Blockchain e a Revolução do Consenso sob Demanda

Chapter · May 2018

CITATIONS
12

READS
1,225

6 authors, including:

Fabíola Greve
Universidade Federal da Bahia
76 PUBLICATIONS **394** CITATIONS
SEE PROFILE

Jauberth Abijaude
Universidade Estadual de Santa Cruz
8 PUBLICATIONS **17** CITATIONS
SEE PROFILE

Antonio Augusto T R Coutinho
Universidade Estadual de Feira de Santana
9 PUBLICATIONS **49** CITATIONS
SEE PROFILE

Italo Valcy da Silva Brito
Universidade Federal da Bahia
7 PUBLICATIONS **30** CITATIONS
SEE PROFILE

Capítulo

5

Blockchain e a Revolução do Consenso sob Demanda

Fabíola Greve, Leobino Sampaio, Jauberth Abijaude, Antonio Coutinho, Ítalo Valcy, Sílvio Queiroz

Abstract

Blockchain introduces a new on-demand consensus paradigm where the set of P2P network nodes confirm the order in which the transaction blocks are aggregated into the chain of blocks, providing a reliable, secure, scalable, and immutable distributed environment for the transactions' execution on the Internet. The blockchain eliminates the need for a trusted third party and digitally creates a decentralized trust entity. The disruptive character of the technology is vast and applications are emerging in several areas: finance, health, arts, government, etc., beyond computing itself. A number of challenges exist and involve security, privacy, storage, availability, performance improvement, scalability, sustainability, etc.. This chapter presents the blockchain, its main elements, properties, models, algorithms and challenges. Special emphasis will be given to distributed consensus protocols for the maintenance of the replicated state machine, as well as the use of the technology in the context of computer networks, fog computing and IoT.

Resumo

A blockchain introduz o novo paradigma do consenso sob demanda, onde o conjunto de nós da rede P2P concorda com a ordem em que blocos de transações vão sendo agregados à corrente de blocos, proporcionando um ambiente distribuído confiável, seguro, escalável e imutável para a realização e armazenamento de transações na Internet. A blockchain elimina a necessidade de uma terceira parte confiável e cria digitalmente uma entidade de confiança descentralizada. O caráter disruptivo da tecnologia é imenso e aplicações estão surgindo em diversas áreas: finanças, saúde, artes, governo, etc., além da própria computação. Diversos desafios estão postos e envolvem segurança, privacidade, armazenamento, disponibilidade, melhoria de desempenho, crescimento em escala, sustentabilidade, etc.. Este capítulo apresenta a blockchain, seus principais elementos, propriedades, modelos, algoritmos e desafios. Especial ênfase será dada aos protocolos de consenso distribuído para manutenção da máquina de estados replicada, além do uso da tecnologia no contexto de redes de computadores, computação em névoa e IoT.

5.1. Introdução

Blockchain é uma tecnologia emergente que oferece suporte distribuído confiável e seguro para realização de transações entre participantes que não necessariamente têm confiança entre si e que estão dispersos em larga escala numa rede P2P. É considerada uma tecnologia disruptiva, pois cria digitalmente uma entidade de confiança descentralizada, eliminando a necessidade de uma terceira parte de confiança. Dessa forma, pode substituir entidades certificadoras e centralizadoras das transações de negócios, tais como bancos, governos, cartórios, etc. O potencial de transformação é imenso e aplicações estão surgindo a partir desta tecnologia em inúmeros setores: finanças, saúde, artes, governo, além da própria computação: protocolos de redes, nuvem e névoa, IoT, etc.

A blockchain implementa uma *máquina de estados replicada* para a manutenção consistente de um estado global compartilhado por um conjunto de pares distribuídos numa rede P2P. Todos os nós possuem e mantêm uma réplica do registro de transações efetuadas, materializado na forma de um livro-razão (*ledger*) distribuído, que é imutável, pode ser verificado e auditado, e está sempre disponível. O conjunto e a ordem em que as transações são executadas é acordada por todos os participantes da rede, através da realização de um protocolo de *consenso byzantino*, tolerante a ações de nós maliciosos, passíveis de subverter o sistema. O consenso é um elemento fundamental para o desenvolvimento de sistemas confiáveis e seguros, pois possibilita com que os participantes de uma computação concordem com as ações que serão realizadas, com o intuito de manter a consistência do sistema e de fazê-lo progredir. Mecanismos de *criptografia* são empregados para garantir a autoridade, autenticidade, não-repúdio, integridade das transações, bem como os requisitos de segurança de todo o sistema.

A blockchain é resultado de uma engenhosa combinação de técnicas robustas provenientes da computação distribuída confiável (tolerância a falhas bzyantinas, sistemas P2P), criptografia (chave assimétrica, funções hash, desafios criptográficos) e teoria dos jogos (mecanismos de incentivos). Agrega elementos eficazes para a implementação de um sistema de acordo em escala global, trazendo respostas para importantes desafios computacionais, dentre os principais, destacamos: (i) realização de consenso numa rede aberta, com participantes desconhecidos, em escala planetária; (ii) tolerância a falhas byzantinas, com participantes anônimos; (iii) resistência ao ataque de duplo-gasto (*double-spending*), garantindo que ativos transacionados (como moedas digitais) não sejam gastos duplamente, para além do seu valor em posse; (iv) resistência a ataques Sybil, garantindo que usuários maliciosos não poderão se personificar em outros (a menos que tenham acesso a sua chave privada); (v) garantia da auditabilidade, autenticidade, não-repúdio e integridade em escala global de todas as transações validadas e armazenadas no livro-razão distribuído.

Na origem da blockchain, está o protocolo do Bitcoin, proposto por Satoshi Nakamoto [Nakamoto 2008], que entrou em operação em 2009. O artigo seminal propõe uma rede P2P onde transações com a criptomoeda (moeda digital) bitcoin, propostas por clientes, são recebidas por servidores (precisamente, mineradores), que irão decidir, através de um protocolo de consenso byzantino à base de desafios criptográficos, sobre a ordem em que serão realizadas e armazenadas permanentemente numa corrente de blocos (*block chain*), replicada em cada servidor. A prova de conceito oferecida pelo Bitcoin

foi suficiente para uma crescente adesão à rede de confiança digital descentralizada que se criou. Seu grande feito? Eliminar a terceira parte de confiança, necessária nas transações financeiras. Para além da tecnologia, a blockchain do Bitcoin traz uma ruptura nas transações de negócios, ao introduzir o mecanismo de incentivos digitais e criptomoedas em diversos níveis (sistemas, serviços, aplicações, etc.) e relações (econômicas, públicas, sociais, etc.). De fato, o consenso do Nakamoto (com sua mineração inovadora) e a geração da moeda (como mecanismo de incentivo) integram-se numa simbiose salutar para viabilizar o acordo entre pares que não se conhecem, e adicionalmente, a oferta de serviços de consenso sob demanda na Internet.

A blockchain original do Bitcoin incorpora uma máquina de estados bem simplificada, com elementos voltados essencialmente para transações com a moeda bitcoin. O projeto Ethereum, idealizado por Vitalik Buterin [Buterin 2014], adota o conceito de *contratos inteligentes*, que permite a execução de uma máquina de Turing completa. Os contratos inteligentes expressam uma lógica de transações mais sofisticada, possibilitando a implementação de aplicações descentralizadas e autônomas, em diversos níveis e escopos. Na evolução de projeto da blockchain, são então destacadas três fases [Bashir 2017]: a Blockchain 1.0 envolve o lançamento do Bitcoin em 2008, com as primeiras implementações das criptomoedas, e um ecossistema de aplicações e pagamentos com o ativo digital. A Blockchain 2.0 inicia-se com a proposta inovadora dos contratos inteligentes em 2013, e toda gama de aplicações financeiras possíveis. A Blockchain 3.0 caracteriza a adoção da tecnologia blockchain no benefício de aplicações em diversas áreas, para além da financeira: governo, comércio, artes, saúde, cidades digitais, etc..

Há diversos projetos, com forte investimento da indústria, visando a implementação de plataformas de blockchain para desenvolvimento de aplicações robustas e descentralizadas nos mais variados segmentos. Para além do Bitcoin e Ethereum, que oferecem uma *blockchain pública*, aberta, sem necessidade de controle dos participantes, há uma gama de setores que requerem participação controlada. As plataformas para *blockchain privada* ou *federada* atendem melhor a interesses corporativos e requerem a autenticação dos participantes na rede; dentre elas, destaca-se o Hyperledger Fabric, em estágio avançado de desenvolvimento. De fato, tal característica da blockchain, no que diz respeito ao tipo de composição e filiação dos membros à rede, vai nortear importantes aspectos da sua infra-estrutura (replicação máquina de estados, consenso, segurança, comunicação), bem como do projeto dos sistemas nas demais camadas (dados, contratos inteligentes, serviços, aplicações) e modelos de negócios associados.

5.1.1. Propriedades da Blockchain

As principais propriedades da tecnologia blockchain que contribuem de forma inovadora para o desenvolvimento de aplicações e sistemas são as seguintes:

- *Descentralização*: As aplicações e sistemas são executados de maneira distribuída, através do estabelecimento de confiança entre as partes, sem a necessidade de uma entidade intermediária confiável. Esse é o principal motivador para o crescente interesse na blockchain.

- *Disponibilidade e Integridade*: Todo o conjunto de dados e transações são replica-

dos em diferentes nós de maneira segura, de forma a manter o sistema disponível e consistente.

- *Transparência e Auditabilidade*: Todas as transações registradas no livro-razão são públicas, podendo ser verificadas e auditadas. Além disso, os códigos da tecnologia costumam ser abertos, passíveis de verificação.

- *Imutabilidade e Irrefutabilidade*: As transações registradas no livro-razão são imutáveis. Uma vez registradas não podem ser refutadas. Atualizações são possíveis a partir da geração de novas transações e realização de novo consenso.

- *Privacidade e Anonimidade*: É possível oferecer privacidade aos usuários sem que os terceiros envolvidos tenham acesso e controle dos seus dados. Na tecnologia, cada usuário gerencia suas próprias chaves e cada nó servidor armazena apenas fragmentos criptografados de dados do usuário. Transações são até certo ponto anônimas, com base no endereço dos envolvidos na blockchain.

- *Desintermediação*: A blockchain possibilita a integração entre diversos sistemas de forma direta e eficiente. Assim, é considerada um conector de sistemas complexos (sistemas de sistemas), permitindo a eliminação de intermediários de maneira a simplificar o projeto dos sistemas e processos [Xu et al. 2016].

- *Cooperação e Incentivos*: Oferta de modelo de negócios à base de incentivos, à luz da teoria dos jogos. O consenso sob demanda passa a ser oferecido como serviço em diversos níveis e escopos.

5.1.2. Organização do Capítulo

Neste capítulo, apresentamos a blockchain, seus principais conceitos, desafios e aplicações em alguns contextos. As pesquisas na área são recentes e a tecnologia emergente. Nosso foco, então, será de apresentação dos elementos fundamentais, com descrição dos modelos, algoritmos e mecanismos. Especial ênfase será dada aos protocolos de consenso distribuído para manutenção da máquina de estados replicada. Além disso, a blockchain pública do Bitcoin será exaurida. Como contraponto, apresenta-se o Hyperledger Fabric, um projeto promissor para blockchains privadas ou federadas. O consenso sob demanda oferecido pela blockchain abre perspectivas para desenvolvimento de sistemas robustos e seguros em diversos escopos da computação, particularmente, em redes de computadores, IoT e computação em névoa. Alguns resultados recentes nesses contextos serão apresentados.

O capítulo dividi-se, então, em três partes: a primeira aborda os elementos fundamentais para o entendimento da blockchain: criptografia, consenso e livro-razão (Seção 5.2), bem como o consenso distribuído para blockchains públicas e privadas (Seção 5.3). A segunda parte apresenta as plataformas Bitcoin (5.4) e Hyperledger Fabric (5.5). A terceira parte explora exemplos de adoção da blockchain para as áreas de redes de computadores, IoT e névoa (Seções 5.6, 5.7, 5.8), e finalmente discute-se desafios e perspectivas na área (Seção 5.9).

5.2. Elementos Fundamentais (Criptografia, Consenso, Livro-Razão)

Nesta seção, serão apresentados os principais elementos da blockchain, que contribuem para suas propriedades: imutabilidade, integridade, atualidade, transparência, disponibilidade, descentralização, desintermediação.

5.2.1. Criptografia

A Blockchain apoia-se fortemente na Criptografia para satisfazer os requisitos de segurança do sistema e das aplicações. Dentre os recursos mais utilizados, destacam-se os *resumos criptográficos e as assinaturas digitais*. Nesta seção, elas serão definidas e, ao longo do capítulo, suas aplicações serão destacadas. Para maiores detalhes, o leitor deve reporta-se ao excelente livro [Narayanan et al. 2016].

5.2.1.1. Resumos Criptográficos

As *funções hash* incorporam algoritmos que fazem a dispersão de dados variáveis x em valores h com quantidade fixa de dígitos, tal que $h = hash(x)$. São excelentes artifícios de representação e transformação de dados de volume variável ($x =$ arquivos, mensagens, etc.) num dado de valor único e com pequena quantidade de dígitos (e.g., 256 dígitos). Além disso, tais algoritmos devem ser eficientes computacionalmente, de tal forma que, se x é uma string de n bits, então $hash(x)$ tem complexidade $O(n)$.

As *funções hash criptográficas* (ou *resumos criptográficos*) são unidirecionais e dificilmente permitem a recuperação do valor original x, a partir do hash h. Assim, são funções hash que, adicionalmente, satisfazem as seguintes propriedades: (1) resistência à colisão (*collision resistance*) e (2) ocultação (*hiding*). Nos próximos parágrafos, vamos explicar cada uma dessas propriedades e suas abrangências.

Uma colisão irá ocorrer quando, ao aplicar a função hash para duas entradas distintas, x e y, o mesmo valor h for encontrado. Ou seja, dada a função hash H, então $H(x) = H(y) = h$. Como o universo de valores de saída h é muito menor do que o universo de valores de entrada, colisões evidentemente serão possíveis. Entretanto, o que se deseja é que elas sejam raras, que sejam evitadas quando as funções forem aplicadas.

Definição 1 ***Resistência à Colisão.*** *Um função hash H é resistente à colisão, quando for inviável achar dois valores x e y, tal que* $x \neq y$ *e* $H(x) = H(y)$.

Podemos deduzir que se a função *hash H* é resistente à colisão, então, dificilmente iremos encontrar dois valores x e y diferentes e com mesmo hash h. Logo, podemos utilizar tais funções para, por exemplo, representar adequadamente resumos de informações (ou *message digests*). Tais resumos podem ser usados em aplicações para atestar, por exemplo, a integridade dos arquivos que tenham sido armazenados ou transmitidos na rede. Assim, dado um arquivo A, e seu $h = H(A)$; podemos usar h como um resumo ímpar do arquivo A original. Como h tem tamanho fixo pequeno (e.g., 256 bits) e A costuma ter tamanho muito grande, em muitos contextos, costuma-se substituir A por h.

Uma característica importante que deve ser associada às funções *hash* é a necessidade de ocultação do valor original x. Ou seja, não seria viável deduzir o valor de x, dado

que o valor $h = H(x)$ é conhecido. Entretanto, é importante perceber que, se o universo de possibilidades de x for muito pequeno, ficaria muito fácil deduzir x. Por exemplo, se x tem apenas duas possibilidades (quente ou frio), bastaria um adversário aplicar *hash* nos dois valores para saber a qual h estaria associado. Dessa forma, espera-se que x seja escolhido de um conjunto de valores bastante aleatório.

Um truque que se costuma utilizar para que a ocultação seja possível, qualquer que seja a entrada x, é o de considerar um valor r, escolhido de um conjunto de valores bastante aleatório e, assim, aplicar a função hash $H(r||x)$ à concatenação ($||$) das duas entradas r e x. Ou seja, se r vem de uma distribuição, onde não há um valor específico provável de ocorrer, fica quase impossível chutar o valor de x, na aplicação da função H. Dessa maneira, a ocultação será possível.

Definição 2 ***Ocultação (Hiding).*** *Um função hash H está ocultando se: quando um valor secreto r é escolhido de um universo com distribuição aleatória muito espalhada, então, dado $H(r||x)$, é inviável identificar x.*

A ocultação é fundamental para estabelecer compromissos (*commitments*). Esses são uma forma de se comprometer digitalmente com uma mensagem *msg*, dado que a *msg* será oculta e, quando necessário, poderá ser revelada. A *msg* poderia ser associada a um contrato, que será colocado num envelope e lacrado. Uma vez firmado o compromisso (ou seja, o contrato), ele não poderá ser alterado. A posteriori, o conteúdo do envelope pode ser revelado e verificado.

Esquema 1 Compromisso (*Commitment*). *Consiste em dois algoritmos:*

- $com := commit(msg, nonce)$ – Função que recebe a *msg* e um valor randômico secreto *nonce* e retorna um valor *com* atestando o compromisso firmado. Em criptografia, o termo *nonce* designa um valor a ser utilizado apenas uma vez.
- $match := verify(com, msg, nonce)$ – Função que verifica se o compromisso *com* é válido. Ou seja, retorna um valor booleano $true$ caso $com == commit(msg, nonce)$ e $falso$, caso contrário.

Seguindo a analogia do contrato no envelope, a função *com* seria o lacre com o envelope, que todos têm acesso. A função $verify$ corresponde a abertura do envelope para verificação. Para cada novo contrato, um *nonce* é utilizado. Como a *msg* deve ficar oculta através de *com*, não podemos recuperá-la, logo o compromisso deve *ocultar* a *msg*. Ao mesmo tempo, uma vez firmado o compromisso, não podemos alterá-lo, assim a *msg* não pode sofrer mudanças; logo deve-se *ligar* a *msg* ao compromisso, de tal forma que é impossível encontrar dois pares $(msg, nonce)$ e $(msg', nonce')$, tal que $msg \neq msg'$, de forma a satisfazer $commit(msg, nonce) == commit(msg', nonce')$. Desta maneira, os compromissos precisam assegurar essas duas propriedades de segurança: *ocultação* e *ligação*. Para implementar o esquema de compromisso, pode-se usar diretamente um resumo criptográfico H, onde *nonce* é valor aleatório de 256 bits. Assim:

$$commit(msg, nonce) := H(nonce||msg).$$

Como H satisfaz ocultação e é resistente à colisão, estaremos satisfazendo as propriedades do compromisso, conforme abaixo:

• *Ocultação*: Dado $H(nonce||msg)$, é inviável encontrar msg.

• *Ligação*: É inviável encontrar dois pares $(msg, nonce)$ e $(msg', nonce')$, tal que $msg \neq msg'$ e $H(nonce||msg) == H(nonce'||msg')$.

Como veremos, compromissos, resumos criptográficos e nonces serão fundamentais para assegurar as propriedades de segurança da blockchain.

5.2.1.2. Assinaturas Digitais

As assinaturas digitais devem atender às mesmas propriedades de assinaturas manuais em documentos. Assim, somente você pode assinar um documento, mas, qualquer um pode verificar a sua autenticidade. Além disso, não é possível forjar a sua assinatura, de tal forma a reutilizá-la em algum outro contexto. Ou seja, assinaturas devem ser irrefutáveis.

As assinaturas digitais são implementadas através de *criptografia de chave assimétrica*. Nesse caso, além da chave privada ou secreta (sk) usada para assinar os documentos, também será utilizada uma chave pública (pk). A chave privada é secreta e não deve ser revelada para ninguém. Já a chave pública deve ser revelada para qualquer um que queira atestar a autenticidade da assinatura.

Esquema 2 Assinatura Digital *Consiste em três algoritmos:*

- $(sk, pk) :=$ *generateKeys(keysize)* – O método *generateKeys*() recebe um tamanho de chave (*keysize*) na entrada e retorna um par de chaves pública (pk) e privada ou secreta (sk).
- $sig := sign(sk, msg)$ – O método *sign*() recebe uma mensagem msg e uma chave secreta (sk) na entrada e retorna a assinatura sig daquela mensagem sob sk.
- $isValid := verify(pk, msg, sig)$ – O método *verify* recebe uma chave pública (pk), uma mensagem msg e uma assinatura (sig) na entrada, e retorna um valor booleano: $isValid = true$ se sig é uma assinatura válida para msg sob pk; $isValid = false$, caso contrário.

