividida em source validity e result validity.

1. **Source Validity:** denota se o estado do remetente satisfaz a condição para a transação, isto é, o remetente possui dinheiro suficiente para realizar a transação;
2. **Result Validity:** denota se o estado do receptor satisfaz o resultado da transação,isto é, o receptor recebe dinheiro adequado;

Nós em um i-*shard* guardam apenas uma *Blockchain* enquanto nós em b-*shards* guardam múltiplas, por conta disso, no pyramid, nós que pertencem aos b-*shards* podem verificar a source validity e a result validity de transações entre partições.

Design de Blocos Cross-Shard

O cabeçalho inclui os hashes dos blocos pais dos i-*shards* relacionados e a raiz da árvore Merkle do corpo. Embora um bloco *cross-shard* válido possa ser proposto por qualquer nó em um b-*shard*, para garantir a consistência dos estados entre o b-*shard* e seus i-*shards* relacionados, o bloco *cross-shard* precisa ser confirmado pelos outros nós no b-*shard* e seus i-*shards* relacionados, através do consenso *layered sharding* que será visto a seguir. A figura 2.18 ilustra um bloco cross-shard relacionado aos i-shards A e B.

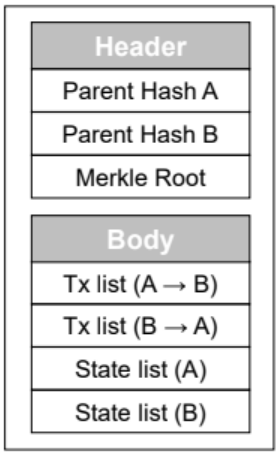

Figure 2.18. Bloco Cross-Shard dos Shards A e B [Hong et al. 2021]

Layered Sharding Consensus

Este consenso se dá em três etapas:

1. **Pré-preparação de bloco:** Nesta etapa o líder propõe o bloco de transação entre shard, e os nós chegam a um consenso através de um protocolo BFT chamado CoSi, que é um protocolo de assinatura coletiva que pode escalar eficientemente. Sendo assim um bloco de transações entre partições com uma assinatura coletiva de uma mais de dois terços de nós atesta que o b-*shard* concorda com isso em um ambiente bizantino;

2. **Preparação de bloco:** Após receber o bloco junto com a prova coletiva do b-*shard*, os nós em cada i-*shard* podem verificar a assinatura coletiva utilizando as chaves públicas salvas na *Blockchain* identidade. Então o i-*shard* também utilizará o protocolo de assinatura coletiva para chegar ao consenso e então enviar uma mensagem de Accept, junto com a hash do Header e a assinatura coletiva para o b-*shard*, caso contrário é enviada uma mensagem de Reject;

3. **commit do Bloco:** Na terceira e última etapa, os nós no b-*shard* inicializam um contador com o número de i-*shards* relacionados. E quando um nó recebe uma mensagem de Accept do i-*shard* associado ele decrementa seu contador e repassa a mensagem para os outros nós do b-*shard*. Quando o contador chega a 0, os nós no b-*shard* podem garantir que o bloco será confirmado. Mas para garantir a confirmação do bloco, uma assinatura coletiva adicional é feita, somente após isso o bloco será confirmado e uma mensagem de commit será enviada aos i-*shards*.

2.6.1.5. Epoch reconfiguration

O Pyramid utiliza reconfiguração total, ou seja, a cada epoch, todos os nós da rede serão designados para uma nova partição aleatoriamente, como apontado na seção Identity establishment and committee formation.

2.6.2. Lightchain

O LightChain [Hassanzadeh-Nazarabadi et al. 2019] replica a ideia de partição usando uma tabela hash distribuída de blocos, transações e nós. Esse modelo fragmenta as três dimensões previstas na partição completa, armazenamento, processamento e comunicação. Toda a cadeia é armazenada de forma distribuída e uniforme entre as instâncias de execução. Além disso, todo bloco ou transação é endereçável dentro da rede, e desta forma qualquer nó da rede pode solicitar um dos blocos ou transações. Na rede um nó que conheça apenas um bloco pode solicitar recursivamente pelo bloco antecessor até alcançar o bloco genesis. Da mesma forma, o LightChain permite solicitar o sucessor até atingir o bloco mais recente da cadeia.

A fragmentação do processamento se dá porque apenas alguns nós aleatórios são acionados para participar de cada processo de consenso ou consulta. Os nós, em momento algum, participam todos ao mesmo tempo de uma mesma operação.

Já a fragmentação da comunicação é consequência da estrutura, uma vez que cada nó possui somente o endereço de seus vizinhos. Entretanto, no LightChain, qualquer nó pode solicitar o estado ou saldo atual de qualquer outro nó da rede, de forma autenticada

e verificável, com complexidade de comunicação de $O(\log n)$. Isso é possível pois um nó não precisa percorrer a *Blockchain* para conseguir essas informações.

O protocolo de consenso proposto, Proof-of-Validation(PoV), possui a mesma fairness do Acordo byzantino. Fairness é a chance de um nó participar do consenso do bloco. Por outro lado, comparado ao acordo byzantino, PoV possui uma complexidade de comunicação bem reduzida. Neste protocolo, a instância que deseja adicionar um bloco na rede recebe uma lista de nós que participaram da decisão. A formação do grupo é feita de forma determinística a partir do estado da rede. Porém não é possível manipular essa escolha.

Apesar do artigo original não fazer menção expressa ao conceito de *shard*, pode-se verificar uma congruência com este conceito no consenso proposto, onde, a lista de nós que participará do consenso se assemelha a um comitê, sendo portanto, uma espécie de comitê ad hoc para cada validação de transação ou formação de bloco.

A Figura 2.19 ilustra a estrutura do LightChain. Nota-se que a cadeia, estrutura base de *Blockchain* se mantém. Porém cada instância armazena uma pequena parte dela, 1 a 2 blocos neste caso, reduzindo muito a necessidade de memória.

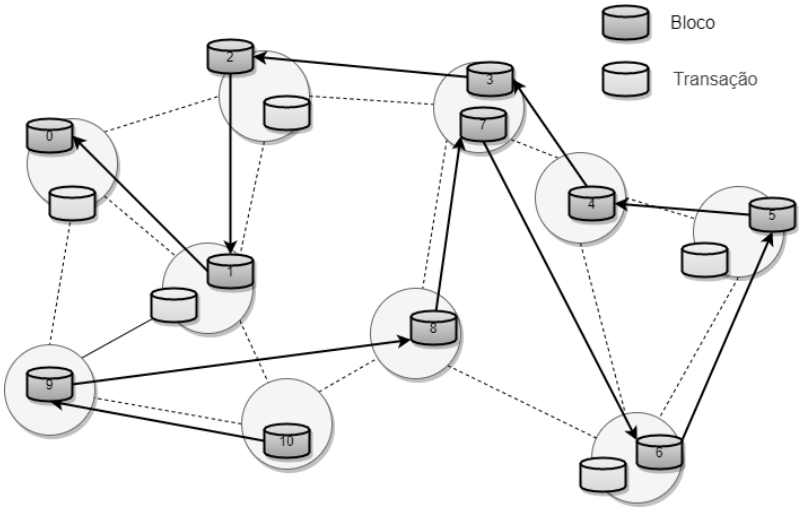

Figure 2.19. LightChain [Hassanzadeh-Nazarabadi et al. 2019]

2.6.3. Tabela Hash Distribuída

A DHT vem para resolver um problema fundamental em aplicações P2P, que é localizar de forma eficiente qual peer possui determinado dado. Consiste de uma tabela onde as informações de localização do objeto de dados são colocadas deterministicamente, nos peers com identificadores correspondentes à chave do objeto de dados. Os nós da rede têm associados a si uma sequência de bits chamado de identificador, e os dados recebem uma sequência chamada chave, oriunda do mesmo conjunto que os identificadores.

2.6.4. Segurança

Normalmente um modelo de partição precisa garantir que: Um bloco seja confirmado ou recusado, ou seja, não fique bloqueado indefinidamente por ação de invasores (vivaci-

dade). E que a distribuição das partições não é manipulável (safety). Porém isso não é necessário para esse modelo pelos motivos a seguir:

Safety: Não existe limite rígido para fração de nós maliciosos. Conceitualmente há segurança se houver ao menos um nó honesto no sistema.

Liveness: Há vivacidade enquanto houver mais de 50% de nós honestos no sistema.