A assinatura digital deve satisfazer às seguintes propriedades:

• *Autenticidade*: É possível verificar assinatura válida, da seguinte maneira:

$$verify(pk, msg, sign(sk, msg)) == true.$$

• *Existencialmente Infalsificável (Unforgeable)*: Assinatura não pode ser forjada.

A autenticidade garante a autoria da assinatura, que poderá ser verificada à posteriori. Como ela é infalsificável, qualquer alteração no documento torna a assinatura inválida, logo ela preserva a *integridade* do documento; também garante o *não repúdio*, e o assinante não pode negá-la, depois de produzida. Do ponto de vista prático, costuma-se assinar o resumo criptográfico H de uma mensagem msg, no lugar da mensagem propriamente dita. Isto porque mensagens costumam ser muito longas e resumos têm quantidade fixa de bits. Assim, adota-se $sig := sign(sk, H(msg))$. Como os hashes são resistentes à colisão, podemos usar $H(msg)$ como um resumo singular para msg.

Os *endereços* na blockchain, importantes para identificar os nós nas transações, são resumos criptográficos das chaves públicas dos envolvidos. O Bitcoin e várias outras blockchains utilizam como resumo criptográfico a SHA256 (cuja saída tem 256 bits), aplicada duplamente *double-SHA256*. O Bitcoin utiliza um tipo especial de esquema de assinatura digital chamado *Elliptic Curve Digital Signature Algorithm* (ECDSA), tipo "secp256k1", que provê 128 bits de segurança [Antonopoulos 2017, Bashir 2017].

5.2.1.3. Infraestrutura de Chave Pública

Uma *infraestrutura de chave pública* - ICP (*PKI - Public Key Infrastructure*) é uma cadeia hierárquica de confiança que viabiliza a emissão de certificados digitais para identificação virtual. Seu funcionamento baseia-se na derivação de confiança a partir do uso de assinaturas digitais. Portanto, caso confie-se em uma autoridade certificadora, todos os certificados emitidos pela instituição, que estejam válidos (dentro do período de validade, não foram revogados, etc.), serão também confiáveis. Uma ICP administra a geração, distribuição e revogação de chaves e certificados digitais.

O *certificado digital* é um documento eletrônico assinado digitalmente por uma autoridade certificadora, que contém diversos dados sobre o emissor e o seu titular. Sua função é vincular uma pessoa ou uma entidade a uma chave pública. No contexto da blockchain privada os certificados digitais estabelecem credenciais de usuário e assinaturas para as mensagens garantindo, assim, que a mensagem não tenha sido alterada.

Normalmente, uma ICP possui uma autoridade certificadora (*CA - Certificate Authority*), uma autoridade de Registro (*RA - Registration Authority*) e um banco de dados de certificados. A RA é responsável pela autenticação e confirmação da legitimidade dos dados, certificados ou outros artefatos utilizados para assegurar a solicitação de um usuário para um ou mais certificados que refletem a identidade ou propriedade do mesmo [Nguyen et al. 2018]. Uma CA, sob recomendação de uma RA, emite certificados digitais para usos específicos e é certificada diretamente ou hierarquicamente para uma CA raiz.

5.2.2. Consenso

Consenso é um problema fundamental em computação distribuída confiável, pois permite com que participantes distribuídos coordenem as suas ações, de forma a alcançar decisões comuns, e assim garantir a manutenção da consistência dos seus estados (*safety*) e o progresso do sistema (*liveness*), apesar da existência de falhas [Greve 2005]. O consenso fatoriza a classe de problemas em que existe necessidade de acordo. Nesse sentido, o consenso é seminal para a tecnologia blockchain, pois possibilita a obtenção do acordo sobre o próximo bloco a ser agregado à blockchain.

Falhas Byzantinas. Na blockchain, o consenso deve ser alcançado apesar da existência de processos maliciosos, que podem agir arbitrariamente de forma a subverter a computação. Ou seja, considera-se o modelo de falhas Byzantinas [Lamport et al. 1982], no qual, diferentemente de um processo *correto*, um processo *falho* pode exibir qualquer comportamento, podendo parar, omitir envios e entregas de mensagens, ou desviar arbitrariamente de sua especificação.

Definição 3 ***Consenso Byzantino***. *No consenso, cada processo p_i propõe um valor v_i e todos os processos corretos devem "decidir" por um único valor v, dentre os propostos. Formalmente, ele é definido pelo seguinte conjunto de propriedades:*

• *Acordo*: dois processos corretos não decidem diferentes valores;

• *Terminação*: todo processo correto decide de maneira definitiva;

• *Validade*: se um processo correto decide o valor *v*, então *v* foi proposto por algum processo. Para a blockchain, especificamente, *v* será o *bloco de transações* a ser adicionado ao livro razão.

5.2.2.1. Replicação Máquina de Estados (RME)

Replicação Máquina de Estados (RME) ou *State Machine Replication (SMR)* é uma técnica de tolerância a falhas que permite manter um serviço replicado de forma consistente. Na RME a execução das diversas requisições, feitas por clientes aos servidores que mantêm as réplicas, é realizada deterministicamente, na mesma ordem total, de tal maneira que réplicas corretas irão passar pelo mesmos estados de transição, gerando por conseguinte os mesmos resultados a cada nova requisição. Importante observar que cada réplica mantém uma cópia completa do serviço replicado.

A RME deve assim atender aos seguintes requisitos [Schneider 1990]:

• *Estado inicial*: todas as réplicas corretas devem iniciar a partir de um mesmo estado;

• *Determinismo*: todas as réplicas corretas ao executarem a mesma operação sobre o mesmo estado produzem o mesmo resultado e mudança de estado subsequente;

• *Coordenação*: todas as réplicas corretas executam a mesma sequência de operações.

A coordenação da RME é assegurada por um protocolo de consenso, conhecido como ***atomic broadcast*** ou ***difusão atômica*** ou ***difusão em ordem total*** [Lamport 1998, Défago et al. 2004]. Ele assegura que as réplicas corretas realizarão o mesmo conjunto de requisições e na mesma ordem total. No modelo de falhas Byzantinas, em que até f réplicas podem se comportar maliciosamente, um cliente identifica que sua requisição foi executada com sucesso quando recebe $f+1$ respostas idênticas dos servidores.

A blockchain implementa assim uma máquina de estados replicada para a manutenção consistente do livro razão distribuído em cada um dos servidores que compõem a rede P2P. Nesse caso, os blocos de transações são incorporados ao livro razão segundo uma ordem total, através de um protocolo de difusão atômica. Tal consenso é necessário para evitar que blocos concorrentes sejam agregados em ordens diferentes ao livro razão por cada um dos servidores, comprometendo assim a execução consistente das transações.

Resultado de Impossibilidade do Consenso. É importante destacar que numa rede clássica, cujos *n* participantes do sistema são identificados, e onde não há limites temporais para a transmissão de mensagens e realização das ações dos processos, ou seja, o *sistema é assíncrono*, o consenso não tem solução determinística, mesmo na presença de uma única falha benigna de processo: falha por parada (*crash*) [Fischer et al. 1985]. Este resultado é conhecido como FLP, devido ao nome dos autores: Fisher, Lynch e Paterson. Assim, resolvê-lo no contexto da blockchain, onde o conjunto de participantes

não é bem conhecido e estão sujeitos a falhas byzantinas, tem complexidade ainda maior. Este resultado fundamental provoca a impossibilidade de se resolver outros problemas, onde a necessidade de acordo é recorrente, como a própria *difusão atômica* [Chandra and Toueg 1996]. Na origem desses resultados negativos, encontramos a mesma causa: as incertezas decorrentes do modelo assíncrono no que diz respeito à detecção de falhas.

Incorporando Sincronia ao Sistema. É interessante notar que, desde que algum grau de sincronia é acrescido ao sistema, vários desses problemas passam a ter solução determinística. Em particular, num sistema assíncrono estendido com detectores de falhas não-confiáveis,[Chandra and Toueg 1996] provaram que existe uma solução determinística ao problema do consenso com falhas por parada, desde que uma maioria de processos ($f < n/2$) seja correta. Além disso, a difusão atômica e o consenso são equivalentes em qualquer modelo de falhas, ou mais precisamente em todo modelo em que a difusão confiável pode ser resolvida [Chandra et al. 1996]. Isto significa que todo algoritmo para o consenso pode ser aplicado para resolver a difusão atômica e vice-versa.

5.2.2.2. Variações de Consenso

Paxos, introduzido por [Lamport 1998], é o algoritmo de consenso mais conhecido para implementação da RME. Ele propõe a extensão do sistema assíncrono com o oráculo de líder, uma abstração equivalente aos detectores de falhas [Chu 1998]. Um oráculo ou detector de líder da classe Ω (Ômega) satisfaz a propriedade de liderança definitiva, em que existe um instante após o qual, todo processo correto sempre confia no mesmo processo correto no sistema (i.e., o mesmo líder) [Lampson 2001]. Na resolução do consenso, o líder será fundamental para coordenar os demais na obtenção do acordo. A classe de detectores de falhas $\diamondsuit\mathscr{S}$ (*eventually strong*) e o detector Ω possuem o mesmo poder computacional e reúnem as condições de sincronia minimais para resolução do consenso em sistemas assíncronos, sujeitos a falhas por parada [Chandra et al. 1996].

O consenso byzantino, introduzido por [Lamport et al. 1982] sob a designação do ***Problema dos Generais Byzantinos***, tem solução num sistema em que, no máximo, $f < n/3$ processos se comportam de maneira maliciosa. Lamport demonstra ser impossível atingir consenso byzantino se essa condição não é satisfeita. O modelo de sistema mais comumente usado na solução do consenso byzantino é o *parcialmente síncrono* de [Dwork et al. 1988]. Neste, existem limites temporais para a transmissão de mensagens na rede e para a realização de qualquer computação, no entanto, esses limites não são conhecidos e não se sabe quando passam a valer. Ele modela bem o comportamento da Internet, em que o sistema passa por períodos de estabilidade (quanto a computação converge dentro de limites temporais esperados), embora tenha comportamento assíncrono, com frequência.

Byzantine Fault Tolerance (BFT). Ao longo dos anos, um número bastante expressivo de protocolos de consenso foi proposto para tolerância a falhas por parada [Guerraoui and Raynal 2007] e falhas byzantinas [Guerraoui et al. 2008], além do desenvolvimento de RME [Charron-Bost et al. 2010]. Muitas soluções focam nos algoritmos e aspectos teóricos, sendo que apenas um pequeno grupo aborda as questões mais práticas. Mais recentemente, com a necessidade da incorporação de técnicas de tolerância a falhas no núcleo dos grandes serviços da Internet (nuvem, *storage*, etc.), oferecidos

pelos grandes players da indústria, um interesse crescente voltou-se para a sua engenharia. Diversos sistemas, como o Chubby da Google [Chandra et al. 2007] e o Apache Zookeeper [Hunt et al. 2010], apoiam-se fortemente em soluções à base do Paxos [Rao et al. 2011, Gafni and Lamport 2003] para prover uma plataforma de sincronização de estados em provedores de nuvens e *data centers* com redundância, alta disponibilidade, robustez e resiliência a falhas por parada.

PBFT (*Practical Byzantine Fault-Tolerant*) foi o primeiro consenso byzantino a abordar questões práticas e um dos mais conhecidos [Castro and Liskov 1999, Castro and Liskov 2002]. O BFT-Smart [Bessani et al. 2014] é um outro projeto promissor em bom estágio de maturidade. Como veremos na Seção 5.5, PBFT é oferecido pelo Hyperledger Fabric como camada de acordo (ordenação de transações). Além disso, o BFT-Smart [Sousa et al. 2018] também foi recentemente incorporado ao projeto.

Consenso Probabilístico. Uma outra alternativa, visando contornar o resultado de impossibilidade FLP, consiste em mudar a especificação do problema e enfraquecer as propriedades do consenso. Nessa linha, surgiram protocolos com garantias de acordo probabilísticas [Ben-Or 1983, Cachin et al. 2000]. Nesse caso, faz-se uso de um oráculo randômico: um *gerador de números aleatórios*. O oráculo randômico de [Ben-Or 1983] soluciona uma versão não-determinística do consenso, assegurando a obtenção de decisões com probabilidade 1, mesmo num cenário em que o adversário conhece tudo acerca do sistema. O consenso[Nakamoto 2008] da blockchain do Bitcoin provê acordo probabilístico. Durante um período de instabilidade, o sistema pode vir a decidir mais de um valor, ocasionando os chamados *forks*; porém, após um período de instabilidade na rede, a decisão tende a ser única.

Como a blockchain é um sistema de confiança per si, ela se beneficia diretamente de todo o avanço que a área de BFT teve ao longo das últimas décadas. Por outro lado, ela está promovendo um revival na área, e os protocolos e sistemas BFT passam a ser adaptados para esse novo e desafiante contexto da blockchain [Cachin and Vukoli'c 2017, Gramoli 2017, Bano et al. 2017].

5.2.2.3. Teorema CAP (*Consistency, Partition, Availability*)

Safety x Liveness. Importante notar que, para contornar o resultado FLP, há duas abordagens: ou enfraquece-se o modelo de sincronia ou enfraquece-se o problema. No primeiro caso, obtemos consenso com garantias de acordo determinísticas, já no segundo caso, teremos consenso com garantias probabilísticas. O consenso byzantino tradicional e a BFT oferecem garantias determinísticas e apoiam-se em redes parcialmente síncronas para assegurar o progresso; neste caso, enquanto a rede não oferece as condições de estabilidade (sincronia) para que se decida, a terminação (*liveness*) ficará comprometida, mas não o acordo (*safety*). Assim, o modelo BFT sempre assegura a consistência do sistema, mas, pode comprometer o seu progresso. Já o consenso randômico, incluindo o do Nakamoto, assegura a decisão (ordenação de blocos), garantindo a terminação; entretanto, pode comprometer o acordo (sobre a corrente de blocos) e assim a consistência do sistema.

Teorema CAP (*Consistency, Partition, Availability*). O antagonismo entre consistência e progresso nos sistemas tolerantes a falhas foi generalizado por [Brewer 2000] através do Teorema CAP (provado por [Gilbert and Lynch 2012]). De maneira geral, um

sistema distribuído apresenta um compromisso entre as seguintes propriedades:

- Consistência (*Consistency*): todas as requisições obtêm os dados mais recentes; ou seja, o estado do sistema é consistente;
- Disponibilidade (*Availability*): toda requisição obtém uma resposta; ou seja, o sistema está operante, executando as ações;
- Tolerância a Partições (*Partition tolerance*): o sistema continua em operação, apesar de particionamentos temporários da rede, devido a falhas.

O Teorema CAP estabelece que não é possível satisfazer plenamente as três propriedades ao mesmo tempo. Assim, alguns sistemas sacrificam uma delas em detrimento das outras duas. Sistemas baseados em BFT clássico, bem como as BFT Blockchain, privilegiam fortemente a consistência, em detrimento da disponibilidade. O mesmo acontece com bancos de dados não-estruturados (NoSQL), no relaxamento das propriedades ACID [Frank et al. 2014] e oferta de baixa disponibilidade. Já a Blockchain do Bitcoin sacrifica a consistência em prol da disponibilidade e tolerância a partições. No caso, ela oferece consistência após um tempo (*eventual consistency*), e permite a obtenção, de forma gradativa, ao longo do tempo, de um estado consistente.

5.2.3. Livro-Razão Distribuído (*Distributed Ledger*)

O livro-razão (*ledger*) é a estrutura de dados imutável, em que transações são registradas e o estado global do sistema é mantido. O *ledger* mantém-se completamente replicado em todos os nós da rede P2P. Logo, o livro-razão distribuído é replicado e imutável. Por sua vez, as transações devem obedecer as mesmas propriedades ACID de banco de dados. Nesse sentido, a criptografia e o consenso são os elementos chaves para manutenção da autenticidade, integridade, consistência e disponibilidade do livro-razão. Nessa seção, o livro-razão será detalhado [Narayanan et al. 2016, Antonopoulos 2017].

Apontador hash *(hash pointer)* é um apontador para aonde encontra-se armazenado o dado d e o seu hash criptográfico $H(d)$. Assim, além de permitir a recuperação do dado d, o apontador hash permite a sua verificação, através de $H(d)$. Eles podem ser usados para construir quaisquer estrutura de dados com encadeamento, como listas, árvores, grafos direcionados acíclicos (DAGs), etc.

Em particular, apontadores hash são empregados para criar uma estrutura de blocos encadeada, de nome ***blockchain***, conforme Figura 5.1. Nesta, temos um conjunto de blocos de dados encadeados numa lista de hashes. Como a função H é resistente à colisão (Definição 1) e satisfaz ocultação (Definição 2), podemos usar H para verificar se o bloco b_k dentro da blockchain sofreu modificação. Se houver alguma mudança em b_k o apontador $H(b_k)$, presente no bloco b_{k+1}, estará inconsistente com o valor do dado armazenado em b_k. Assim, se o *header* $H()$ de toda estrutura é guardado de forma segura, pode-se verificar quando a blockchain sofreu violação, mesmo quando adversários tentam modificar todo o conjunto de apontadores armazenados na lista encadeada.

A tecnologia blockchain é assim formada por blocos encadeados por apontadores hashes numa lista. O primeiro bloco da lista é designado bloco *genesis*. Cada bloco contém um conjunto de transações. Estas, por sua vez, são estruturados numa estrutura de

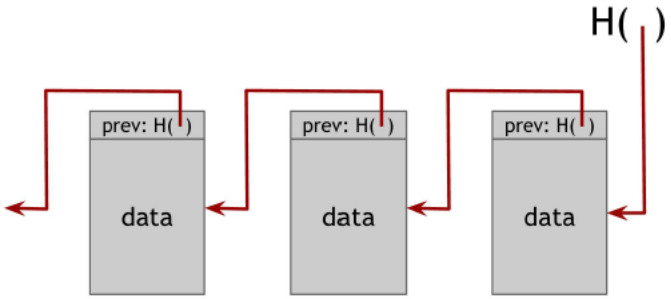

Figure 5.1. Estrutura de Blockchain com Apontadores Hash (*fonte: [Narayanan et al. 2016]*)

árvore binária de apontadores hash. No caso da rede Bitcoin essa estrutura é a ***árvore de Merkle***. A Figura 5.2 ilustra essa formação. Nesta árvore, no último nível, estão as folhas que contêm os dados (especificamente, apontadores para as transações propriamente ditas); no penúltimo nível, os pais possuem apontadores hash para esses dados. Em seguida, em cada nível, os pais apontadores são agrupados dois a dois, até que se alcance a raiz da árvore. O hash da raiz da árvore é então armazenado de forma segura dentro do *header* do bloco, em conjunto com demais informações. Pela propriedade dos apontadores hash, se houver qualquer modificação nos dados da árvore, ela será identificada pela verificação dos hashes. Além disso, o espaço necessário para o seu armazenamento é muito pequeno, comparativamente ao tamanho dos dados.