Por fim é necessário provar que é improvável que um invasor consiga aprovar um bloco invalido. Para isso ele precisaria encontrar uma fração de invasores maior que $1/3$ no conjunto de nós do consenso em cada tentativa. Como não há limite para o tamanho do grupo e quanto maior o grupo menor a probabilidade um bloco invalido ser aprovado. É possível determinar um tamanho mínimo para que a chance aprovar tal bloco seja menor que $2^{-\lambda}$.

2.7. Conclusoes, Desafios de Pesquisa em Aberto e Referências Bibliográficas

Nesta seção iremos revisitar as propostas discutidas até então, procurando explicitar as diferenças entre elas observadas por meio das tabelas 2.1 e 2.2. Nesta comparação focamos em quatro aspectos: Protocol Settings, Intra-Committee Consensus, Inter-Committee Consensus e Safety and Performance [Wang et al. 2019].

2.7.1. Protocol Settings

- **formação do comitê**: se refere aos critérios usados para permitir que os nós se juntem ao comitê, que descreve os mecanismos para estabelecer a associação, por exemplo, associação com base em PoW ou PoS. Este é um aspecto importante dos sistemas descentralizados e não permissionados para impedir ataques Sybil. No entanto, para blockchain permissionadas, não precisamos lidar com ataques Sybil, uma vez que os sistemas permissionados operam em um ambiente relativamente confiável, onde os nós participantes são membros do comitê com base nesta política organizacional.

- **consistência**: A consistência mostra a probabilidade de que o sistema chegue a um consenso sobre o valor proposto, ela pode ser classificada como forte ou fraca. Em geral, os protocolos BFT clássicos oferecem consistência forte, mas estão sujeitos ao problema de escalabilidade.

- **modelo de rede**: O modelo de rede mostra a sincronia da rede de comunicação subjacente. Normalmente, as redes de comunicação podem ser categorizadas em três tipos: fortemente síncronas, parcialmente síncronas e assíncronas

2.7.2. Intra-Committee Consensus

- **configuração do comitê**: A configuração do comitê representa como os membros do comitê são atribuídos ao comitê em uma configuração de comitê único, por exemplo, ou os membros servem no comitê permanentemente (estático) ou são alterados em intervalos regulares (rotativos ou de troca) para os protocolos baseados em época.

- **incentivos**: Os incentivos mostram os mecanismos que mantêm os nós participantes motivados a participar do processo de consenso e seguir suas regras. Distinguimos os incentivos em dois aspectos: um é o processo de adesão e o outro é o processo de participação.
- **Líder**: Indica, dentro de um comitê específico, de onde vem o líder. Ele pode ser eleito entre o comitê atual (internamente), externamente ou flexível (por exemplo, por meio dos contratos inteligentes especificados).
- **Complexidade**: A complexidade mostra a complexidade da comunicação dentro de um comitê no nível da mensagem, onde não se refere ao número de nós participantes.

2.7.3. Inter-Committee Consensus

- **configuração inter-committee**: A configuração entre comitês mostra como os membros são designados aos comitês em um ambiente de comitês múltiplos, que pode ser estático ou dinâmico. Uma abordagem dinâmica é normalmente baseada na aleatoriedade gerada na época anterior.
- **Mediado**: indica como mediar as transações cross-shard. Pode ser opcionalmente mediado por um recurso externo, por exemplo, o cliente.
- **Incentivo**: indicam, para os mediadores, se eles receberão alguma recompensa por seus esforços de mediação.

2.7.4. Safety and Performance

Safety

- **TX Censorship Resistance**: O TX Censorship Resistance mostra a resiliência do sistema às transações propostas sendo suprimidas (ou seja, censuradas) por nós maliciosos envolvidos no processo de consenso;
- **resistência DoS**: Representa a resiliência dos nós envolvidos no consenso para ataques de negação de serviço (DoS). Se os participantes do protocolo de consenso forem conhecidos com antecedência, um adversário pode lançar um ataque DoS contra eles.
- **Modelo de Adversário**: Representa a fração de nós maliciosos ou defeituosos que o protocolo de consenso pode tolerar (por exemplo, o protocolo ainda funciona corretamente, apesar da presença de tais nós). Observe que, para diferentes modelos de adversário, ele pode ter diferentes taxas de resistência.

Performance

- **Throughput**: A taxa de transferência é a taxa máxima na qual as transações podem ser acordadas pelo protocolo de consenso;

- **Latência**: Representa o tempo que leva desde o momento em que uma transação é proposta até que se chegue a um consenso sobre ela.
- **Escalabilidade**: Mostra se o sistema tem a capacidade de atingir um maior rendimento quando o consenso envolve um número maior de nós.

Nas tabelas 2.1 e 2.2 considere:

- ✓: possui a propriedade;
- ✗: não possui a propriedade;
- *: possui parcialmente;
- !: não consta a informação;
- -: não se aplica à proposta.

Table 2.1. Tabela Comparativa Elastico, ChainSpace, OmniLedger e RapidChain

		Elastico	**ChainSpace**	**OmniLedger**	**RapidChain**
Protocol Settings	Formação do Comitê	PoW	Flexible	PoW/PoX	Offline PoW
	Consistência	Forte	Forte	Forte	Forte
	Modelo de Rede	Partial Sync.	Async	Partial Sync.	Sync.
Intra-Committee Consensus	Configuração do Comitê	Full Swap	Flexible	Rolling (Subset)	Partical Swap
	Incentivos (Join,Participate)	(✓, ✗)	(✗, ✗)	(✓, ✗)	(✓, ✗)
	Líder	Internal	Internal	Internal	Internal
	Complexidade	$O(n^2)$	$O(n^2)$	$O(n)$	$O(n)$
Inter-Committee Consensus	Configuração Inter-Committee	Dynamic (Random)	✗	Dynamic (Random)	Dynamic (Random)
	Mediado	!	✗	Client	✗
	Incentivo	!	✗	✗	✗
Safety	TX Censorship Resistance	✗	✓	✓	✓
	Resistência DoS	✓	✓*	✓	✓
	Modelo de Adversário	33%	33%	33%	33%
Performance	Throughput	16 block in $110s$ (1)	350 tx/s (2)	$\approx 10k\ tx/s$ (3)	$\approx$ 7.300 tx/s (4)
	Latência	110s for 16 blocks	<1s	$\approx 1s$	8.7s for 7300tx
	Escalabilidade	✓	✓	✓	✓

(1): 100 nós por *shard* e 16 *shards* no total; (2): 4 nós por *shard* e 15 *shards* no total; (3): 72 nós por *shard*(12.5% adversários) e 25 *shards* no total; (4): 250 nós por *shard* e 4000 nós no total.

2.7.5. Desafios de Pesquisa em Aberto

Para melhor compreensão dos desafios de pesquisa em aberto na área de Blockchain, iremos dividir tais desafios em dois tópicos, Performance e Segurança, como proposto em [Yu et al. 2020] e [Huang et al. 2021].

Table 2.2. Tabela Comparativa CycLedger, Sharper, Pyramid e LightChain

		CycLedger	SharPer	Pyramid	LightChain
Protocol Settings	Formação do Comitê	Cryptographic Sortition	Geographical Distance	PoW	-
	Consistência	Forte	Forte	Forte	Forte
	Modelo de Rede	Sync.	Async.	Partial Sync.	Sync
Intra-Committee Consensus	Configuração do Comitê	Full Swap	Static	Full Swap	-
	Incentivos (Join,Participate)	(✓, ✓)	(-, -)	(✗, ✗)	(-, ✓)
	Líder	External	Internal	Internal	Internal
	Complexidade	$O(n)$	!	!	O(log n)
Inter-Committee Consensus	Configuração Inter-Committee	Dynamic (Random)	Static	Dynamic (Random)	-
	Mediado	✗	Client	✗	-
	Incentivo	✗	✗	✗	-
Safety	TX Censorship Resistance	✓	-/✓	✓	✓
	Resistência DoS	✓*	-/✓	✓	✓
	Modelo de Adversário	33%	50%/33%	33%	33%
Performance	Throughput	!	$\approx 27k\ tx/s$ (5)	$\approx 12k\ tx/s$ (6)	!
	Latência	!	240ms for 27000tx	$\approx 5s$	!
	Escalabilidade	✓	✓	✓	✓

(5):20 nodes, 10% cross-shard transactions ; (6): 20 shards, 4000 nós(12.5% adversários), w(parâmetro de escalabilidade) = 0.5.
Na coluna da proposta SharPer, temos separado por "/" os valores considerando Crash Only Nodes e Byzantine Nodes.