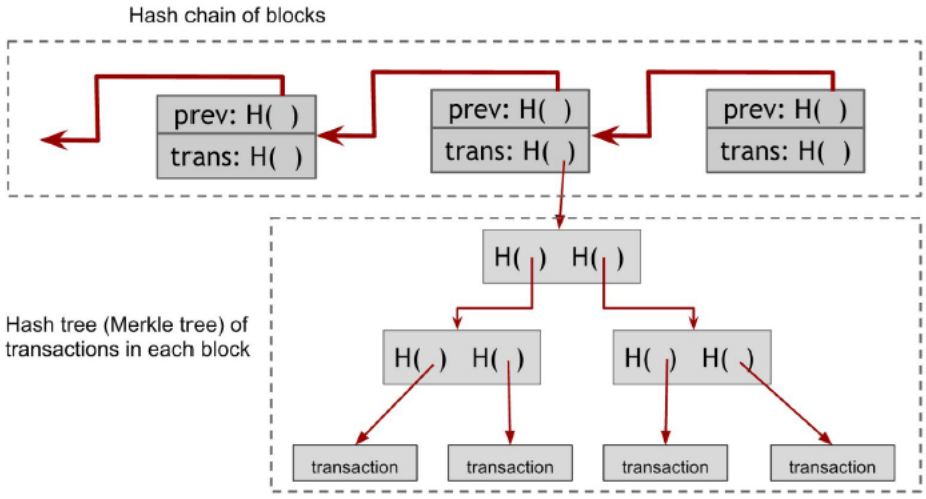

Figure 5.2. Blockchain com Árvore Binária de Merkle (*fonte: [Narayanan et al. 2016]*)

Uma outra vantagem de uso da árvore de Merkle é a facilidade de prova de filiação (*proof of membership*), sendo possível verificar se uma transação t_i pertence a árvore T em tempo $O(log(n))$, onde n é o conjunto de nós da árvore. Assim, para provar que $t_i \in T$, basta apresentar os hashes do caminho em que t_i é folha seguindo até a raiz da árvore. Esse caminho de autenticação ou *caminho de merkle* terá o hash do complemento dos pares, a cada nível. Na Figura5.3, o caminho de merkle para se verificar que a transação

H_K pertence à árvore consiste nos quatro hashes seguintes: $H_L, H_{IJ}, H_{MNOP}, H_{ABCDEFG}$. Dessa forma, qualquer nó da rede, de posse do caminho de merkle, poderá calcular os hashes complementares em $O(log(n))$ operações.

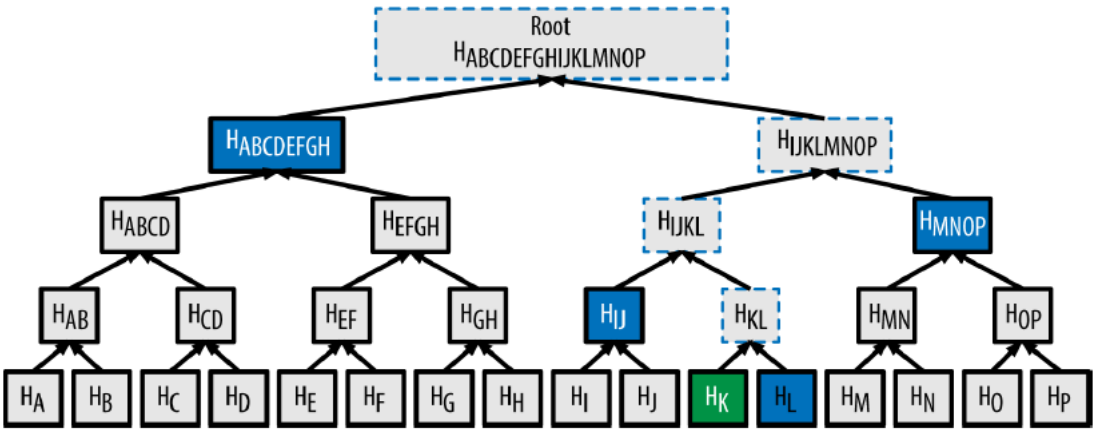

Figure 5.3. Caminho de Merkle (*fonte: [Antonopoulos 2017]*)

5.2.3.1. Transações

Uma transação estabelece uma sequência de operações sobre estados. Ela incorpora uma transferência de ativo ou, de forma geral, um contrato inteligente. Num caso básico, a transação envolve uma assinatura digital do emissor (que detém o ativo) e o endereço do receptor, além de entradas e saídas com o valor transacionado. Se a transação refere-se a contratos inteligentes pode envolver a invocação de um método, com suas entradas; ou, se a intenção é o *deploy* do contrato, deve conter o código de inicialização.

A Figura 5.4 apresenta um exemplo simplificado de quatro transações envolvendo bitcoins (para facilitar, cada bloco apresenta uma única transação). Em cada transação, deve-se representar as entradas (INPUTS) e saídas (OUTPUTS), tal como num livro contábil. As entradas referenciam os hashes das transações anteriores que têm relação com a atual. No exemplo, a transação t_1 de geração da moeda, efetua a transferência de 25 bitcoins a Alice. A transação t_2 representa uma transferência de 17 bitcoins de Alice para Bob; para tanto, Alice deve assinar t_2, indicando a sua autorização. Na entrada de t_2, o valor $1[0]$ aponta para o hash de t_1, indicando que o ativo deve ser transferido de t_1, conforme saída 0. Na transação t_4, Alice transfere 6 bitcoins para David, para tanto indica como entrada o hash da transação t_2, conforme saída 1.

Validação de uma transação t: envolve sobretudo: (i) a verificação das assinaturas, (ii) a confirmação dos valores existentes a partir dos hashes das transações anteriores referenciadas e, ainda, (iii) a confirmação de que o valor não foi anteriormente gasto por nenhuma outra transação, ou seja, existem fundos para a transação; nesse caso, será necessário efetuar uma busca na blockchain entre o bloco da transação referenciada até o último bloco da estrutura. As transações são validadas, de forma independente, por cada nó da rede blockchain, e essa característica contribui para a descentralização do processo.

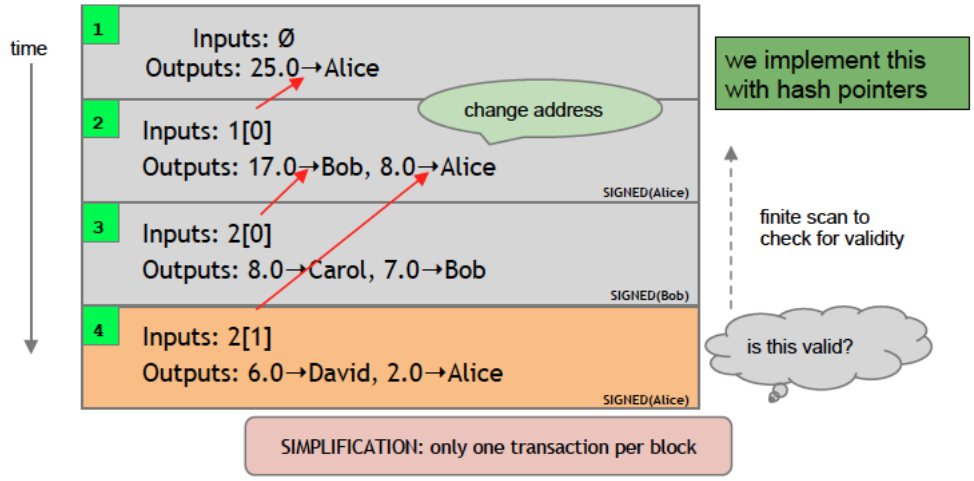

Figure 5.4. Esquema de Conjunto de Transações em Bitcoin (*fonte: [Narayanan et al. 2016], slides: Montresor (UniTN)*)

5.2.3.2. Blocos

O bloco contém um *header* com informações necessárias para a manutenção da corrente e sua validação. A Figura 5.5 apresenta um esboço simplificado de bloco no Bitcoin. Os principais campos são: (i) *prev*: o apontador hash do bloco anterior da corrente, (ii) *mrkl_root*: o apontador hash da árvore de Merkle do conjunto de transações do bloco, (iii) *nonce*, um valor variável a ser obtido na resolução do enigma criptográfico durante o processo de consenso (via mineração), de tal maneira que o *hash* do bloco seja válido, i.e., seja menor que o *alvo* estipulado pela prova de trabalho. Para além desses campos (aqui representados), o header do bloco também possui um *timestamp*, que indica o tempo cronológico em que o bloco é acrescido à blockchain, e a *dificuldade alvo*, estabelecido pela prova de trabalho.

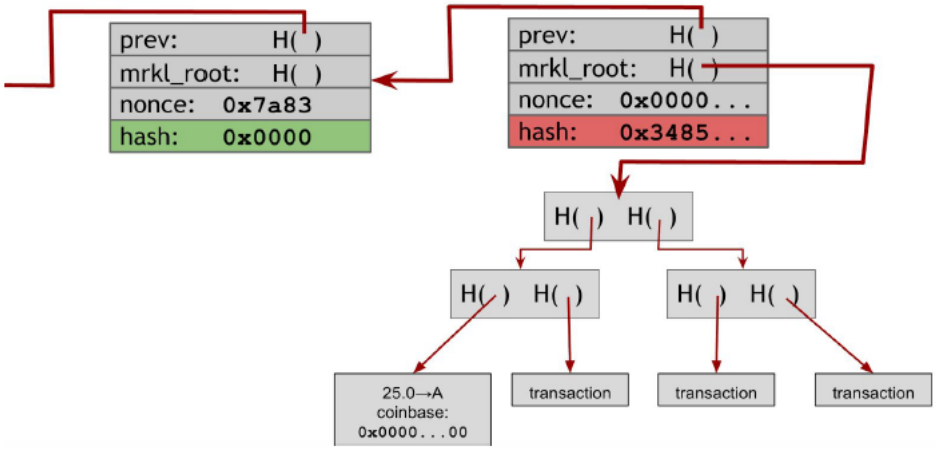

Figure 5.5. Bloco com Header Simplificado na Bitcoin (*fonte: [Narayanan et al. 2016]*)

Além do header, o bloco também possui *metadados*, que ficam armazenados separadamente da blockchain e não são transmitidos na rede. São eles: o *hash do header do bloco*, contendo o resumo que identifica univocamente o bloco perante toda a rede; a *altura* ou posição do bloco na corrente. O primeiro bloco (genesis) tem altura 0, os demais terão a altura do bloco anterior acrescida de 1. Ou seja, sua altura estabelecerá a quantidade de blocos criados até então após o bloco genesis[Antonopoulos 2017].

Validação de um bloco b: consiste em verificar (i) se a sua estrutura é bem formada, (ii) o seu *hash* é válido (atende ao desafio), (iii) o seu tamanho está dentro do limite aceito pela rede, (iv) o conjunto de transações dentro do bloco é válido, (v) a primeira transação (e somente a primeira) é a *coinbase transaction* – que incorpora a geração de novas criptomoedas no sistema, além de atuar como mecanismo de recompensa. Os blocos são validados, de forma independente, por cada nó da rede blockchain, e essa característica contribui para a descentralização do processo.

5.3. Blockchain Pública (sem permissão) X Federada/Privada (com permissão)

As redes blockchain podem ser categorizadas em dois grupos: blockchain pública ou *permissionless* (sem permissão, de acesso aberto), e blockchain federada/privada ou *permissioned* (com permissão e acesso controlado).

Na *blockchain pública*, o conjunto de nós da rede P2P é desconhecido e sua composição (*membership*) é dinâmica, permitindo entradas e saídas aleatórias de nós. Como os nós não precisam de identificação, costumam ser anônimos. A blockchain pode atuar em escala planetária, sem controle dos seus participantes, que inclusive não se confiam mutuamente. Nesta categoria, encontra-se a rede do Bitcoin, do Ethereum e de diversas outras criptomoedas [Bashir 2017, Antonopoulos 2017].

A *blockchain federada* ou *privada* tem uma composição conhecida, formada por *n* processos, cujas entradas e saídas estão sujeitas a permissões. Os nós são identificados, autenticados e autorizados. A blockchain irá atender melhor a interesses corporativos ou privados, onde os participantes têm papéis bem definidos e podem inclusive se organizar em grupos. Nesta categoria, encontra-se o Hyperledger Fabric [Cachin et al. 2016] e alguns outros projetos [Cachin and Vukoli'c 2017].

Importante salientar que uma gama nova de consenso para a blockchain tem sido proposta, principalmente como suporte a criptomoedas. Entretanto, muitos têm negligenciado as dificuldades associadas à resolução do problema, decorrentes, inclusive, dos resultados de impossibilidade, e têm apresentado protocolos inconsistentes. A comunidade BFT vem alertando para tais distorções [Cachin and Vukoli'c 2017, Gramoli 2017]. Como veremos nesta seção, o consenso da blockchain é fortemente voltado ao tipo de rede a que se destina (com permissão ou sem permissão).

5.3.1. Consenso para Blockchain Pública

Devido às diversas incertezas em relação aos participantes da rede, as blockchains públicas costumam adotar consensos abertos baseados em mineração, onde os mineradores competem entre si pela liderança do consenso, a partir de poder computacional, poder

de posse sobre a criptomoeda ou outros poderes de relevância para a eleição e que não podem ser monopolizados (de tal forma que os mesmos nós sempre saem vitoriosos).

Como a rede é aberta e ad-hoc, costuma-se retribuir o trabalho dos mineradores com incentivos financeiros, que são concretizados através das criptomoedas. Os incentivos são fundamentais na estratégia montada para tolerar ataques byzantinos, como ataques em conluio para subverter a rede. A governança dessa rede costuma estabelecer um conjunto de regras mínimas para a sua existência, além de favorecimento mútuo, de quem usa e de quem serve.

O consenso à base de provas resolve o desafio fundamental de realização de acordo em escala global num ambiente sujeito a falhas byzantinas, com participantes desconhecidos e anônimos (sem necessidade de autentificação). Atualmente, a prova de trabalho (*proof of work*) é uma das poucos abordagens de consenso bem sucedidas e resilientes a ataques Sybil[Douceur 2002] (ataques de personificação, quando usuários maliciosos se fazem passar por outros).

5.3.1.1. Consenso do Nakamoto e Prova de Trabalho – *Proof of Work* (PoW)

Como a blockchain implementa uma máquina de estados replicada, todos os nós da rede P2P enviarão requisições de transações para serem ordenadas pela máquina ao longo do tempo. As transações serão agrupadas em blocos e um consenso será executado em *rodadas* para possibilitar a ordenação total dos blocos (conjunto de transações). O consenso do Bitcoin proposto pelo Nakamoto [Nakamoto 2008] utiliza um *oráculo randômico* para eleição de um líder, que irá coordenar o consenso naquela rodada. Assim, para se eleger, o nó precisa realizar uma *prova de trabalho* ou *proof of work (PoW)*, resolvendo um desafio criptográfico, que exigirá bastante poder computacional. Essa eleição descentralizada no jargão do Bitcoin é chamada *mineração*, e os nós que a realizam, *mineradores*.

O nó vencedor da eleição randômica (a mineração) é aquele que primeiro vence o desafio. Quando isso ocorre, o vencedor se auto-declara líder para os demais, propagando o seu valor, ou seja o seu bloco. Esse bloco somente será aceito por cada nó se for válido. Assim, a rodada do consenso chega ao seu término. É importante observar que a rodada do consenso termina sem que os nós interajam para efetuar confirmação do valor proposto pelo líder. Eles simplesmente validam e agregam o bloco aceito ao livro-razão distribuído, seguindo para uma próxima rodada de consenso. Todo esse processo de eleição e de validação descentralizado é a chave para o sucesso do consenso "descentralizado" do Bitcoin. Um esquema geral do consenso Nakamoto é apresentado no Esquema3.

Esquema 3 Consenso Nakamoto *(Esquema do Algoritmo)*

1. Request: Clientes enviam transações para todos os nós da rede;
2. Collect: Cada nó p_i da rede, ao receber as transações, as adicionam a um bloco b_i;
3. Election: Em cada rodada k do consenso, um *oráculo randômico* escolhe um nó líder p_l para propagar o seu bloco b_l aos demais;

4. Validate: Cada nó p_i aceita o bloco b_l se ele é *válido* e se as transações contidas em b_l são *válidas*. Ver procedimentos de validação nas Seções 5.2.3.1 e 5.2.3.2.
5. Update: O nó p_i ao aceitar o bloco b_l irá adicioná-lo ao final do livro-razão e finaliza a rodada k. A posteriori, irá agregar o hash $H(b_l)$ ao próximo bloco a ser criado, mantendo assim a estrutura de corrente criptográfica.

O coração do consenso Nakamoto é o *oráculo randômico*, que promove a eleição descentralizada, através da mineração. Tal como em protocolos BFT tradicionais, ele faz uso de um oráculo para contornar o resultado de impossibilidade FLP. Tal como no Paxos, o oráculo elege um líder para coordenar o processo. Mas, diferentemente do Paxos, que propõe um protocolo determinístico, ele apresenta um protocolo probabilístico, cujas garantias de terminação e, sobretudo consistência, estarão comprometidas, e só serão obtidas ao longo do tempo. Isso ocorre porque o processo de eleição randômica poderá derivar diversos líderes que poderão propor blocos a serem subsequentemente validados e aceitos para a blockchain. Nesse sentido, a corrente pode se bifurcar, ocasionando os chamados *forks*. A forma de implementação do oráculo é um dos elementos inovadores trazidos pelo Nakamoto e será apresentada em detalhes na Seção 5.4.1.

Um outro aspecto inovador do Bitcoin é o *mecanismo de incentivos*, nunca proposto dentro da BFT tradicional. Ele permite com que o sistema contorne duas grandes dificuldades de projeto: o consenso com participantes anônimos e a tolerância a falhas byzantinas nesse contexto de anonimidade. Graças a esse mecanismo, os nós maliciosos tendem a seguir a ordem instituída pelos nós honestos, pois os ganhos (financeiros) em agir honestamente, tendem a compensar atuações maliciosas. O incentivo também torna-se viável porque a geração da moeda bitcoin integra-se ao processo de eleição randômica. O nó que vence o desafio recebe bitcoins em troca do trabalho. Assim, consenso (e sua eleição) e geração da moeda (como incentivo) integram-se numa simbiose salutar para viabilizar o acordo e, adicionalmente, a oferta de serviços de acordo sob demanda na Internet. A Seção 5.4.3 discorre sobre o mecanismo de incentivos.

Finalmente, diferentemente da BFT tradicional, os nós da rede aberta (e do Bitcoin), não precisam se identificar, de maneira a serem autenticados. Mas, é preciso que a cada nó seja atribuído um *endereço*; é através dele que o nó é reconhecido nas transações do ativo, e todo o processo de validação pode ser realizado. Para tanto, o Bitcoin adota uma *gerência de identidades descentralizada*, que possibilita a *pseudo anonimidade* da rede. Esse é mais um aspecto inovador que será explorado na Seção 5.4.2.

5.3.1.2. Prova X (PoX)

Apesar das diversas inovações, o PoW do Bitcoin apresenta desvantagens evidentes. A primeira diz respeito à latência de decisão do bloco, que é relativamente alta, e em consequência a vazão (*throughput*) (quantidade de transações validadas no tempo), que é relativamente baixa. Uma outra, bem mais crítica, seria a sustentabilidade do processo de mineração, que exige alto poder computacional, gasto desmesurado de energia, sem retorno proporcional. Há ainda o risco de controle da rede por pool de mineradores que tenderiam a centralizar a decisão e realizar ataques diversos (*selfish mining attacks*).

Diversos trabalhos vêm sendo propostos no intuito de mitigar tais aspectos e favorecer decisões do consenso com base em outros critérios [Bonneau et al. 2015, Bano et al. 2017]. Eles são chamados *Prova X (PoX)*, pois propõem a substituição do desperdício computacional por trabalho válido, derivado de recursos alternativos e acessíveis. Eles advogam a adoção de uma "mineração virtual".

Prova de Posse – Proof of Stake **(PoS)**. É uma das proposta promissoras de PoX. O princípio: no lugar de gastar tempo e recursos computacionais para sair vitorioso da eleição do consenso, estabelece-se critérios sustentáveis para a eleição, com base na posse (ou interesse) que cada nó tem na rede, mais especificamente, com base na quantidade de recursos em criptomodedas que cada nó possui. Nesse caso, os votos dos participantes para a eleição é ponderado pelo investimento já realizado na rede; o peso pode obedecer a vários critérios. Nota-se que a PoS mantém de certa forma a lógica da PoW, pois a posse de criptomoedas na rede é diretamente proporcional ao poder computacional na rede.

Um dos desafios da PoS é o acompanhamento da evolução do status de cada interessado (*stakeholder*) na rede. A Peercoin foi uma das primeiras moedas a propor a PoS e, atualmente, o Ethereum projeta migrar de PoW para PoS. Alguns protocolos são: Ouroboros [Kiayias et al. 2016], Snow-White [Bentov et al. 2016]. Variações da PoS são a *Prova de Idade (Proof-of-coin-age)*, quando a prova da posse é pelo tempo em que as moedas foram alocadas à rede; *Prova de depósito (Proof-of-deposit)*, quando os mineradores alocam uma qtde específica da moeda para a rede (sem possibilidade de gasto).