2.7.5.1. Performance

Escalabilidade: A escalabilidade ainda é um grande desafio para a maioria dos sistemas de *Blockchain*. Por exemplo, os protocolos de consenso PBFT emitem um número $O(n^2)$ de mensagens, onde n é o número de participantes. O grande número de mensagens torna a escalabilidade irreal.

Mecanismos Resilientes para Sharding: Os mecanismos resilientes para fragmentar blockchains ainda estão faltando [Huang et al. 2021].

Performace de Transações Cross-Shard: Embora vários protocolos de fragmentação tenham sido propostos, esses protocolos só podem suportar no máximo 1/3 de adversários. Assim, protocolos de acordo bizantino mais robustos precisam ser elaborados. Além disso, todos os protocolos baseados em sharding incorrem em latências e tráfegos cross-shard adicionais devido às transações entre shard. Portanto, o desempenho do cross-shard em termos de taxa de transferência, latência e outras métricas deve ser bem garantido em estudos futuros.

Transações Cross-Chain: Este tópico diz respeito a interoperabilidade entre *Blockchains*, prevendo que, no futuro, teremos uma grande variedade de sistemas *Blockchain* e eles precisarão se comunicar entre si, tal tópico foi pouco explorado e teve o pontapé inicial dado por [Jin et al. 2018].

Soluções de aceleração assistidas por hardware para redes *Blockchain*: Para melhorar o desempenho de *blockchains*, por exemplo, para reduzir a latência de confirmação de transação, algumas tecnologias de rede avançadas, como *Remote Direct Memory Access* (RDMA) e placas de rede de alta velocidade, podem ser exploradas para acelerar o acesso a dados entre as mineradoras em redes *blockchain*.

Otimização de desempenho em diferentes camadas de rede Blockchain: A rede *blockchain* é construída sobre as redes P2P, que incluem várias camadas típicas, como camada mac, camada de roteamento, camada de rede e camada de aplicativo. Os protocolos baseados em BFT funcionam essencialmente para a camada de rede. Portanto, melhorias de desempenho podem ser alcançadas ao propor vários protocolos, algoritmos e modelos teóricos para outras camadas da rede *blockchain*.

Redes de *Big Data* assistidas por *Blockchain*: Embora big data e *blockchain* tenham várias métricas de desempenho que são contrárias entre si. Por exemplo, *big data* é uma tecnologia de gerenciamento centralizado com ênfase na preservação da privacidade orientada para diversos ambientes de computação. Os dados processados pela tecnologia de *big data* devem garantir a não redundância e a arquitetura não estruturada em uma rede de computação em grande escala. Em contraste, a tecnologia *blockchain* se baseia em uma arquitetura descentralizada, transparente e imutável, na qual o tipo de dados é simples, estruturado e altamente redundante. Além disso, o desempenho de *blockchain* requer escalabilidade e o paradigma de computação fora da cadeia. Portanto, como integrar essas duas tecnologias e buscar o benefício mútuo uma da outra é uma questão em aberto que merece estudos aprofundados. Por exemplo, os tópicos de pesquisa em potencial incluem como projetar uma nova arquitetura de *blockchain* adequada para tecnologias de *big data* e como quebrar as ilhas de dados isoladas usando *blockchain*, garantindo os problemas de privacidade de *big data*.

2.7.5.2. Segurança

Preservação de privacidade em *Blockchains*: A maioria dos trabalhos existentes nessa categoria está discutindo a preservação da privacidade e a segurança que a *blockchain* fornece para as aplicações. O fato é que a segurança e a privacidade também são questões críticas da *blockchain* em si. Por exemplo, a privacidade das transações poderia ser hackeada por invasores. Embora estudos sejam escassos, é possível apontar o Monero [Alonso et al. 2020] como solução que se destaca no assunto. Fazendo uso extenso de criptografia, o Monero propõe a preservação da privacidade através do sigilo sistêmico de quatro informações: o endereço público do emissor e do receptor, o montante transferido, e o endereço IP dos envolvidos através de rede sobreposta I2P ou Tor.

Mecanismos anti-criptojacking para mineradores mal-intencionado: Mineradores maliciosos fazendo uso de códigos que confiscam os recursos de hardware, como capacidade computacional e memória, dos usuários da web para utilizá-los em seu

proveito. De acordo com [Tahir et al. 2019] tais ataques ocorrem em navegadores web, portanto, é necessário desenvolver mecanismos e estratégias anti-cryptojacking para proteger os usuários normais da web e manter a rede *blockchain* justa.