Prova de Capacidade – Proof of Capacity (PoC). Nesse caso, os votos dos participantes para a eleição são ponderados pela alocação de uma quantidade não negligenciável de espaço em disco. Um exemplo é o Permacoin [Miller et al. 2014] que usa a PoC para implementação de um storage distribuído robusto.

Prova de Tempo Decorrido – Proof of Elapsed Time (PoeT). O projeto Hyperledger Sawtooth propõe uma eleição à base de temporização a partir do *Trusted Execution Environment (TEE)* presente na arquitetura do Intel SGX. No caso, os participantes solicitam uma fatia de tempo para o seu enclave de confiança. O chip com menor tempo de espera recebido do enclave é eleito o líder. De certa maneira, a PoeT adota uma estratégia próxima a de consensos para ambientes síncronos, com uso de um modelo híbrido, à base de *wormholes*: um componente de confiança em HW.

5.3.2. Consenso para Blockchain Federada/Privada

Como a rede é controlada, com n participantes bem identificados pela federação e uma composição controlada, protocolos clássicos BFT e consenso byzantino determinístico podem ser adaptadas para a blockchain. Adicionalmente, não há necessidade de uso de incentivos para obtenção do acordo, dado que a federação de interessados (*stakeholders*) pode estabelecer o seu próprio modelo financeiro de retribuição. Os incentivos podem ser usados para outros fins, mas, diferentemente do consenso à base de provas, não são imprescindíveis ao consenso.

A principal vantagem de uso do consenso BFT para a blockchain é a latência de decisão (quase em tempo real, comparativamente ao consenso à base de provas) e a alta vazão de transações. Há algumas dificuldades inerentes aos protocolos byzantinos:

(i) eleição de um líder correto (não Byzantino) para guiar o acordo e mudança de visão do grupo, na ocorrência de falhas e sobretudo *churn* (entradas e saídas aleatórias de nós na rede, de forma intensa); (ii) adoção de requisitos de segurança custosos, como um cripto-sistema de emissão de chaves públicas para garantir autenticação (PKI); (iii) maior vulnerabilidade a ataques Sybil. Em [Cachin and Vukoli'c 2017, Bano et al. 2017] diversos desses protocolos são apresentados, com suas principais inovações e desafios. Nesta seção, destacamos os principais.

5.3.2.1. PBFT – Practical Byzantine Fault Tolerance

O clássico PBFT (*Practical BFT*)[Castro and Liskov 1999, Castro and Liskov 2002] é um modelo de referência para muitos dos protocolos BFT que vem sendo adaptados para a blockchain. Segundo as análises, o seu desempenho para implementação de uma máquina de estados replicada byzantina é similar ao de uma máquina em que réplicas falham por parada. Seu principal diferencial foi propor uma solução eficiente com artifícios de segurança mínimos, como a eliminação de assinaturas de chave pública, na autenticação de mensagens, e substituição por MAC (*message authentication code*).

O BFT-Smart [Bessani et al. 2014] é um outro projeto promissor para implementação da RME byzantina, adotando um consenso similar ao PBFT. Vem sendo considerado uma das implementações mais robustas e eficientes de BFT, tanto para redes pequenas como em larga escala. Além disso, assim como o PBFT, o BFT-Smart [Sousa et al. 2018] também foi recentemente incorporado ao projeto Hyperledger.

O consenso PBFT também é realizado em rodadas coordenadas por um líder. Os líderes se sucedem em sequências de visões. Dentro de cada visão, os nós monitoram o comportamento do líder, que pode ser honesto ou malicioso; assim, caso os nós suspeitem da falha do líder, seja por parada (comprometendo a *liveness* do protocolo), seja por comportamento Byzantino (comprometendo a *safety*), eles promovem uma mudança na visão do grupo (*view change*). Além de eleger novo líder, o protocolo de mudança de visão deve garantir acordo sobre mensagens entregues durante a visão anterior, ou seja, deve assegurar a *comunicação em sincronia com as visões* [Charron-Bost et al. 2010]. O PBFT adota como premissa o modelo *parcialmente síncrono*[Dwork et al. 1988], logo depende da satisfação de requisitos temporais para garantir a sua terminação. Ele apresenta complexidade de (n^2) mensagens para n processos (réplicas) e atende às propriedades do consenso byzantino, desde que $n > 3f$, onde f é o número máximo de processos maliciosos.

O Esquema 4 apresenta um esboço do funcionamento de uma rodada de protocolo à base do PBFT (PBFT-like) para o caso normal, quando o líder não é suspeito de falhar.

Esquema 4 PBFT-like *(Esquema do Algoritmo para o Caso Normal)*

1. Request. Clientes enviam requisições de transações para processo líder ou para demais processos que irão repassar ao líder;
2. Pre-Prepare. O processo líder p_l monta um bloco $b[k]$ com transações válidas recebidas. O bloco $b[k]$ irá apontar para o hash do último bloco $b[k-1]$ agregado

à corrente de blocos. Em seguida, envia uma mensagem PRE-PREPARE para todos os demais processos propondo o acordo sobre $b[k]$;

3. Validate. Um processo p_i aceita receber PRE-PREPARE se a mensagem foi autenticada, a proposta do líder p_l é válida e o bloco $b[k]$ nunca foi decidido. Ou seja, se as transações têm assinaturas digitais válidas, não incorporam duplo-gasto e não foram confirmadas anteriormente (Ver validação nas Seções 5.2.3.1 e 5.2.3.2);
4. Prepare. Quando um processo p_i aceita o PRE-PREPARE, ele envia uma mensagem PREPARE contendo (p_l, $b[k]$) para todos os demais processos;
5. Execute. Quando um processo p_i recebe $\lceil (n+f)/2 \rceil$ mensagens PREPARE contendo (p_l, $b[k]$), então ele envia uma mensagem COMMIT contendo (p_l, $b[k]$) para todos os demais processos;
6. Commit. Quando um processo p_i recebe $\lceil (n+f)/2 \rceil$ mensagens COMMIT contendo (p_l, $b[k]$), então ele confirma o consenso sobre (p_l, $b[k]$), adiciona $b[k]$ ao final do livro-razão, e finaliza a rodada do consenso.

5.3.2.2. Outros Consensos BFT

Algumas propostas de consenso byzantino para a blockchain federada têm sido feitas com mecanismos e abstrações distintas do PBFT. Podemos classificá-las em dois grandes grupos: (1) aquelas apresentadas como *white papers*, sendo a base de diversas criptomoedas. Em sua grande maioria, como não fornecem os elementos completos para análise pública e verificação formal dos protocolos, não há como garantir que de fato implementam as propriedades do consenso byzantino e têm o desempenho anunciado; (2) propostas recentes apresentadas pela comunidade BFT e de segurança em forma de artigos científicos.

No grupo (1) encontramos protocolos bem inovadores, como o recente ***Swirlds Hashgraph*** [Baird 2016], que propõe um consenso BFT probabilístico, para modelo assíncrono com $f < n/3$ participantes, cuja blockchain é armazenada num *DAG (Directed Acyclic Graph)*. Diferentemente da PoW do Bitcoin, os autores argumentam que resolvem o BFT com garantias de consistência forte, atingindo decisões com probabilidade 1 e resistência a ataques Sybil. A estrutura do Hashgraph admite bifurcações como mecanismo natural do seu crescimento. Vários blocos podem ser minerados e a latência de decisões é em tempo quase real. Diferentemente dos demais, ele não se fundamenta em eleição de líder e não utiliza o mecanismo de troca de mensagens com votos. De fato, não precisam encaminhar votos na rede porque propõem um protocolo de votos virtual. Em resumo, o *Swirlds Hashgraph* apresenta muitas inovações, em diversos níveis, mas, é um protocolo proprietário e ainda não há implementações e análises independentes sobre seu funcionamento.

No grupo (2), alguns protocolos adotam o **Raft** [Ongaro and Ousterhout 2014], uma variação do clássico Paxos [Lamport 1998]. O Raft possui algumas implementações abertas, é adotado no projeto Quorum, do JPMorgan Chase, para desenvolvimento de uma blockchain à base do Ethereum, além dos projetos Juno e Tangaroa [Copeland and Zhong 2014]. [Crain et al. 2017] apresentam um consenso BFT diferenciado, que não se baseia em eleição de líder, nem em assinaturas, nem em randomização. Ele propõe uma

redução do consenso bizantino de valor múltiplo ao consenso bizantino binário, e com base em instâncias simultâneas de consenso binário converge a uma decisão, num modelo parcialmente síncrono.

5.4. Bitcoin e Mineração

A criptomoeda Bitcoin (www.bitcoin.org) é a prova de conceito seminal para o sucesso da blockchain. Nesta seção, serão explorados alguns conceitos técnicos associados ao protocolo original [Nakamoto 2008], com a apresentação dos elementos inovadores associados ao consenso PoW, descrito na Seção 5.3.1.1. Também serão abordadas as redes de mineradores e suas associações, além de possíveis oportunidades de ataques.

5.4.1. Mineração e Prova de Trabalho

No jargão do Bitcoin, o processo realizado pelos nós para vencer a eleição randômica do consenso é designado *mineração*. Os mineradores são computadores com hardware e software dedicados a resolver um desafio computacional (*hash puzzle*) que envolve essencialmente o uso de força bruta.

A mineração consiste em aplicar a função *hash H* sobre o *header do bloco* sucessivas vezes, pela variação do valor do *nonce*, até que o resultado do *hash* seja menor do que a dificuldade alvo imposta (*target*). No geral, o *hash* será aplicado aos campos descritos na Seção 5.2.3.2, a saber: o *nonce*, o apontador para o bloco anterior (*prev*), o hash da raiz da árvore Merkel (*mrkl_root*), além do *timestamp* e dificuldade alvo (*target*). Assim, espera-se determinar um *nonce* que satisfaça:

$$H(nonce \,||\, prev \,||\, mrkl\_root \,||\, timestamp \,||target) \;<\; target$$

Como a função *hash* é resistente à colisão (Definição 1), a determinação do *nonce* que satisfaça a inequação não será nada trivial, necessitando um algoritmo força-bruta para que diversos valores sejam testados, um a um, até que se encontre o *nonce* adequado. A mineração consiste em aplicar este processo de tentativa e erro até que se ache um acerto. O nós minerados são aqueles que têm suficiente poder computacional para realizar esse jogo.

Dificuldade A dificuldade para resolver o *puzzle* aumenta com a diminuição do alvo. A intuição é a seguinte: Se a função H tem resultado de n bits, então ela pode obter qualquer valor dentre os 2^n possíveis. Resolver o *puzzle* consiste em encontrar um valor de saída que seja $< target$ (esse alvo deve ser relativamente menor do que o espaço de 2^n soluções possíveis). Mas, se $target$ é um valor alto, temos muitas possibilidades de escolha e o desafio é trivial; por outro lado, se tem um valor muito baixo, o *puzzle* é extremamente difícil. Assim, quanto menor o espaço de busca determinado pelo *target*, maior será a dificuldade para se achar o *hash* desejado, e a única possibilidade de acerto é variar o *nounce* um a um até que o hash atinja o valor alvo.

Em consequência, o tempo de mineração depende do alvo adotado. Pelas regras instituídas, a dificuldade de mineração muda a cada 2016 blocos. Como a rede está programada para minerar um bloco a cada 10 minutos, isso corresponde a cerca de 2 semanas

(2016 $*$ 10 minutos). O alvo é ajustado com base na eficiência dos mineradores em encontrar o bloco, considerando-se o período passado de mineração para os 2016 blocos anteriores, de acordo com a fórmula:

*next_target = (previous_target * 2016 * 10) / (time to mine the last 2016 blocks)*

Corrente de Blocos Mais Longa Os mineradores que conseguem efetuar a prova de trabalho, imposta pelo desafio, vencem a competição de mineração e são autorizados a publicar um bloco de transações pendentes na blockchain. Ao mesmo tempo, propagam o seu bloco vitorioso para os demais nós da rede P2P.

Como o processo é randômico, há a possibilidade de haver mais de um vencedor para a mesma rodada de eleição, além disso, a rede é assíncrona, logo, diversos blocos de mesma altura (relativos à mesma rodada de consenso) podem vir a serem adicionados à blockchain pelos nós da rede P2P, ocasionando uma "bifurcação" na corrente, o chamado *fork*. A Figura 5.6 ilustra esse processo, quando um bloco pai passa a ter mais de um filho. De fato, devido à bifurcação, não haverá um estado global único compartilhado, e diversas correntes (ou diversas visões da blockchain) passam a coexistir ao mesmo tempo.

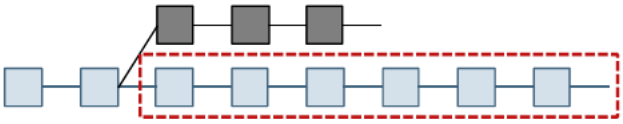

Figure 5.6. Bifurcação da Corrente de Blocos

Para resolver esta inconsistência eventual no estado da blockchain, os nós devem sempre estender a corrente mais longa (a que acumula maior altura) ou, em outras palavras, a que representa a maior prova de trabalho acumulada, ou o maior poder computacional empregado no sistema. Se todos nós adotam essa mesma estratégia, ao cabo de um tempo, a blockchain irá convergir para um estado global consistente. Assim, quanto mais antiga é uma decisão sobre um bloco na corrente, maior é a probabilidade de que esta decisão não seja demovida por outro líder minerador. *O acordo do consenso passa a ser contínuo no tempo*.

O intervalo de 10 minutos definido para mineração dos blocos é uma solução de compromisso entre o "tempo de confirmação das transações" x "a probabilidade de bifurcações". Assim, ao diminuir o intervalo, aumenta-se a vazão de transações confirmadas, mas, também aumenta-se a frequência de bifurcações. Inversamente, se o intervalo for maior, reduz-se a vazão e em consequência a quantidade de bifurcações. Na prática, estima-se que são necessários cerca de 6 rodadas de consenso (1 hora) para que a decisão tomada seja confiável.

5.4.2. Endereços e Pseudo Anonimidade

O Bitcoin adota uma *gerência de identidades descentralizada* para atribuição de *endereços* aos nós da rede. Assim, no lugar de existir uma única autoridade centralizadora,

responsável por emitir novas identidades (passíveis de autenticação), cada nó, independentemente, gera a sua própria identidade (no caso, endereço) e a anuncia para os demais nós da rede, quando pertinente. O endereço do Bicoin será o hash H da chave pública pk, i.e., $H(pk)$, gerada pelo nó para participar de transações. Eles são criados livremente pelos nós, de acordo com a sua conveniência.

De certa maneira, a gerência de identidades descentralizada apresenta um mecanismo auto-imune aos ataques Sybil, já que, por projeto, a rede Bitcoin aceita diversas identidades (endereços) para um mesmo nó; portanto, se protege de ataques de personificação. Por outro lado, tal descentralização possibilita ao Bitcoin o oferecimento de uma rede cujas transações ocorrem de maneira pseudo-anônima. Ou seja, conhece-se os endereços dos emissores e receptores dos ativos, mas, não se sabe exatamente quem são, dado que os endereços não estão ligados a um nome universal ou endereço IP. Essa pseudo anonimidade é um dos princípios basilares do Bitcoin.

5.4.3. Mecanismo de Incentivo e Geração da Criptomoeda

O incentivo no Bitcoin ocorre em dois contextos: blocos e transações. O primeiro acontece quando o minerador vence a eleição, quebrando o desafio proposto. Como recompensa, ganham uma quantidade de bitcoins recém-emitidos. O segundo ocorre na execução das transações propriamente ditas. Os nós, nesse caso, são recompensados por executarem as ações previstas nas transações.

Recompensa de Bloco. Na geração do bloco, o minerador tem autorização para incorporar uma transação especial, a *coinbase transaction*, que consiste na geração de novas criptomoedas (no caso, o bitcoin), segundo uma regra pre-estabelecida pela rede. Essa transação especial incorpora o endereço para o qual as moedas deverão ser destinadas. Assim, seguindo a lógica da recompensa, o endereço de destino será determinado pelo minerador.

Geração da Moeda. No artigo original, Nakamoto propõe um engenhoso mecanismo de geração da moeda, que obedece a uma série geométrica decrescente e, que, por coincidência, é próxima à curva de distribuição do ouro (de onde vem a expressão do bitcoin ser o "ouro digital"). Ele estabeleceu que, a cada 4 anos, a quantidade α de moedas gerada por bloco deveria ser dividida pela metade. Assim, nos período de 2009-2013, a recompensa por bloco minerado foi de $\alpha = 50$ bitcoins; de 2013-2017, $\alpha = 25$ bitcoins e, atualmente $\alpha = 12,5$ bitcoins. Ao mesmo tempo, se somarmos os valores da série geométrica, que é finita, teremos um total de 21 milhões de bitcoins, gerados ao longo dos anos. Assim, o artigo propõe criar uma moeda, com geração controlada, cuja emissão irá cessar no ano de 2140, de acordo com a série. Ou seja, o sistema está programado para emitir apenas 21 milhões de bitcoins! Este é, inclusive, um dos motivos pelos quais há uma forte especulação em torno da moeda.

Taxa de Transação. A taxa de transação pode ser estabelecida de forma a incentivar os mineradores a agregarem aquelas transações no seu bloco; ou, de modo geral, de forma a oferecer aos usuários demandantes uma qualidade de serviço desejada. Ela pode ser determinada, por ex., como uma taxa aplicada aos valores transacionados (dentro do bloco). De modo geral, ela equivale à subtração da soma dos valores de entrada menos os de saída. O mecanismo de compensação para as transações é mais sofisticado no projeto

no Ethereum, quando os demandantes costumam estimar valores de gás como forma de retribuição pela execução dos métodos dos contratos inteligentes. A recompensa é ainda uma área de pesquisa muito pouco estudada, exigindo elementos da teoria dos jogos para a sua apreensão e incorporação num modelo de negócios promissor.

O consenso PoW, juntamente com o mecanismo de recompensa, são a chave para tolerar ataques de agentes maliciosos na rede. Não é possível para uma minoria de mineradores manipular as transações, pois a rede como um todo não aceitaria transferência de ativos (ou pagamentos) que não foram autorizados pelo proprietário dos ativos (ou bitcoins). Além disso, os nós sempre estendem a corrente mais longa. Para um ataque cibernético, seria necessário que uma maioria (51%) de nós mineradores entrasse em conluio. Isso envolveria um custo computacional altíssimo [Nakamoto 2008].

5.4.4. Ataques e o Problema do Duplo Gasto

Nas moedas tradicionais, o problema do gasto duplo é inexistente, uma vez que são compostas de matéria e possuem propriedades físicas distintas. O bitcoin, no entanto, não possui tais características, sendo formado essencialmente por bits, cujo conceito é abstrato. Sendo assim, em princípio, seria simples duplicá-los e, por exemplo, pagar duas vezes com a mesma moeda em duas transações diferentes. Como não há um banco ou lastro confiável que possa impedir que os usuários gastem o mesmo dinheiro duas vezes, torna-se necessário criar uma solução adequada [Karame et al. 2012, Conti et al. 2017].

Um ataque abordando a possibilidade de duplo gasto explora o seguinte cenário: Um cliente desonesto `Cd` cria uma transação `T` entre ele e uma loja (L), em um tempo `t` usando um conjunto de bitcoins `Bc` para comprar um produto de `L`. `Cd` envia pela rede a mensagem `T`. Quase que instantaneamente, em `t'`, `Cd` cria e envia outra transação `T'` usando o mesmo conjunto de bitcoins `Bc` para outra loja `L'` ou outra carteira que esteja sob seu controle. Se a loja `L` entregar o produto, segundo este cenário, o duplo gasto pode se concretizar.

Na blockchain, o gasto duplo é evitado com a validação do bloco e das transações sempre que há uma decisão. Quando uma transação é incluída em um bloco, prova-se que nenhuma transação anterior gastou os mesmos bitcoins e que as transações futuras também não o farão (Seção 5.3.1.1). Assim, nas transações `T` e `T'`, o minerador estará apto a identificar que ambas as transações tentam usar a mesma entrada `Bc` durante a propagação da transação, fazendo com que uma seja rejeitada e outra aceita.

Entretanto, apesar do uso do PoW, do rigoroso ordenamento das transações, ainda assim, é possível ataque de duplo gasto, com base na exploração da bifurcação da corrente da blockchain, conforme o seguinte cenário: (i) parte da rede de mineradores aceita a transação `T` e envia a confirmação para a loja, que então entrega o produto ao cliente desonesto `Cd`; (ii) ao mesmo tempo, outra parte da rede de mineradores aceita a transação `T'` e leva a blockchain a criar um *fork* na rede; (iii) a loja recebe a confirmação da transação `T'` depois de aceitar a transação `T`, então perde o produto, e (iv) a maioria dos mineradores decide pela transação `T'` como válida. Existem variantes deste tipo de ataque: (a) *Finney attack*; (b) *Vector 76*; (c) *Ballance*; (d) *Goldfinger* ou *>50%*. Cada uma delas explora vulnerabilidades que permitem realizar o duplo gasto [Conti et al. 2017].

5.4.5. Associação de Mineradores

No contexto da mineração de criptomoedas, a associação de mineradores pode ser definida como um agrupamento de recursos computacionais dos mineradores, que compartilham seu poder de processamento em uma rede e dividem a recompensa igualmente, de acordo com a quantidade de trabalho contribuída por cada um para a probabilidade de encontrar um bloco. Em uma associação de mineradores pode haver centenas ou milhares de associados [Eyal 2015].

Este modelo de mineração foi motivado devido ao aumento de competição para geração de um bloco e obtenção de receitas sazonais. Como solução, os mineradores reuniram seus recursos de forma a gerar blocos mais rapidamente e, portanto, receber uma parte da recompensa do bloco ao invés de aleatoriamente em intervalos cada vez maiores [Eyal and Sirer 2014]. Embora ingressar em um pool não altere a receita esperada de um minerador, ele diminui a variação e torna as receitas mensais mais previsíveis. Foram propostos diferentes métodos para repartir as receitas, como por exemplo *Pay-per-Share*, *Proportional*, *Bitcoin Pooled Mining*, *Pay-per-last-N-chares*, *Geometric* e *Double Geometric* [Rosenfeld 2011].

Atualmente, as associações de mineradores são controladas por gerenciadores que encaminham unidades de trabalho não resolvidas para os membros. Os mineradores geram provas de trabalho parciais (PPoWs) e provas de trabalho completas (FPoWs), e as submetem ao gerente como ações. Uma vez que um mineiro descobre um novo bloco, ele é submetido ao gerente junto com o FPoW. O gerente transmite o bloco na rede Bitcoin para receber a recompensa de mineração e distribui a recompensa aos mineradores participantes com base na fração de ações contribuída quando comparada com os outros associados. Assim, os participantes são recompensados com base nos PPoWs, que não têm absolutamente nenhum valor no sistema Bitcoin [Conti et al. 2017].

5.4.5.1. Segurança e Ataques

Nos últimos anos, a exploração de vulnerabilidades e ataques às associações de mineradores aumentou, sendo possível classificá-los em ataques internos e ataques externos. Os ataques internos são aqueles onde um grupo de mineradores desonestos agem de forma maliciosa com a associação, objetivando a coleta de mais recompensas do que lhe seria devido ou para interromper a funcionalidade da associação de forma a distanciar-se das tentativas de mineração bem sucedida. Nas investidas externas, os mineradores podem usar do seu poder computacional para executar fraudes como o duplo gasto.

Um tipo bastante popular de ataque é o *Block Discarding* (BD). Mineradores desonestos realizam a ocultação de informação, retendo um bloco minerado e realizando a sua revelação de maneira muito seletiva, visando: (a) obter uma recompensa injusta maior do que a sua quota de poder de computação gasto, e (b) confundir outros mineradores e levá-los a desperdiçar seus recursos em uma direção errada [Courtois and Bahack 2014].

O *Block Withholding* (BW) é um tipo ataque similar ao BD, onde um membro da associação nunca publica um bloco minado para sabotar a receita a associação. No entanto, ele submete ações consistindo em PPoWs, mas não em FPoWs. Dois cenários

de *Block Withholding* são destacados: a) “sabotagem” e b)“espera”. Em (a), o adversário não ganha nenhum bitcoin, apenas faz com que outros membros do pool percam. Em (b) o adversário realiza um bloqueio complexo que esconde um ataque semelhante ao *Block Discarding* [Rosenfeld 2011, Bag et al. 2017].

O *Fork after withholding* (FAW) [Kwon et al. 2017], proposto recentemente, mostra que a recompensa dos atacantes do BWH é o limite inferior dos atacantes do FAW. A recompensa extra por um ataque FAW ao operar em múltiplas associações de mineradores é cerca de 56% maior do que o ataque de BWH e chega a até quatro vezes ocorrências por associações do que o ataque do BWH.

O *Pool Hopping* (PH) usa as informações sobre o número de ações enviadas pela associação. Neste ataque, o minerador desonesto realiza uma análise contínua do número de ações submetidas por colegas mineradores ao administrador do grupo, a fim de descobrir um novo bloco [Rosenfeld 2011]. A ideia é que, se um grande número de ações já tiver sido submetido e nenhum novo bloco tiver sido encontrado até agora, o adversário receberá uma parcela muito pequena da recompensa, porque ela será distribuída com base nas ações enviadas. Portanto, em algum momento, o minerador egoísta pode decidir por mudar para outro grupo ou minerar de forma independente.

Por fim, o *Bribery* [Bonneau 2016] pode obter a maioria dos recursos de computação por um curto período de tempo via suborno. Discute-se três maneiras de introduzir suborno na rede: (i) O minerador desonesto paga diretamente ao proprietário pelos recursos de computação e estes proprietários mineram os blocos atribuídos ao minerador desonesto, (ii) O minerador desonesto forma uma associação pagando maior retorno pela mineração, e (iii) O atacante tenta subornar através do Bitcoin criando um *fork* contendo dinheiro de suborno disponível a qualquer minerador que adote a *fork* desonesta.

5.5. Hyperledger

O Hyperledger (www.hyperledger.org) é uma plataforma de código aberto, mantida pela Linux Foundation e diversas corporações, dentre elas a IBM, que propõe o Fabric [Cachin et al. 2016]. O Fabric é uma implementação de blockchain privada voltada para aplicações empresariais com objetivo de atender aos múltiplos e variados requisitos dos aplicativos de negócio, alavancar os estudos da tecnologia blockchain e permitir a escalabilidade de serviços nele baseados [Fabric 2017] [Nguyen et al. 2018]. Sua arquitetura modular oferece altos graus de confiabilidade, resiliência, flexibilidade e escalabilidade [Nguyen et al. 2017], possibilitando a criação de vários módulos conectáveis e viabilizando a implantação de serviços diferenciados, como consenso, armazenamento e filiação de membros [Fabric 2017] [Nguyen et al. 2018].

No Fabric cada participante registra sua participação utilizando o serviço de filiação a rede (*Membership Service Provider* - MSP) para obter acesso ao sistema [Nguyen et al. 2018], possibilitando, assim, o uso de consensos bizantinos e desobrigando o uso de consensos baseados em prova de trabalho (PoW) para validar as transações e proteger a rede, e provendo transações seguras, privadas e confidenciais [Fabric 2017].

O Fabric executa contratos inteligentes (*chaincodes*), disponibiliza o consenso PBFT [Castro and Liskov 1999], permite a um grupo de participantes criar um livro razão

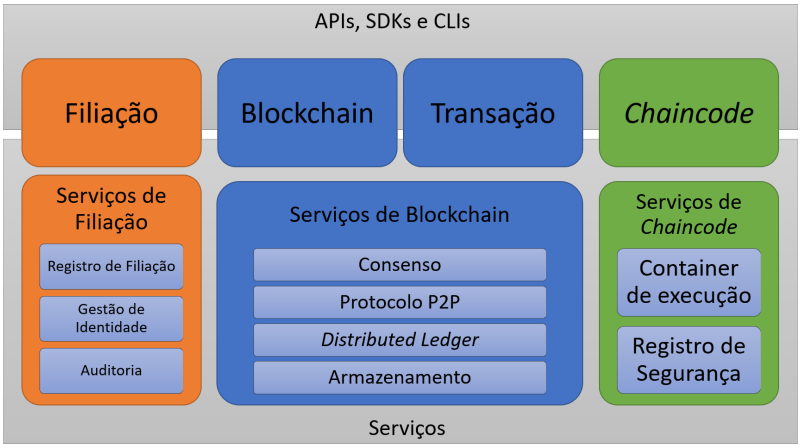

Figure 5.7. Arquitetura de Referência Hyperledger

separado das transações no qual apenas os participantes deste grupo possuem cópias do *ledger* através da criação de canais, permite persistência utilizando chave-valor facilitando as consultas através de um banco de dados não relacional, suporta eventos, abstrai da complexidade dos protocolos de comunicação através do *standard developed Kit* (SDK), promove segurança através de autoridades certificadoras e promove suporte a REST APIs e CLIs.

O Fabric diferencia os nós pela sua responsabilidade e os categoriza em: nós de validação e nós de não-validação. Os nós de validação são responsáveis pela execução de consenso, validação de transações e manutenção do ledger. Os nós de não-validação funcionam como um *proxy* para conectar clientes para os nós de validação (emitir transações) e podem validar transações. Um nó de não-validação não executa transações [Cachin et al. 2016]. A responsabilidade de cada nó é atribuída durante o processo de filiação à rede através do serviço de filiação. O serviço de filiação será apresentado posteriormente.

5.5.1. Arquitetura

O Fabric possui uma arquitetura de referência organizada em 3 categorias lógicas: Serviços de filiação ou *Membership*, Serviços de Blockchain e Serviços de *Chaincode*. A Figura 5.7 apresenta a organização destas categoria no Fabric.

5.5.1.1. Serviços de Filiação ou Membership

Os serviços de filiação (MSP- *Membership Service Provider*) consistem em uma infraestrutura composta por várias entidades que fornecem abstrações para operações de associação: registro e gerenciamento de identidade, privacidade, confidencialidade e auditoria na rede [Nguyen et al. 2018]. Estes serviços validam a identidade do usuário, registram-no no sistema e fornecem todas as credenciais necessárias para um participante na rede, seja ele um nó ativo ou uma aplicação capaz de invocar um *chaincode*.

O framework base do serviço de filiação é uma associação da criptografia de chave pública e o consenso. Através do uso da infraestrutura de chave pública, ICP, as mensagens podem ser trocadas de maneira segura e a identidade das partes podem ser ates-

tadas. Com o uso das entidades que compõem o serviço de gerenciamento de membros é possível identificar um usuário individual e permitir a execução de operações a partir das permissões atribuídas ao mesmo [Nguyen et al. 2018]. Com o uso deste mecanismo o Fabric classifica os nós do sistema, identifica e credencia usuários e permite a instalação e invocação de contratos inteligentes.

A abordagem adotada pelo Fabric para conciliar o gerenciamento de identidade com a privacidade de usuário implementa três artifícios: adicionar certificados a transação para implementar uma blockchain permissionada; utilizar um sistema com dois níveis certificados: certificados de inscrição estáticas (*static enrollment certificates* - ECerts) juntamente com certificados de transação (*transaction certificates* - TCerts); e oferecer mecanismos para ocultar o conteúdo das transações para membros não autorizados do sistema [Nguyen et al. 2018]. Os ECerts são certificados de longo prazo. Eles são emitidos para todos os papéis da rede: usuários, nós validadores e não-validadores [Nguyen et al. 2018]. Os TCerts são certificados de curto prazo para cada transação do usuário. Eles são emitidos pela Autoridade de Certificação de Transação (*Transaction Certificate Authority* - TCA) após a autenticação das credenciais de inscrição fornecidas pelo usuário. São emitidas exclusivamente para usuários e autorizam, de forma segura, uma transação. Podem ser configuradas para não revelar ou revelar seletivamente as identidades envolvidas na transação [Nguyen et al. 2018]. Desta forma, os TCerts possibilitam a utilização de pseudônimos para representar usuários inscritos na rede.

5.5.1.2. Serviços de Blockchain

O *ledger* no Fabric, conjunto de blockchain e logs de transações, encontra-se replicado na rede, isto é, cada participante possui sua própria cópia do *ledger*. Neste contexto, os serviços de blockchain gerenciam o estado do *ledger* distribuído entre os nós através de mensagens bidirecionais utilizando comunicação P2P sobre o http [Nguyen et al. 2018].

Como regra básica, as transações que compõem o blockchain devem ser escritas no *ledger* na ordem em que ocorrem, mesmo que possam estar entre diferentes conjuntos de participantes dentro da rede. Para que isso aconteça, a ordem das transações devem ser estabelecida e um método para rejeitar transações ruins que tenham sido inseridas no *ledger* por engano (ou maliciosamente) deve ser posto em prática [Nguyen et al. 2017].

O Fabric, por ser uma implementação de blockchain privada, permite a implementação de protocolos de consenso BFT, como por exemplo o PBFT [Castro and Liskov 1999], no qual fornece um mecanismo para que as réplicas do *ledger* se comuniquem entre si para manter cópias consistente, mesmo em caso de corrupção. Tal implementação é alavancada através dos serviços de blockchain, que efetuam a comunicação entre os nós através de 4 tipos de mensagem: mensagem de descoberta, mensagem de transação, mensagem de sincronização e mensagem de consenso.

As mensagens de descoberta são utilizadas como ponto de partida para descobrir todos os pares da rede e iniciar, caso o número do último bloco do nó emissor for menor que a do nó receptor, a sincronização do *ledger* com a rede. Todo nó possui pelo menos um endereço e porta de um participante da rede para efetuar a descoberta.

Após a descoberta, é iniciado o protocolo de sincronização, no qual um nó sin-

croniza seu ledger com os demais nós da rede. Algumas das mensagens usadas são: SYNC_GET_BLOCKS, SYNC_STATE_GET_SNAPHOT ou SYNC_STATE_GET_DELTAS e recebe SYNC_BLOCKS, SYNC_STATE_SNAPSHOT ou SYNC_STATE_DELTAS, respectivamente. Na mensagem de sincronização o plug-in de consenso determina como a sincronização deverá ser realizada enviando sua respectiva mensagem.

As mensagens de transação manipulam o *ledger* e estão sempre associadas a um *chaincode* e um ambiente de execução (permissões e linguagem de execução). Estas mensagens podem ser de 3 tipos: deploy, invoke e query. Uma mensagem de *deploy* instala o *chaincode* especificado no blockchain, particularmente nos nós de não-validação, enquanto que *invoke* e *query* invocam e consultam um *chaincode* respectivamente [Nguyen et al. 2017]. As mensagens de consenso tratam as transações a serem incluídas no *ledger* convertendo uma mensagem CHAIN_TRANSACTION para CONSENSOS emitida para os nós de validação. O plug-in de consenso recebe esta mensagem e a processa de acordo com seu algoritmo. Na versão 1.1 do Fabric é disponibilizado o consenso PBFT através do pacote abcpbft.

O ciclo de vida de uma transação incia-se nos nós de validação que as recebem e processam nas seguintes fases: Pré-validação: valida o certificado de transação na autoridade certificadora raiz, verifica a assinatura do certificado de transação incluído na transação e verifica se a transação não é repetida; Consenso: adicionam a transação em ordem total; Pré-execução: validam a validade do certificado de transação/inscrição em relação ao período de validade atual, descriptografam o corpo da mensagem e verificam se a mesma encontra-se bem formada; Execução: o chaincode é enviado ao container e executado; Confirmação: as atualizações advindas da transação são incluídas no *ledger* e o cliente que invocou a transação recebe a confirmação da transação

5.5.1.3. Serviços de Chaincode

Os serviços de *chaincode* fornecem uma maneira segura e otimizada para a execução de contratos inteligentes. É um container para execução do *chaincode*, com políticas de permissão e conta com SDK na linguagem GO para a execução. O *chaincode* é um código de nível de aplicativo implantado como uma transação para ser distribuído na rede e gerenciado por cada nó de validação.

A comunicação entre os nós de validação e seus *chaincodes* é baseada em um fluxo bidirecional de comunicação através da *shim layer*. A *shin layer* é uma camada entre o container de execução e o *chaincode* responsável pela manipulação do protocolo de comunicação entre os nós de validação e o *chaincode*. As mensagens trocadas são do tipo *ChaincodeMessage* e possuem os campos *type, payload, uuid*. *Type* indica o tipo da mensagem, *payload* possui informações importantes a comunicação e seu conteúdo depende do *type* da mensagem e *uuid* é o identificador único da mensagem.

Interface de Programação de Aplicativos (API) e Interface de Linha de Comando (CLI) As APIs e CLIs são interfaces que permitem aos aplicativos / desenvolvedores registrar usuários, consultar *chaincode* e emitir transações. Há um conjunto de comandos para interagir com os *chaincode*.

5.6. Aplicação em Redes de Computadores

O consenso sob demanda oferecido pela tecnologia blockchain abre perspectivas para os novos paradigmas e arquiteturas de redes recentemente discutidas na literatura [Xylomenos et al. 2014, Mijumbi et al. 2016, Blenk et al. 2016]. Esta seção discute como alguns desses paradigmas e arquiteturas podem se beneficiar através de base de dados distribuídos dos blockchains[Sankar et al. 2017, Yin et al. 2018], apresenta aplicações de redes e, por fim, alguns trabalhos em andamento em cada uma das áreas citadas.

5.6.1. SDN e Controladores Distribuídos

Em redes implementadas através do paradigma das Redes Definidas por Software (do inglês, *Software-Defined Networking* – SDN) [Kreutz et al. 2015], o consenso da blockchain pode oferecer suporte à implementação de soluções com controladores distribuídos, que precisam compartilhar uma mesma visão, global e logicamente centralizada, da rede [Sharma et al. 2017a]. Ainda que em estágios iniciais, a literatura atual aponta para trabalhos que sugerem o uso de blockchains na implementação de planos de controle distribuídos e, assim, endereçar questões em aberto relacionadas à disponibilidade de controladores [Blenk et al. 2016, Mijumbi et al. 2016], sobretudo em cenários de IoT [Sharma et al. 2017b]. Um arquitetura SDN baseada no uso de blockchains pode estar menos vulnerável a ataques no plano de controle quando a base de informações sobre as regras de fluxos da rede está distribuída entre múltiplas entidades ao invés de estar associada uma autoridade central. Diversos trabalhos na literatura já apresentam soluções de tolerância a falhas em SDN através de planos de controle distribuidos [Blenk et al. 2016, Mijumbi et al. 2016, Kreutz et al. 2015], porém as mesmas ainda não exploram os benefícios de um sistema distribuído que faz o controle de transações, como na tecnologia blockchains. Numa discussão mais ampla, em [Bozic et al. 2016] os autores apresentam um conjunto de casos de uso de blockchains.

Além dos benefícios da implementação de soluções de tolerância a falhas, o uso de blockchains pode facilitar a instalação de regras de encaminhamento para um conjunto específico de comutadores (*switches*). Tais regras podem ser validadas de forma conjunta, evitando inconsistência ao longo do caminho e garantindo a lógica definida pelo controlador [Agarwal et al. 2014]. Apesar de algumas iniciativas recentes, estas são questões de pesquisa em aberto para as quais o consenso por blockchain pode trazer novas soluções.

5.6.2. Virtualização de Funções de Rede

Em anos recentes, a operação de redes de larga escala têm sido impulsionada na academia e indústria através de Virtualização de Funções de Rede (do inglês, *Network functions Virtualization* – NFV) [Mijumbi et al. 2016]. Trabalhos recentes têm sido propostos na implementação de arquiteturas de NFV, em que funções de rede (*Virtual Network Functions* – VNFs) são gerenciadas, configuradas e migradas de forma segura através de blockchains. Em [Alvarenga et al. 2018], os autores propõem uma plataforma de VNF para garantir imutabilidade e não-repúdio das funções, implantar mecanismos de auditoria a partir do histórico de configurações e prover integridade e consistência de informações armazenadas. Ainda relacionado a NFV, em [Bozic et al. 2017], os autores apresentam uma discussão sobre como blockchains pode ser utilizado para prover a orquestração de

operações de NFV em ambientes de nuvem. Neste trabalho é proposto um protocolo de autenticação que atua em um nó orquestrador de máquinas virtuais. O protocolo faz uso de blockchains para gerenciar requisições nas máquinas virtuais (criação, destruição, cópia, migração) como transações que ficam armazenadas no banco de dados compartilhado.