Problemas de segurança de *blockchains* de criptomoeda: Poucos esforços foram dedicados à investigações teóricas para as questões de segurança de *blockchains* de criptomoeda. Por exemplo, a exploração de punição e cooperação entre mineradores em várias cadeias é um tópico interessante para *blockchains* de criptomoeda. Portanto, é plausível o surgimento de perspectivas mais amplas de modelagem dos comportamentos de atacantes e contra-atacantes no contexto de ataques de *blockchains* monetárias.

Lei Geral de Proteção de Dados Pessoais - LGPD e *Blockchains*: Uma das principais premissas da LGPD (ou sua equivalente europeia General Data Protection Regulation - GDPR) é o controle do usuário sob seus dados, com prerrogativa de exclusão. Isto vai de encontro à imutabilidade das blockchains. Alguns estudos propõem o desacoplamento do controle e dos dados, transferindo este ultimo para um armazenamento off-chain, enquanto o primeiro permanece na blockchain. Como o armazenamento off-chain não será imutável, alcançar o mesmo nível de descentralização e segurança na parte off-chain é um desafio. [Truong et al. 2020] [Daudén-Esmel et al. 2021]

References

[Al-Bassam et al. 2017] Al-Bassam, M., Sonnino, A., Bano, S., Hrycyszyn, D., and Danezis, G. (2017). Chainspace: A sharded smart contracts platform. *arXiv preprint arXiv:1708.03778*.

[Alonso et al. 2020] Alonso, K. M. et al. (2020). Zero to monero.

[Amiri et al. 2019] Amiri, M. J., Agrawal, D., and Abbadi, A. E. (2019). Sharper: Sharding permissioned blockchains over network clusters. *arXiv preprint arXiv:1910.00765*.

[Andrychowicz and Dziembowski 2015] Andrychowicz, M. and Dziembowski, S. (2015). Pow-based distributed cryptography with no trusted setup. In *Annual Cryptology Conference*, pages 379–399. Springer.

[bit2me 2020] bit2me (2020). O que é sharding?

[Boneh et al. 2018] Boneh, D., Bonneau, J., Bünz, B., and Fisch, B. (2018). Verifiable delay functions. In *Annual international cryptology conference*, pages 757–788. Springer.

[Bootle et al. 2016] Bootle, J., Cerulli, A., Chaidos, P., and Groth, J. (2016). Efficient zero-knowledge proof systems. In *Foundations of security analysis and design VIII*, pages 1–31. Springer.

[CasaDaMoeda 2015] CasaDaMoeda (2015). Origem do dinheiro.

[Castro et al. 1999] Castro, M., Liskov, B., et al. (1999). Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186.

[Centieiro 2021] Centieiro, H. (2021). What's proof of elapsed time.

[Corso 2019] Corso, A. (2019). *Performance analysis of proof-of-elapsed-time (poet) consensus in the sawtooth blockchain framework*. PhD thesis, University of Oregon.

[Danezis et al. 2014] Danezis, G., Fournet, C., Groth, J., and Kohlweiss, M. (2014). Square span programs with applications to succinct nizk arguments. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 532–550. Springer.

[Daudén-Esmel et al. 2021] Daudén-Esmel, C., Castellà-Roca, J., Viejo, A., and Domingo-Ferrer, J. (2021). Lightweight blockchain-based platform for gdpr-compliant personal data management. In *2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*, pages 68–73.

[Douceur 2002] Douceur, J. R. (2002). The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer.

[Eichmann 2018] Eichmann, K. (2018). The future client of an energy utility company will be a machine.

[Fernando Wosniak Steler 2017] Fernando Wosniak Steler, A. H. C. (2017). Tudo o que você queria saber sobre blockchain e tinha receio de perguntar.

[Hassanzadeh-Nazarabadi et al. 2019] Hassanzadeh-Nazarabadi, Y., Küpçü, A., and Özkasap, Ö. (2019). Lightchain: A dht-based blockchain for resource constrained environments. *arXiv preprint arXiv:1904.00375*.

[Hong et al. 2021] Hong, Z., Guo, S., Li, P., and Chen, W. (2021). Pyramid: A layered sharding blockchain system. *IEEE INFOCOM*.

[Huang et al. 2021] Huang, H., Kong, W., Zhou, S., Zheng, Z., and Guo, S. (2021). A survey of state-of-the-art on blockchains: Theories, modelings, and tools. *ACM Computing Surveys (CSUR)*, 54(2):1–42.

[IntelSGX 2021] IntelSGX (2021). Intel software guard extensions.

[Jake Frankenfield 2020] Jake Frankenfield, S. G. A. (2020). Proof of elapsed time (poet) (cryptocurrency).

[Jin et al. 2018] Jin, H., Dai, X., and Xiao, J. (2018). Towards a novel architecture for enabling interoperability amongst multiple blockchains. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1203–1211. IEEE.

[King and Nadal 2012] King, S. and Nadal, S. (2012). Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19:1.

[Kokoris-Kogias et al. 2018] Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., and Ford, B. (2018). Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 583–598.

[L. 2019] L., K. (2019). The blockchain scalability problem the race for visa-like transaction speed.

[Lamport et al. 2001] Lamport, L. et al. (2001). Paxos made simple. *ACM Sigact News*, 32(4):18–25.

[Luu et al. 2016] Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., and Saxena, P. (2016). A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30.

[Maymounkov and Mazieres 2002] Maymounkov, P. and Mazieres, D. (2002). Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer.

[Mearian 2019] Mearian, L. (2019). Sharding: What it is and why many blockchain protocols rely on it.

[Micali et al. 1999] Micali, S., Rabin, M., and Vadhan, S. (1999). Verifiable random functions. In *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*, pages 120–130. IEEE.

[Muratov et al. 2018] Muratov, F., Lebedev, A., Iushkevich, N., Nasrulin, B., and Takemiya, M. (2018). Yac: Bft consensus algorithm for blockchain. *arXiv preprint arXiv:1809.00554*.

[Nakamoto 2008] Nakamoto, S. (2008). Bitcoin whitepaper. *URL: https://bitcoin. org/bitcoin. pdf-(: 17.07. 2019)*.

[Pahlajani et al. 2019] Pahlajani, S., Kshirsagar, A., and Pachghare, V. (2019). Survey on private blockchain consensus algorithms. In *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*, pages 1–6. IEEE.

[Rebello et al. 2019] Rebello, G., Camilo, G., Silva, L., Souza, L., Guimarães, L., Alchieri, E., Greve, F., and Duarte, O. (2019). Correntes de blocos: Algoritmos de consenso e implementação na plataforma hyperledger fabric. *Sociedade Brasileira de Computação*.

[REVOREDO 2019] REVOREDO, T. (2019). Blockchain: tudo que você precisa saber (potencial e realidade).

[Syta et al. 2016] Syta, E., Jovanovic, P., Kogias, E. K., Gailly, N., Gasser, L., Khoffi, I., Fischer, M. J., and Ford, B. (2016). Scalable bias-resistant distributed randomness. Cryptology ePrint Archive, Report 2016/1067. https://eprint.iacr.org/2016/1067.

[Tahir et al. 2019] Tahir, R., Durrani, S., Ahmed, F., Saeed, H., Zaffar, F., and Ilyas, S. (2019). The browsers strike back: Countering cryptojacking and parasitic miners on the web. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 703–711. IEEE.

[Truong et al. 2020] Truong, N. B., Sun, K., Lee, G. M., and Guo, Y. (2020). Gdpr-compliant personal data management: A blockchain-based solution. *IEEE Transactions on Information Forensics and Security*, 15:1746–1761.

[Wang et al. 2019] Wang, G., Shi, Z. J., Nixon, M., and Han, S. (2019). Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 41–61.

[Wang and Wang 2019] Wang, J. and Wang, H. (2019). Monoxide: Scale out blockchains with asynchronous consensus zones. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, pages 95–112.

[Wood et al. 2014] Wood, G. et al. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32.

[Yu et al. 2020] Yu, G., Wang, X., Yu, K., Ni, W., Zhang, J. A., and Liu, R. P. (2020). Survey: Sharding in blockchains. *IEEE Access*, 8:14155–14181.

[Zamani et al. 2018] Zamani, M., Movahedi, M., and Raykova, M. (2018). Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 931–948.

[Zhang et al. 2020] Zhang, M., Li, J., Chen, Z., Chen, H., and Deng, X. (2020). Cycledger: A scalable and secure parallel protocol for distributed ledger via sharding.