5.6.3. Redes Centradas na Informação

Diante das limitações impostas pelo IP, novos projetos para a chamada Internet do futuro têm sido frequentemente investigados pela acadêmica nos últimos anos. Dentre esses projetos, a arquitetura de Redes Centradas na Informação (do inglês, *Information-Centric Networking* – ICN) consiste em um novo paradigma no qual, diferentemente da arquitetura IP, apresenta um modelo de consumo/requisição centralizado no detentor do conteúdo, em que o nó não precisa conhecer a *priori* a localização e identidade do produtor de um determinado conteúdo. Em arquiteturas ICN utiliza-se o nome do próprio conteúdo, ao invés do endereço IP, para realizar uma requisição [Jacobson et al. 2009, Xylomenos et al. 2014, Zhang et al. 2014].

Em trabalhos recentes [Fotiou and Polyzos 2016a, Jin et al. 2017a] foi aplicado o conceito de ICN para prover suporte a tecnologia Blockchain. Apesar de Blockchain, como visto anteriormente, ser uma tecnologia robusta e altamente resiliente, esta ainda está sujeita às limitações impostas pela arquitetura IP atual no que se refere à roteamento, endereçamento e escalabilidade que introduzem problemas na performance da aplicação. Como consequência da ineficiência das redes IP nesse cenário é possível citar as conexões falhas e protocolos impróprios que contribuem para a geração de forks no blockchain [Jin et al. 2017b]. Outro problema apresentado por [Jin et al. 2017b] destaca que apesar da natureza distribuída do blockchain, o estabelecimento de conexões entre nós homogêneos é algo complexo. Para entender melhor esse problema segue um breve exemplo. Na descoberta de vizinhos no bitcoin, alguns endereços IP de possíveis voluntários são recomendados como opções de conexões para os usuários. Esses nós voluntários atuam como uma infraestrutura para ajudar usuários normais a se comunicarem. Entretanto, essa abordagem aumenta a probabilidade de existir supernodes e com isso fatores de insegurança como fraudes e pontos únicos de falha.

Já em [Fotiou and Polyzos 2016a] é proposto o inverso, no qual a tecnologia blockchain é utilizada para adicionar mais segurança a ICN. Essa camada adicional de segurança está presente na distribuição de conteúdos que, como apontado pelos autores, em ICN a informação é identificada pelo seu próprio nome, portanto, é natural considerar o nome como uma primitiva básica de segurança. Isso abre uma gama de possíveis superfícies de ataque uma vez que, diferente de das chaves públicas RSA, o nome está em texto claro e é humanamente legível o que o torna fácil de memorizar e distribuí-lo. Os nomes também são previsíveis tornando possível que um atacante crie um conteúdo falso, que ainda não foi concebido por seu produtor original, e o propague na rede. Por fim, os nomes podem ser hierárquicos o que expõe organizações e relações de negócios. Com base nesses problemas, os autores em [Fotiou and Polyzos 2016b] utilizaram o blockchain para permitir que qualquer "nó inscrito" possa checar a validade de um conteúdo com base em seu nome. No sistema proposto cada entidade compartilha informações públicas, a partir disso, o inscrito que conhecer essas informações pode verificar a assinatura da informação independente do provedor.

Redes Centradas na Informação provêm novas arquiteturas de comunicação da Internet em que projetistas podem se beneficiar do consenso oferecido pela tecnologia blockchains. Por exemplo, a proposta BlockNDN [Jin et al. 2017a], propõe a implementação de uma rede de dados nomeados a partir do uso de blockchains. Os autores propõem partes de uma arquitetura ICN, incluindo um novo esquema de nomeação baseada no uso de blocos e mecanismos de encaminhamento de interesses.

Além de novas arquiteturas, modelos de reputação podem ser construídos de forma a garantir apenas a entrada de produtores de conteúdos confiáveis na rede, um problema ainda com muitas questões em aberto e investigado na literatura recente [Salah and Strufe 2016]. Ao garantir a entrada apenas de nós confiáveis na rede, modelos de segurança podem ser criados para evitar ataques, tais como de negação de serviço [Dai et al. 2013, Compagno et al. 2013, Wang et al. 2014, Seixas et al. 2017]. Além disso, tais sistemas podem ajudar na manutenção de cache confiável entre os nós [Jin et al. 2017b, Tourani et al. 2017]. Ainda relacionado a segurança em ICN, o consenso por blockchains pode ser utilizado como estratégia de mitigação de ataques de negação de serviço através de técnicas de prova de trabalho nos clientes [Li and Bi 2014].

5.6.4. Segurança de Redes

No que diz respeito à segurança de redes, a aplicabilidade de blockchains começa a ser vista na implantação de modelos de confiança [Yang et al. 2017, Sharma et al. 2017a] para, dentre outras coisas, possibilitar autenticação descentralizada e anônima de dispositivos móveis em redes C-RAN [Yang et al. 2017] e a validação de anúncios de recursos de numeração (i.e., IPv4/v6, ASN) no roteamento BGP [Paillisse et al. 2017]. Nesta seção, algumas dessas aplicações serão apresentadas e discutidas.

5.6.4.1. Autenticação Anônima e Confiável Baseada em Blockchain para C-RAN

A arquitetura descentralizada e segura da tecnologia de blockchain pode beneficiar a confiabilidade na comunicação e armazenamento distribuído, agregando características chaves como persistência, anonimidade e auditoria. Tais características mostram-se promissoras, por exemplo, no contexto de autenticação de terminâveis móveis 5G em uma Rede de Acesso de Rádio em Nuvem (do inglês, *Cloud Radio Access Network* - C-RAN), cujo funcionamento originalmente baseia-se em uma autenticação centralizada no núcleo da rede móvel, gerando impactos na operação e no consumo dos recursos de rede. A partir do uso de um modelo de autenticação baseado em blockchain, é possível melhorar a a confiabilidade no acesso de clientes e com baixo custo de rede nesse ambiente.

[Yang et al. 2017] propõe uma estratégia de autenticação confiável baseada em blockchain (BTA, do inglês *Blockchain-based Trusted Autentication*) em C-RAN para 5G, incorporando um esquema de acesso anônimo e confiável baseado em blockchain. Para isso, os autores utilizam uma arquitetura integrada do controlador SDN com uma plataforma de blockchain. O papel do controlador SDN é realizar a orquestração da frequência de rádio, do espectro óptico e dos recursos de processamento das unidades de banda base (BBU). Já a blockchain consiste essencialmente de um livro-razão público, onde as identificações de acesso confirmadas são armazenadas na cadeia de blocos.

Na arquitetura proposta, o processo de autenticação e acesso anônimo de clientes

é executado de forma tripartite: pelo dispositivo cliente (*device*), pelo fabricante (*manufacturer*) e pelo operador (*operator*). O processo funciona conforme descrito a seguir. Para garantir a anonimidade do dispositivo, o fabricante primeiro cria e envia para o operador uma chave pública de proveniência comprovada. Ao conectar na rede, um dispositivo cliente requisita informações criptográficas do fabricante, utilizando técnicas de assinatura às cegas. Ao receber o material criptográfico do fabricante, o dispositivo utiliza-o para gerar sua chave secreta independente da identidade, que por sua vez é enviada ao operador para verificação de proveniência. A partir daí, o controlador SDN faz o setup do caminho entre o dispositivo e o BBU, considerando informações de utilização do rádio, dos canais ópticos e do BBU. Finalmente o dispositivo envia uma requisição de acesso para o operador, usando técnicas de desafio-resposta, e caso a validação seja bem sucedida, o cliente recebe acesso à C-RAN e é registrado na blockchain.

5.6.4.2. Validação de Recursos de Numeração de Rede (IPv4/v6)

O roteamento na Internet é baseado, principalmente, no protocolo BGP (do inglês *Border Gateway Protocol*), cujo funcionamento consiste, em linhas gerais, no anuncio de prefixos IP entre Sistemas Autônomos (AS) e em um conjunto de políticas de roteamento que influenciam na escolha dos caminhos. O BGP é comumente referenciado como a "cola que une a Internet", habilitando a comunicação entre grandes redes operadas por diferentes organizações. Ao passo que o BGP desempenha um papel fundamental na comunicação da Internet, ele incrivelmente se mantém vulnerável à uma série de ataques. Grande parte dessas vulnerabilidades derivam do excesso de confiança entre os pares e da falta de mecanismos de checagem das informações trocadas. Diversas soluções foram propostas ao longo dos anos, geralmente baseadas em padrões de Infraestrutura de Chaves Públicas (ICP), que inclusive deram origem aos mecanismos atuais de segurança: RPKI (Resource Public Key Infrastructure) e BGP-SEC (BGP Security Extension). Estes mecanismos tratam-se de um conjunto de protocolos, padrões e sistemas que melhoram a segurança dos recursos de numeração da Internet como os endereços IPv4 e IPv6, possibilitando a verificação do direito de uso. Embora promissores, esses mecanismos ainda possuem baixa adoção pelos provedores. Por outro lado, a tecnologia de blockchain pode beneficiar ou complementar alguns dos desafios desse modelo centralizado das ICPs e incorporar melhorias para a segurança no roteaento interdomínio.

[Paillisse et al. 2017] analisa como a tecnologia de blockchain pode ser utilizada para adicionar segurança ao processo de alocação, delegação e mapeamento dos recursos de numeração (i.e. endereços IP), provendo um grau de segurança similar à arquitetura tradicional baseada em ICP. Em linhas gerais, criptomoedas e endereços IP compartilham características fundamentais que motivam o uso de blockchain para proteção: endereços IP são alocados de forma não ambígua para entidades (AS's); endereços IP podem ser transferidos entre entidades; um bloco de rede não pode ser alocado para duas entidades ao mesmo tempo; os blocos podem ser divididos em sub-blocos até um certo limite. A partir dessas propriedades em comum, é possível conceber uma blockchain que permite aos seus participantes (AS's) delegarem blocos de endereços IP de forma similar às transações de criptomoeadas. Por exemplo, a IANA poderia escrever uma transação alocando endereços para os RIRs (do inglês, *Regional Internet Registries*), como o LACNIC por exemplo, que, por sua vez, poderiam lançar mão de uma outra transação alocando um sub-bloco para um

Registro Nacional de Internet (e.g. Registro.br) ou para um provedor diretamente, e assim sucessivamente. Em contrapartida, utilizar a tecnologia de blockchain nesse contexto adiciona algumas vantagens:

- Descentralização: não haveria uma entidade centralizadora controlando o blockchain, ele é compartilhado entre todos os participantes;
- Sem necessidade de certificados, ACs (Autoridades Certificadoras) e CRLs (do inglês, *Certificate Revocation List*);
- Resistência a censura: uma vez que o controle das transações depende do detentor da chave privada, a revocação ou mudança de uma alocação de IP sem a permissão do proprietário legítimo involveria o comprometimento da chave privada do proprietário. Mesmo que a chave privada da entidade hierarquicamente superior na alocação do bloco IP seja comprometida, a confiabilidade da transação atual é preservada, o que não é válido no caso de comprometimento da chave privada de uma AC ou de ACs maliciosas no cenário com ICP.
- Auditoria: alocações e delegações podem ser rastreadas na blockchain para determinar a origem do proprietário legítimo de uma transação.

Um aspecto importante na modelagem desse problema em uma blockchain é a escolha do algoritmo de consenso. Em particular, algoritmos de prova de trabalho (PoW) apresentam algumas desvantagens, principalmente pelo fato de que o incentivo para um participante legítimo da blockchain de endereços IPs não é diretamente ligado ao seu poder computacional, ou seja, um atacante poderia investir em adquirir alto poder computacional para reescrever a blockchain com dados falsos. Já com os algoritmos de prova de posse (PoS) a capacidade de alteração da blockchain permaneceria no próprio proprietário do bloco. Tipicamente o AS que é proprietário do bloco de endereços possui negócios na Internet associados àquele bloco, de tal forma que ele possui claros incentivos em manter uma operação correta e segura das suas transações.

5.7. Contratos Inteligentes na Internet das Coisas

A Internet das Coisas (IoT), ao mesmo tempo em que possibilita uma gama de aplicações interessantes (cidades inteligentes, saúde, casas inteligentes, etc.), apresenta desafios, sobretudo em nível de segurança e privacidade, dado o grande número de dados gerados, compartilhados e a exposição a que os mesmos estão sujeitos. Nesse sentido, o casamento da IoT com a blockchain é celebrado por fornecer uma nova camada computacional para compartilhamento e análise segura dos dados, com garantias de privacidade. Assim, essa camada pode ser utilizada para autenticar, autorizar, controlar e auditar os dados gerados pelos objetos inteligentes [Christidis and Devetsikiotis 2016].

5.7.1. Introdução à IoT

A Internet das Coisas (do inglês *IoT*) nasceu do avanço de áreas como microeletrônica, sistemas embarcados, comunicação e sensoriamento [Santos et al. 2006] e hoje é uma realidade perene do cotidiano. A IoT é composta por Objetos Inteligentes(OI) capazes de enviar informações sobre si mesmo, sobre o lugar em que estão inseridos ou agir em determinadas situações, trazendo para o mundo real uma dinâmica nunca antes experimentada.

Esta definição gera implicações e cria conceitos inovadores como *weareable*, casas, carros e cidades inteligentes, *smart grids*, indústria 4.0, etc. Todos possuem como base um ou mais dispositivos IoT, os quais realizam quatro funções básicas: *sensoriamento, reação, coleta e comunicação* [Bashir 2017].

Os sistemas de IoT possuem, geralmente, cinco camadas: Aplicação, Serviços, Rede, Dispositivo e Física. Cada uma possui funções e componentes definidos. A primeira camada possui as aplicações de caráter financeiro, transporte, saúde, segurança, ambiental, etc.. Na camada de serviços estão tarefas como processamento de dados, gerenciamento de segurança e de fluxo de dados. As camadas de rede e dispositivos, tratam, respectivamente, da transmissão dos dados e dos sensores, atuadores e OIs. Por último, na camada física estão as pessoas, os carros, os animais, as salas e tudo mais que precise ser monitorado e controlado.

Como exemplo de tais sistemas, considere um cenário típico da cadeia de suprimentos (I) composta por: (a) Indústria; (b) Transporte até porto/aeroporto; (c) Transporte até um Centro de Distribuição; (d) Entrega em uma Loja e (e) Venda para um cliente. Este processo envolve diversos agentes desde (a) até (e). Cada um destes, possui um sistema e um banco de dados proprietários, onde guardam informações sobre os produtos. Para manter estes dados atualizados, uma equipe de pessoas trabalha, alimentando o sistema. Com o uso de IoT, todo processo é simplificado e os próprios OIs postam as informações sobre o seu status, local, data, etc. Adicionalmente, algumas categorias de OIs podem necessitar de atualização de software ou firmware (II). O próprio OI pode iniciar este processo ou ser acionado remotamente pelo fabricante para que receba a nova versão.

Esta é uma situação hipotética onde as próprias máquinas podem cuidar de si mesmas. No entanto, questões como confiança, segurança, transparência, auditabilidade, etc. estão carentes. Por consequência, atualmente, muitos autores defendem a inserção, nos sistemas de IoT, da solução de blockchain e de contratos inteligentes como forma de suprir estas lacunas [Brody and Pureswaran 2014].

No caso em tela, a blockchain proporciona um avanço. Cada um dos atores do cenário (I) e (II) enviam/recebem mensagens de forma segura, auditável e irrefutável. Os contratos convencionais são transformados em contratos inteligentes, predeterminando as condições e permitindo aos participantes autorizados o acesso às informações e o monitoramento eletrônico das transações. Especialmente no cenário (II), o fabricante pode implementar um SC que permita armazenar o hash da última atualização do firmware na rede. Os OIs ou possuem gravados ou descobrem por mecanismos adequados o endereço deste contrato Inteligente e podem verificar se há um novo firmware disponível, solicitando-o por meio de um novo hash, a um sistema de arquivos distribuído tipo P2P [Christidis and Devetsikiotis 2016].

5.7.2. Contratos Inteligentes

O termo *contrato inteligente* (*smart contract*) (SC) foi formalizado por Nick Szabo na década de 90, e significa: "um protocolo de transação informatizado que executa os termos de um contrato. Os objetivos gerais são satisfazer condições contratuais comuns (como condições de pagamento, ônus, confidencialidade e até mesmo cumprimento), minimizar as exceções maliciosas, acidentais e a necessidade de intermediários confiáveis. Os obje-

```
pragma solidity ^0.4.0;
contract TemperaturaIdeal{

    int public temperatura;
    address sensor;
    address termostato;

    event Instruction(address device, string instruction);
    event reward(address device, string instruction);

    function IoT (address sensor, address reg){
        sensor = _sensor;
        termostato=reg
    }

    function atualizaTemperatura (int temperatura){
        if(msg.sender != senor) throw;
        temperatura = temp;
        if(temperatura >=25)
            Instruction(termostato, "Violação");
        else reward(termostato,"Compliant")
    }
}
```

Figure 5.8. Contrato Inteligente para Verificação de Temperatura.

tivos econômicos relacionados incluem redução de perdas por fraude, arbitragens e custos de execução e outros custos de transação ” [Szabo 1997].

Na blockchain, os SCs tomam a forma de scripts, armazenados com endereçamento exclusivo na própria blockchain. Aciona-se um contrato inteligente endereçando uma transação para ele. Em seguida, ele é executado de maneira independente e automática, conforme a prescrição, em todos os nós da rede, de acordo com os dados incluídos na transação. Isto significa que cada nó, habilitado por contrato inteligente, está executando uma máquina virtual (MV) e que a rede blockchain atua como uma MV distribuída. [Christidis and Devetsikiotis 2016]. Na IoT, a principal vantagem do uso dos SC deve-se ao gerenciamento das interações entre as entidades inteligentes envolvidas, podendo realizar uma das 4 funções básicas: sensoriamento, reação, coleta e comunicação.

A Figura 5.8 exemplifica um SC para o Ethereum, usando o Solidity (linguagem de programação para SC, similar ao JavaScript). Neste exemplo, quando uma certa temperatura não for mantida, é gerada uma violação. Se a temperatura estiver no intervalo esperado, o nó é recompensado na forma de moedas virtuais.

5.7.3. Aplicações na IoT

Há diversas aplicações possíveis no casamento da IoT com a Blockchain. Governo eletrônico e voto eletrônico, segurança interna, controle de fronteiras, serviços cartoriais, judiciais e fazendários, além de identidade digital são alguns pontos que podem ser atendidos pelos SCs [Ølnes et al. 2017]. Infra-estrutura específica para a área de saúde e registros e acompanhamento médico em processos que exigem, dentre outras coisas, transparência, auditoria e controle são casos em que os SCs e IoT atuam [Salahuddin et al. 2017, Ekblaw et al. 2016]. Aplicações na área de finanças podem ser encontradas em [Peters and Panayi 2016] e para artes, música, cópias digitais e material multimídia em [Bhowmik and Feng 2017]. Nos serviços e infra-estrutura computacional, por exemplo, pode-se cobrar pelo uso de chamadas de API’s ou uso de espaço em disco [Christidis and Devetsikiotis 2016]. A camada da blockchain pode possibilitar funcionalidades como os micro-pagamentos entre dispositivos digitalmente aprimorados, através de criptografia ultra-leve e de SCs. Na

área de rastreabilidade de produtos, em geral, as diversas partes envolvidas possuem bancos de dados próprios para controlar a mobilidade de insumos; com o uso da blockchain surge um banco de dados compartilhado, com recursos criptográficos e auditável.

5.7.4. Segurança em Contratos Inteligentes

Os SCs interpretam o código de maneira objetiva e implacável - "O Código é a lei". No entanto, um conjunto de contratos inteligentes (DAO) escritos para uma plataforma de investimentos continha um erro no código e foi alvo de um ataque em junho de 2006. Isto desviou cerca de 50 milhões de dólares e obrigou o Ethereum a realizar um "*hard fork*" para promover uma recuperação, quebrando este paradgima [Bashir 2017]. Mas, fazer isto não é uma operação trivial, portanto, as principais ameaças aos contratos inteligentes precisam ser estudadas e minimizadas.

As principais fragilidades dos SCs, com ênfase na plataforma Ethereum, podem ser agrupadas em três níveis: Linguagem Solidity, Máquina Virtual Ethereum (MVE) e Blockchain). Todas elas podem ser exploradas para ataques e roubos de dados e dinheiro dos contratos [Atzei et al. 2017]. Para melhor compreensão, são introduzidos os conceitos de MVE e Gas. A MVE é o nome dado ao ambiente que a plataforma Ethereum disponibiliza para a execução das aplicações e contratos inteligentes, garantindo que eles não tenham acesso aos estados uns dos outros e que a comunicação possa ser estabelecida sem interferências prejudiciais. O Gas é uma forma de desacoplar o custo das transações no Ethereum do câmbio flutuante da criptomoeda Éther, estabelecendo um custo para cada trabalho computacional.

De acordo com [Atzei et al. 2017], as principais ameaças existentes em relação ao uso do Solidity são: (a) ***Primitivas call, send e delegatecall*** - Usadas no Solidity para invocar funções e transferir dinheiro, podem ter efeito colateral de invocar a função fallback do destinatário; (b) ***Exceptions*** - Há diversas situações onde uma *exception* pode ser disparada e o Solidity não responde de modo padrão a todas elas; (c) ***Gasless*** - O uso da função *send* para transferir ether de um contrato com o saldo de *gas* zerado; (d) ***Declaração de atributos*** - O compilador Solidity pode detectar alguns erros, como por exemplo, atribuir um valor inteiro a uma variável do tipo string; (e) ***Re-entrada*** - A atomicidade e a sequencialidade das transações podem induzir os programadores a acreditar que, quando uma função não recursiva é invocada, ela não pode ser reinserida antes de sua finalização. No entanto, isso nem sempre é o caso, porque o mecanismo de fallback pode permitir que um invasor entre novamente na função do chamador. Isso pode resultar em comportamentos inesperados e possivelmente também em loops de invocações que eventualmente consomem todo o *gas* e (f) ***Informações Sensíveis*** - Os campos nos contratos podem ser públicos ou privados. Declarar um campo como privado não garante seu sigilo. Isso porque, para definir o valor de um campo, os usuários devem enviar uma transação adequada aos mineradores, que a publicarão no blockchain. Como o blockchain é público, todos podem inspecionar o conteúdo da transação e inferir o novo valor do campo. Portanto, o SC deve usar mecanismos de criptografia adequados.

Em relação à MVE, tem-se os seguintes riscos: (a) ***Bugs imutáveis*** - Um contrato após publicado, não pode ser modificado. O comportamento em tempo de execução é garantido pelos protocolos de consenso, portanto, caso haja um bug, ele vai permanecer;

(b) ***Perda de ethers na transferência*** - Ao enviar ethers de uma SC para outro, o endereço pode não estar mais associado ao SC de destino ou ao usuário, fazendo com que os ethers se percam para sempre e (c) ***Tamanho da pilha de chamadas*** - Cada vez que um contrato invoca outro contrato, a pilha de chamadas associada à transação cresce um quadro. O limite é 1024 quadros. Quando esse limite é atingido, uma outra chamada lançará uma exceção. Sem o tratamento adequado, um adversário poderá ter sucesso em um ataque.

Por fim, as ameaças para os SCs em relação à Blockchain: (a) ***Estado do contrato*** - O valor dos campos e saldo determinam o estado do SC. Um usuário, ao invocar algum SC, não tem certeza sob seu estado, pois outras transações podem modificá-lo ou pode ter ocorrido um *fork*. Em alguns casos, isto pode gerar vulnerabilidades e proporcionar o roubo de ethers; (b) ***Geração aleatória*** - A execução dos bytecodes da MVE é determinística e na ausência de mau comportamento, todos os mineradores que executam uma transação terão os mesmos resultados. No entanto, alguns contratos, como por exemplo, loterias, jogos, etc., geram números pseudo-aleatórios com a mesma semente para todos os mineradores. Isto permite que todos tenham a mesma visão da Blockchain, proporcionando a um minerador malicioso influenciar a rede; (b) ***Restrições de Tempo*** - Diversos aplicativos usam restrições temporais para determinar ações permitidas ou obrigatórias no instante t de tempo usando timestamps do bloco, que são acordados por todos os mineradores. Os SCs podem recuperar o registro de data e hora em que o bloco foi extraído e todas as transações dentro de um bloco compartilham o mesmo timestamp. O minerador que cria um novo bloco pode escolher o timestamp a ser usado com um certo grau de arbitrariedade, ou seja, uma tolerância de 900 segundos. Caso ele detenha uma participação em um contrato, poderá obter uma vantagem, ao escolher um registro de data e hora adequado para o bloco que está explorando.

5.8. Aplicação em Névoa

De maneira similar a blockchain, um ambiente de névoa provê uma plataforma virtualizada em uma escala amplamente distribuída que oferece serviços entre os dispositivos finais e os centros de dados convencionais da computação em nuvem. Nesta seção, vamos introduzir o ambiente de névoa e as vantagens tecnológicas com a integração desses dois paradigmas. Como exemplos de aplicação, serão apresentados trabalhos em andamento sobre arquiteturas de névoa que empregam a blockchain na implementação de seus modelos e de suas funcionalidades.

5.8.1. Ambientes de Computação em Névoa

Um modelo baseado em nuvem tornou-se a abordagem padrão para a IoT. Entretanto, propostas recentes estão promovendo uma grande mudança em direção a uma arquitetura descentralizada [Coutinho et al. 2016]. A computação em névoa (*fog computing*) e borda (*edge computing*) estendem os serviços de nuvem para as extremidades da rede buscando superar as limitações das arquiteturas IoT. Sua motivação está apoiada em aplicações que exigem latência mínima e o processamento de uma grande quantidade de dados, o que é difícil de se realizar através da rede.

A Figura 5.9 apresenta a arquitetura horizontal do sistema de névoa. Sua infraestrutura envolve a execução de serviços e aplicativos entre os dispositivos de borda

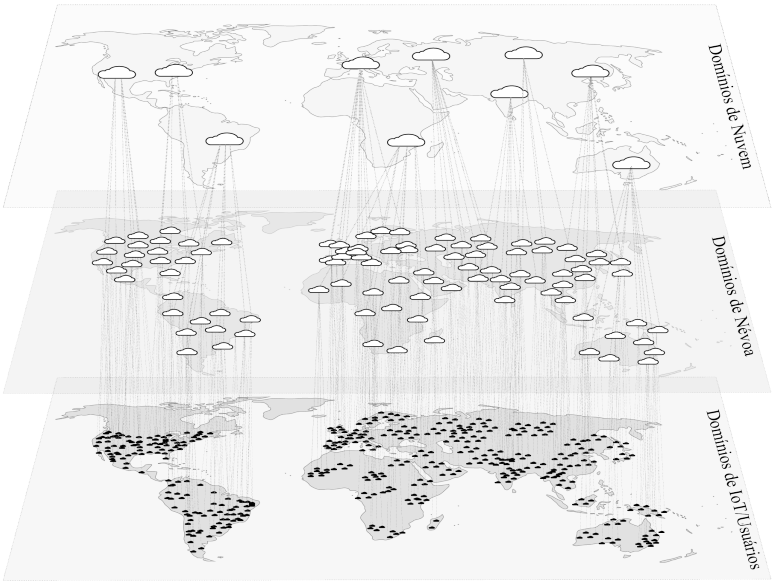

Figure 5.9. Arquitetura Horizontal do Sistema de Névoa.

e a nuvem. Eles são distribuídos hierarquicamente em uma infraestrutura de rede com pelo menos três camadas. Os componentes que podem fornecer recursos de processamento e armazenamento aos dispositivos de IoT são chamados nós de névoa. Eles ajudam no processamento de dados de aplicações de IoT e são responsáveis por filtrar e enviar periodicamente informações para a nuvem.

As aplicações podem usar recursos tanto da nuvem quanto da névoa para fornecer soluções inteligentes para usuários na borda da rede. O software de gerenciamento que coordena globalmente a infraestrutura de névoa é organizado em diferentes serviços que trabalham de forma distribuída e integrada. Seu principal objetivo é fornecer níveis aceitáveis de latência enquanto busca otimizar o desempenho das aplicações através de soluções eficientes que distribuem demandas de execução de tarefas em nós de névoa ótimos, bem localizados e disponíveis.

Espera-se que os ambientes de névoa suportem milhões de dispositivos de IoT e de usuários finais. Ambientes de névoa podem envolver um grande número de nós de névoa e aplicativos localizados em diferentes domínios de rede. Os sistemas de névoa precisam ser seguros, operacionais e escaláveis de forma elástica, ou seja, capazes de diminuir ou aumentar o número de recursos alocados quando necessário. Para atingir os objetivos das redes de IoT, faz-se necessária a existência de soluções autônomas e descentralizadas.

5.8.2. Integração Entre Sistemas de Névoa e Blockchain

A computação em névoa e a tecnologia blockchain são soluções independentes, voltadas à cenários de aplicação específicos. Entretanto, esses ambientes apresentam características fundamentais semelhantes: (*i*) são redes distribuídas em larga escala, que podem ser formadas por milhares de nós; (*ii*) precisam oferecer conectividade generalizada entre dispositivos geograficamente dispersos em uma região, país ou no mundo inteiro; (*iii*) os nós que formam a rede devem possuir recursos de processamento e de armazenamento para suporte à serviços e aplicações. (*iv*) são ambientes heterogêneos, onde os nós da rede podem apresentar diferentes capacidades, interfaces e configurações de hardware.

Existem diferentes motivações para aplicação da tecnologia blockchain no projeto e no desenvolvimento de sistemas de computação em névoa. Algumas das motivações identificadas em [Sharma et al. 2017a] são indicadas a seguir:

- *Descentralização*: Criando confiança sem a necessidade de uma entidade intermediária confiável, acredita-se que a blockchain promoverá a revisão dos sistemas de nuvem atuais e facilitará a implementação adequada de sistemas em névoa através de soluções autônomas e descentralizadas;

- *Acesso sobre demanda*: a blockchain pode facilitar o acesso sob demanda e a cobrança no uso de recursos utilizados. Por exemplo, executando o algoritmo de recursos sob demanda em um contrato inteligente, o pagamento ocorrerá automaticamente após a conclusão do serviço solicitado;

- *Redundância e disponibilidade*: de forma análoga aos sistemas P2P, a blockchain permite construir sistemas de névoa onde os dados e serviços da aplicação são distribuídos em dezenas de nós distintos, de maneira inteligente e bem localizada, tornando extremamente difícil causar interrupções significativas;

- *Privacidade e Integridade*: na tecnologia blockchain cada usuário gerencia suas próprias chaves e cada nó de bloco armazena apenas fragmentos criptografados de dados do usuário. É possível oferecer privacidade aos usuários sem que os terceiros envolvidos tenham acesso e controle dos dados;

- *Qualidade de Serviço*: usando a tecnologia blockchain, é possível oferecer rastreabilidade do uso de recursos visando a verificação adequada do contrato de nível de serviço tanto pelo cliente quanto pelo provedor de serviços;

- *Cooperação e Incentivos*: Os ambientes de névoa com suporte a blockchain podem fomentar novos modelos que ofereçam incentivos aos seus usuários e cooperação entre provedores. Neste ambiente, provedores independentes podem cooperar entre si de forma transparente para construir uma infraestrutura de TI escalável e oferecer sua capacidade computacional excedente aos usuários, enquanto os pagamentos são feitos através da blockchain;

- *Baixo custo*: Fazendo um paralelo entre as infraestruturas financeiras e as de computação em névoa, a tecnologia blockchain pode ser empregada para substituir os sistemas de nuvem existentes por sistemas que cobram taxas menores pelos serviços. Isso é possível devido à eficiência - sobre cada nó da névoa/blockchain incide custos que, quando somados, são menores do que os custos dos provedores de nuvem atuais, salientando que esses provedores ainda tem de realizar lucros consideráveis para seus investidores.

5.8.3. Soluções em Névoa Baseadas em Blockchain

Pesquisas atuais que propõem a integração dessas duas tecnologias tem como base às vantagens mútuas obtidas pela combinação de suas funcionalidades. Dependendo do escopo da pesquisa, as propostas podem ter como objetivo geral: o desenvolvimento da plataforma de névoa; ou o emprego da névoa/blockchain para aperfeiçoar modelos, sistemas

ou aplicações distribuídas existentes. A seguir, descrevemos os aspectos principais da aplicação da tecnologia blockchain na arquitetura de nuvem/névoa proposta por [Sharma et al. 2017a]. Neste trabalho, é empregado um modelo aberto na construção de um ambiente formado por diferentes provedores de serviço. Este modelo permite o amplo uso de uma infraestrutura de armazenamento distribuída pelos seus usuários.

Modelo de Negócio - O modelo proposto em [Sharma et al. 2017a] reúne dados de fornecedores e consumidores de recursos, consistindo nas quatro etapas a seguir: (*i*) seleção de fornecedores de recursos, (*ii*) prestação de serviços, (*iii*) registro de transações e pagamento. Na primeira etapa, o usuário seleciona o fornecedor dos recursos a partir de uma pesquisa sobre os provedores de serviço disponíveis na névoa baseada em blockchain. Após o processo de alocação, o provedor fornece os recursos requisitados para o usuário. Após fornecer os recursos, o provedor registra a transação na blockchain compartilhada entre todos os provedores de serviço de mesmo nível. Por fim, o usuário recompensa o provedor pelos serviços prestados e utilização dos recursos.

Protocolo de Consenso - De maneira diferente de uma blockchain convencional como o Bitcoin, que depende de um protocolo baseado em PoW para validar suas transações, a névoa baseada em blockchain proposta em [Sharma et al. 2017a] utiliza um protocolo de consenso nomeado como prova de serviço (*Proof-of-Service*) para validar as contribuições e permitir que as transações associadas possam ser incluídas na blockchain. Uma contribuição é uma ação que ocorre fora da blockchain como o processamento de uma computação, a transferência de um arquivo ou o fornecimento de dados. Isso levará à ocorrência de transações entre os membros da rede. Este protocolo usa a técnica de 2-*hop* blockchain proposta por [Duong et al. 2016] que combina os mecanismos de PoW e PoS. A segurança desta blockchain depende se o participante legítimo controla a maior parte dos recursos totais, que incluem tanto as participações quanto o poder de computação.

Alocação de Recursos: No projeto de sistemas de névoa, o algoritmo de alocação de recursos é um elemento essencial para garantir a qualidade dos serviços prestados [Coutinho et al. 2016]. O modelo proposto em [Sharma et al. 2017a] vincula uma solicitação de recurso a uma oferta de recurso utilizando contratos inteligentes. Um contrato de alocação descreve os requisitos para executar a tarefa como o ambiente de virtualização esperado, o tempo de execução da GPU, o espaço em disco necessário, etc. Um contrato de alocação pode estar relacionado à diferentes algoritmos de alocação de recursos.

Existem trabalhos que não possuem como objetivo geral o projeto de uma plataforma de névoa, mas que utilizam o modelo de névoa para propor sistemas inovadores baseados em blockchain. Por exemplo, em [Huang et al. 2018] o objetivo do trabalho é aprimorar o sistema tradicional de caixa eletrônico que requer uma autoridade confiável, ou seja, um banco para gerar o *token* de pagamento. Os usuários do sistema (*outsourcers*) podem fazer diretamente uma transação para os nós de névoa (*workers*). Nesta proposta, é apresentado um esquema de pagamento justo baseado em bitcoin entre *outsourcers* e *workers* que ofertam seus recursos para tarefas de computação distribuídas. No protocolo proposto, os *outsourcers* e os *workers* não confiam uns nos outros. Foram utilizados contratos inteligentes para garantir que, independente do quão malicioso seja o comportamento de um *outsourcer*, um *worker* honesto que executa a tarefa sempre receberá o pagamento. Diferentemente de trabalhos anteriores, não existe uma entidade centrali-

zadora envolvida no pagamento. O protocolo proposto foi mais eficiente em termos de processamento e comunicação em relação à soluções vigentes analisadas.

5.9. Desafios e Perspectivas

Os desafios para a consolidação e melhoria da blockchain, tanto científicos quanto tecnológicos, são inúmeros e diversos. Os principais envolvem sustentabilidade, escalabilidade [Eyal et al. 2016], desempenho [Dinh et al. 2017b], adaptabilidade, armazenamento [Dinh et al. 2017a], segurança, privacidade, regulação, ordenação das transações [Natoli and Gramoli 2016], etc. Muitos desafios foram apresentados ao longo de todo o texto. Nessa seção, faremos um breve panorama de alguns deles, considerando, sobretudo, o serviço de coordenação e consenso.

Na blockchain pública os maiores desafios são a sustentabilidade, escalabilidade e desempenho da rede. Alguns trabalhos interessantes avaliam o impacto desses fatores para a adoção abrangente da tecnologia [Dinh et al. 2017b, Vukolic 2015, Croman et al. 2016]. Na Bitcoin, a sustentabilidade é crítica devido ao alto gasto de energia, com exigência de poder computacional cada vez mais crescente. As propostas de consenso PoX trazem uma resposta a esse ponto. A escalabilidade está diretamente relacionada ao desempenho, no que diz respeito à latência (tempo de confirmação de uma transação) e vazão (*throughput*) (taxa de confirmação de transações no tempo). A vazão depende diretamente do tamanho máximo do bloco e do intervalo de tempo entre blocos. Na Bitcoin, esse valor é ainda muito baixo, comparativamente às demandas de negócios; atualmente, a latência é de 10 min. e a vazão máxima é de $3-7$ tx/seg, enquanto sistemas de transações financeiros, como operadoras de cartão de crédito, exigem para além de 10^3 tx/seg. Já a Ethereum apresenta melhor vazão (blocos são decididos a cada $12-15$ seg), porém o aumento da frequência de decisões proporciona maior número de *forks* na rede.

Alguns trabalhos recentes foram propostos para mitigar o baixo desempenho da Bitcoin. O *Bitcoin-NG* [Eyal et al. 2016] advoga a realização do consenso em duas fases distintas: (i) eleição do líder e (ii) serialização de transações, de tal forma que, durante um período estabelecido (*epoch*), o líder teria direito a serializar transações, até que um novo líder seja selecionado; as análises de desempenho apresentadas são favoráveis à estratégia. O *Bitcoin lightening network* [Poon and Dryja 2016] propõe a realização de uma parte das transações off-blockchain. As transações ocorreriam em rede de canais de micro-pagamento e a transferência de valor passaria a ser realizada off-line.

A blockchain privada/federada tem excelentes taxas de desempenho: a latência é instantânea, proporcional à latência da rede, enquanto a vazão de confirmação é da ordem de 10^4 tx/seg. Entretanto, diferentemente da PoW, os protocolos de consenso BFT, adequados para as redes controladas não escalam com o aumento do número de nós; seu bom desempenho fica restrito a um conjunto limitado de $20-30$ nós.

[Vukolić 2017] destaca algumas limitações de projeto da blockchain privada, que permitem aperfeiçoamento da tecnologia: (i) *Execução sequencial*: a execução em série das transações e contratos inteligentes, após a realização do consenso, é bastante limitante, contribuindo negativamente para a melhoria da vazão. Pesquisas nas áreas de compução paralela e fragmentação de dados (*data sharding*) são um caminho para superar esse aspecto; (ii) *Execução não-determinística*: para preservação da consistência do *ledger*, é

importante que a execução da máquina de estados replicada ativamente seja determinística, de tal maneira a forçar com que todas as réplicas executem exatamente os mesmos passos, na mesma ordem. Ocorre que os sistemas de execução e as próprias linguagens de programação de propósito geral incorporam elementos de não determinismo. O desafio consiste em conciliar o não determinismo inerente aos sistemas com um modelo de falhas e confiança realista para as aplicações na blockchain; (iii) *Execução em todos os nós da rede*: o padrão de execução dos contratos inteligentes em todos os nós da rede é o mais comum, sobretudo em redes abertas. Mas, em redes controladas esse padrão pode ser enfraquecido, de tal forma a restringir a execução dos contratos a apenas um subconjunto de nós. A definição de um modelo adequado para estabelecimento do conjunto de nós, da sua quantidade e tipo de interação para essa execução é um desafio interessante de projeto; (iv) *Modelo de confiança flexível*: deve-se flexibilizar o modelo, de tal forma a separar o modelo de execução do consenso do modelo de execução dos contratos inteligentes. As restrições impostas para a camada inferior de coordenação e consenso (como a limitação de $f < n/3$ nós maliciosos na rede) não precisam ser as mesmas para execução dos contratos inteligentes (cujas aplicações podem exigir limitações diferenciadas). (v) *Consenso hard-coded*: a grande totalidade das plataformas de blockchain adota consensos cujo código é imbricado de tal forma que a sua manutenção, evolução e substituição é muito dificultada. Como não há protocolos únicos para todas as aplicações, o desejado para uma plataforma de blockchain genérica seria prover um consenso modular, passível de substituição, tal como se observa no projeto Hyperledger.

A aplicação da tecnologia blockchain na promoção de uma infra-estrutura de confiança abrange diversas áreas da computação. Neste capítulo, abordamos algumas aplicações emergentes em redes de computadores, computação em névoa e IoT. Em redes, ela é vislumbrada na implementação de planos de controles mais robustos por meio de controladores distribuídos de redes definidas por software. Além disso, blockchains têm sido utilizadas no suporte à implementação e uso de funções de redes virtualizadas. Por fim, é preciso destacar o uso da tecnologia em paradigmas de redes recentes, tais como as redes centradas na informação e na promoção da segurança em redes.

Trabalhos recentes têm proposto uma arquitetura de referência e componentes de uma plataforma de névoa. No entanto, as soluções propostas continuam a ser desenvolvidas como prova de conceito, implementadas em ambientes limitados e condições restritas [Coutinho et al. 2018]. Neste sentido, a blockchain poderá ser o arcabouço que dará suporte ao processamento e coordenação das transações entre os elementos da névoa, tornando viável a operação em larga escala de aplicações voltadas à IoT. Ao mesmo tempo, a névoa pode oferecer um ambiente estruturado para implementação de diferentes soluções blockchain. Para a IoT, a blockchain vem ofertar uma nova camada computacional para compartilhamento e análise segura dos dados, com garantias de privacidade. Assim, essa camada pode ser utilizada para autenticar, autorizar, controlar e auditar os dados gerados pelos objetos inteligentes [Christidis and Devetsikiotis 2016].

References

[Agarwal et al. 2014] Agarwal, K., Rozner, E., Dixon, C., and Carter, J. (2014). Sdn traceroute: Tracing sdn forwarding without changing network behavior. In *3rd Workshop on Hot Topics in SDN*, HotSDN '14, pages 145–150, New York, NY, USA. ACM.

[Alvarenga et al. 2018] Alvarenga, I. D., Rebello, G. A. F., and Duarte, C. (2018). Securing configuration management and migration of virtual network functions using blockchain. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*.

[Antonopoulos 2017] Antonopoulos, A. (2017). *Mastering Bitcoin: Programming the Open Blockchain*. O'reilly, 2nd edition.

[Atzei et al. 2017] Atzei, N., Bartoletti, M., and Cimoli, T. (2017). A survey of attacks on ethereum smart contracts (sok). In Maffei, M. and Ryan, M., editors, *Principles of Security and Trust*, pages 164–186, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Bag et al. 2017] Bag, S., Ruj, S., and Sakurai, K. (2017). Bitcoin block withholding attack: Analysis and mitigation. *IEEE Transactions on Information Forensics and Security*, 12(8):1967–1978.

[Baird 2016] Baird, L. (2016). The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance. Technical Report SWIRLDS-TR-2016-01, SWIRLDS Project.

[Bano et al. 2017] Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., and Danezis, G. (2017). Sok: Consensus in the age of blockchains. In *arxiv.org/abs/1711.03936*.

[Bashir 2017] Bashir, I. (2017). *Mastering Blockchain: Distributed ledgers, decentralization and smart contracts explained*. Packt Publishing.

[Ben-Or 1983] Ben-Or, M. (1983). Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *Proc. of 2rd ACM Symp. on Principles of Distributed Computing*, pages 27–30.

[Bentov et al. 2016] Bentov, I., Pass, R., and Shi, E. (2016). Snow white: Provably secure proofs of stake. *IACR Cryptology ePrint Archive*, 2016:919.

[Bessani et al. 2014] Bessani, A. N., Sousa, J., and Alchieri, E. A. P. (2014). State machine replication for the masses with bft-smart. *44th IEEE/IFIP Int. Conf. on Dependable Systems and Networks*, pages 355–362.

[Bhowmik and Feng 2017] Bhowmik, D. and Feng, T. (2017). The multimedia blockchain: A distributed and tamper-proof media transaction framework. In *Digital Signal Processing (DSP), 2017 22nd International Conference on*, pages 1–5. IEEE.

[Blenk et al. 2016] Blenk, A., Basta, A., Reisslein, M., and Kellerer, W. (2016). Survey on network virtualization hypervisors for software defined networking. *IEEE Communications Surveys Tutorials*, 18(1):655–685.

[Bonneau 2016] Bonneau, J. (2016). Why buy when you can rent? In *International Conference on Financial Cryptography and Data Security*, pages 19–26. Springer.

[Bonneau et al. 2015] Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., and Felten, E. W. (2015). Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. *IEEE Symposium on Security and Privacy*, pages 104–121.

[Bozic et al. 2016] Bozic, N., Pujolle, G., and Secci, S. (2016). A tutorial on blockchain and applications to secure network control-planes. In *3rd Smart Cloud Networks Systems (SCNS)*, pages 1–8.

[Bozic et al. 2017] Bozic, N., Pujolle, G., and Secci, S. (2017). Securing virtual machine orchestration with blockchains. In *1st Cyber Security in Networking Conference (CSNet)*, pages 1–8.

[Brewer 2000] Brewer, E. A. (2000). Towards robust distributed systems (abstract). In *Symp. on Principles of Distributed Computing (PODC)*, pages 7–7. ACM.

[Brody and Pureswaran 2014] Brody, P. and Pureswaran, V. (2014). Device democracy: Saving the future of the internet of things. *IBM, September*.

[Buterin 2014] Buterin, V. (2014). A next-generation smart contract and decentralized application platform. Technical report, White paper.

[Cachin et al. 2016] Cachin, C., Caro, A. D., Christidis, K., and Yellick, J. (2016). Architecture of the hyperledger blockchain fabric. Technical report, IBM Research - Zurich.

[Cachin et al. 2000] Cachin, C., Kursawe, K., and Shoup, V. (2000). Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. *Journal of Cryptology*, 18:219–246.

[Cachin and Vukoli'c 2017] Cachin, C. and Vukoli'c, M. (2017). Blockchains consensus protocols in the wild. In *IBM Research – Zurich*.

[Castro and Liskov 1999] Castro, M. and Liskov, B. (1999). Practical byzantine fault tolerance. In *3rd Symp. on Operating Systems Design and Implementation*, OSDI'99, pages 173–186, Berkeley, CA, USA. USENIX Association.

[Castro and Liskov 2002] Castro, M. and Liskov, B. (2002). Practical Byzantine fault-tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461.

[Chandra and Toueg 1996] Chandra, T. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267.

[Chandra et al. 2007] Chandra, T. D., Griesemer, R., and Redstone, J. (2007). Paxos made live: an engineering perspective. In *Symp. on Principles of Distributed Computing (PODC)*, pages 398–407.

[Chandra et al. 1996] Chandra, T. D., Hadzilacos, V., and Toueg, S. (1996). The weakest failure detector for solving consensus. *Journal of the ACM*, 43(4):685–722.

[Charron-Bost et al. 2010] Charron-Bost, B., Pedone, F., and Schiper, A. e. (2010). *Replication Theory and Practice*, volume 5959. Springer.

[Christidis and Devetsikiotis 2016] Christidis, K. and Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303.

[Chu 1998] Chu, F. (1998). Reducing ω to $\diamond s$. *Inf. Process. Lett.*, 67(6):289–293.

[Compagno et al. 2013] Compagno, A., Conti, M., Gasti, P., and Tsudik, G. (2013). Poseidon: Mitigating interest flooding DDoS attacks in Named Data Networking. In *38th IEEE Conf. on Local Computer Networks*, pages 630–638.

[Conti et al. 2017] Conti, M., E, S. K., Lal, C., and Ruj, S. (2017). A survey on security and privacy issues of bitcoin. *CoRR*, abs/1706.00916.

[Copeland and Zhong 2014] Copeland, C. T. and Zhong, H. (2014). Tangaroa: a byzantine fault tolerant raft.

[Courtois and Bahack 2014] Courtois, N. T. and Bahack, L. (2014). On subversive miner strategies and block withholding attack in bitcoin digital currency. *CoRR*, abs/1402.1718.

[Coutinho et al. 2018] Coutinho, A., Greve, F., Prazeres, C., and Cardoso, J. (2018). Fogbed: A rapid-prototyping emulation environment for fog computing. In *(ICC) Int. Conf. on Communications*. IEEE.

[Coutinho et al. 2016] Coutinho, A. A. T. R., Carneiro, E. O., and Greve, F. (2016). Computação em névoa: Conceitos, aplicações e desafios. In *Minicursos do XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos*, chapter 6, pages 266–315. SBC.

[Crain et al. 2017] Crain, T., Gramoli, V., Larrea, M., and Raynal, M. (2017). (leader/randomization/signature)-free byzantine consensus for consortium blockchains. *CoRR*, abs/1702.03068.

[Croman et al. 2016] Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A. E., Miller, A., Saxena, P., Shi, E., Sirer, E. G., Song, D. X., and Wattenhofer, R. (2016). On scaling decentralized blockchains. In *Proc. 3rd Workshop on Bitcoin and Blockchain Research*.

[Dai et al. 2013] Dai, H., Wang, Y., Fan, J., and Liu, B. (2013). Mitigate DDoS attacks in NDN by interest traceback. In *IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 381–386.

[Défago et al. 2004] Défago, X., Schiper, A., and Urban, P. (2004). Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Computing Surveys*, 4:372–421.

[Dinh et al. 2017a] Dinh, T. T. A., Liu, R., Zhang, M., Chen, G., Ooi, B. C., and Wang, J. (2017a). Untangling blockchain: A data processing view of blockchain systems. *IEEE Transactions on Knowledge and Data Engineering*, pages 99–.

[Dinh et al. 2017b] Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C., and Tan, K.-L. (2017b). Blockbench: A framework for analyzing private blockchains. In *SIGMOD Conference*.

[Douceur 2002] Douceur, J. (2002). The sybil attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer-Verlag.

[Duong et al. 2016] Duong, T., Fan, L., and Zhou, H.-S. (2016). 2-hop blockchain: Combining proof-of-work and proof-of-stake securely. Technical report, Cryptology ePrint Archive, Report 2016/716, 2016. https://eprint. iacr. org/2016/716.

[Dwork et al. 1988] Dwork, C., Lynch, N. A., and Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of ACM*, 35(2):288–322.

[Ekblaw et al. 2016] Ekblaw, A., Azaria, A., Halamka, J. D., and Lippman, A. (2016). A case study for blockchain in healthcare:"medrec" prototype for electronic health records and medical research data. In *IEEE Open & Big Data Conference*, volume 13, page 13.

[Eyal 2015] Eyal, I. (2015). The miner's dilemma. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 89–103. IEEE.

[Eyal et al. 2016] Eyal, I., Gencer, A. E., Sirer, E. G., and van Renesse, R. (2016). Bitcoin-ng: A scalable blockchain protocol. In *Proc. of the 13th Usenix Conf. on Networked Systems Design and Implementation (NSDI)*, pages 45–59.

[Eyal and Sirer 2014] Eyal, I. and Sirer, E. G. (2014). Majority is not enough: Bitcoin mining is vulnerable. In *Int. Conf. on financial cryptography and data security*, pages 436–454. Springer.

[Fabric 2017] Fabric, H. (2017). What is hyperledger fabric.

[Fischer et al. 1985] Fischer, M. J., Lynch, N. A., and Paterson, M. D. (1985). Impossibility of distributed consensus with one faulty process. *Journal of ACM*, 32(2):374–382.

[Fotiou and Polyzos 2016a] Fotiou, N. and Polyzos, G. C. (2016a). Decentralized name-based security for content distribution using blockchains. In *IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 415–420.

[Fotiou and Polyzos 2016b] Fotiou, N. and Polyzos, G. C. (2016b). Decentralized name-based security for content distribution using blockchains. In *Conf. on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 415–420. IEEE.

[Frank et al. 2014] Frank, L., Pedersen, R. U., Frank, C. H., and Larsson, N. J. (2014). The cap theorem versus databases with relaxed acid properties. In *8th Int. Conf. on Ubiquitous Information Management and Communication (ICUIM)*, pages 1–7. ACM.

[Gafni and Lamport 2003] Gafni, E. and Lamport, L. (2003). Disk paxos. *Distributed Computing*, 16(1):1–20.

[Gilbert and Lynch 2012] Gilbert, S. and Lynch, N. A. (2012). Perspectives on the cap theorem. *IEEE Computer*, 45:30–36.

[Gramoli 2017] Gramoli, V. (2017). From blockchain consensus back to byzantine consensus. *Future Generation of Computer Systems - available on line Sep/2017*.

[Greve 2005] Greve, F. (2005). Protocolos fundamentais para o desenvolvimento de aplicações robustas. In *Minicursos do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos*, chapter 5, pages 330–398. SBC.

[Guerraoui et al. 2008] Guerraoui, R., Knezevic, N., Quéma, V., and Vukolic, M. (2008). The next 700 bft protocols. *ACM Trans. Comput. Syst.*, 32:12:1–12:45.

[Guerraoui and Raynal 2007] Guerraoui, R. and Raynal, M. (2007). The alpha of indulgent consensus. *Comput. J.*, 50(1):53–67.

[Huang et al. 2018] Huang, H., Chen, X., Wu, Q., Huang, X., and Shen, J. (2018). Bitcoin-based fair payments for outsourcing computations of fog devices. *Future Generation Computer Systems*, 78:850–858.

[Hunt et al. 2010] Hunt, P., Konar, M., Junqueira, F. P., and Reed, B. (2010). Zookeeper: Wait-free coordination for internet-scale systems. In *USENIX Annual Technical Conference*, pages 11–11.

[Jacobson et al. 2009] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009). Networking named content. In *5th Int. Conf. on Emerging networking experiments and technologies*, pages 1–12. ACM.

[Jin et al. 2017a] Jin, T., Zhang, X., Liu, Y., and Lei, K. (2017a). Blockndn: A bitcoin blockchain decentralized system over named data networking. In *9th Int. Conf. on Ubiquitous and Future Networks (ICUFN)*, pages 75–80.

[Jin et al. 2017b] Jin, T., Zhang, X., Liu, Y., and Lei, K. (2017b). Blockndn: A bitcoin blockchain decentralized system over named data networking. In *9th Int. Conf. on Ubiquitous and Future Networks (ICUFN)*, pages 75–80. IEEE.

[Karame et al. 2012] Karame, G. O., Androulaki, E., and Capkun, S. (2012). Double-spending fast payments in bitcoin. In *ACM Conf. on Computer and communications security*, pages 906–917. ACM.

[Kiayias et al. 2016] Kiayias, A., Konstantinou, I., Russell, A., David, B. M., and Oliynykov, R. (2016). A provably secure proof-of-stake blockchain protocol. *IACR Cryptology ePrint Archive*, 2016:889.

[Kreutz et al. 2015] Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

[Kwon et al. 2017] Kwon, Y., Kim, D., Son, Y., Vasserman, E., and Kim, Y. (2017). Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin. In *ACM SIGSAC Conf. on Computer and Communications Security*, pages 195–209. ACM.

[Lamport 1998] Lamport, L. (1998). The part-time parliament. *ACM Transactions Computer Systems*, 16(2):133–169.

[Lamport et al. 1982] Lamport, L., Shostak, R., and Pease, M. (1982). The Byzantine generals problem. *ACM Trans. on Programing Languages and Systems*, 4(3):382–401.

[Lampson 2001] Lampson, B. (2001). The abcd's of paxos. In *Proceedings of the 20th annual ACM Symposium on Principles of Distributed Computing*, page 13.

[Li and Bi 2014] Li, Z. and Bi, J. (2014). Interest cash: an application-based countermeasure against interest flooding for dynamic content in named data networking. In *9th Int. Conf. on Future Internet Technologies*, page 2. ACM.

[Mijumbi et al. 2016] Mijumbi, R., Serrat, J., Gorricho, J. L., Bouten, N., Turck, F. D., and Boutaba, R. (2016). Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262.

[Miller et al. 2014] Miller, A., Juels, A., Shi, E., Parno, B., and Katz, J. (2014). Permacoin: Repurposing bitcoin work for data preservation. *2014 IEEE Symposium on Security and Privacy*, pages 475–490.

[Nakamoto 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Technical report, Bitcoin Org.

[Narayanan et al. 2016] Narayanan, A., Bonneau, J., Felten, E., Miller, A., and Goldfeder, S. (2016). *Bitcoin and cryptocurrency technologies*. Princeton University Press.

[Natoli and Gramoli 2016] Natoli, C. and Gramoli, V. (2016). The blockchain anomaly. *IEEE 15th Int. Symp. on Network Computing and Applications (NCA)*, pages 310–317.

[Nguyen et al. 2017] Nguyen, B. Q., Androulaki, E., and et. al., A. D. C. (2017). Hyperledger: Blockchain. Hyperledger Org.

[Nguyen et al. 2018] Nguyen, B. Q., Androulaki, E., and et. al., A. D. C. (2018). Hyperledger: Protocol specification. Hyperledger Org.

[Ongaro and Ousterhout 2014] Ongaro, D. and Ousterhout, J. K. (2014). In search of an understandable consensus algorithm. In *USENIX Annual Technical Conference*, pages 305–320.

[Paillisse et al. 2017] Paillisse, J., Rodriguez-Natal, A., Ermagan, V., Maino, F., and Cabellos, A. (2017). An analysis of the applicability of blockchain to secure ip addresses allocation, delegation and bindings. In *Internet draft (draft-paillisse-sidrops-blockchain-01)*. IETF.

[Peters and Panayi 2016] Peters, G. W. and Panayi, E. (2016). Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money. In *Banking Beyond Banks and Money*, pages 239–278. Springer.

[Poon and Dryja 2016] Poon, J. and Dryja, T. (2016). The bitcoin lightning network: Scalable off-chain instant payments. Technical report, Lightning Network.

[Rao et al. 2011] Rao, J., Shekita, E. J., and Tata, S. (2011). Using paxos to build a scalable, consistent, and highly available datastore. *Proc. of the VLDB Endowment (PVLDB)*, 4:243–254.

[Rosenfeld 2011] Rosenfeld, M. (2011). Analysis of bitcoin pooled mining reward systems. *CoRR*, abs/1112.4980.

[Salah and Strufe 2016] Salah, H. and Strufe, T. (2016). Evaluating and mitigating a collusive version of the interest flooding attack in ndn. In *IEEE Symp. on Computers and Communication (ISCC)*, pages 938–945.

[Salahuddin et al. 2017] Salahuddin, M. A., Al-Fuqaha, A., Guizani, M., Shuaib, K., and Sallabi, F. (2017). Softwarization of internet of things infrastructure for secure and smart healthcare. *Computer*, 50(7):74–79.

[Sankar et al. 2017] Sankar, L. S., Sindhu, M., and Sethumadhavan, M. (2017). Survey of consensus protocols on blockchain applications. In *4th Int. Conf. on Advanced Computing and Communication Systems (ICACCS)*, pages 1–5.

[Santos et al. 2006] Santos, B. P., Silva, L. A. M., and Celes, C. S. F. S. e. a. (2006). Internet das coisas: da teoria á prática. In SBC, editor, *Minicursos XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos*, pages 1–50. SBC.

[Schneider 1990] Schneider, F. B. (1990). Implementing fault-tolerant service using the state machine aproach: A tutorial. *ACM Computing Surveys*, 22(4):299–319.

[Seixas et al. 2017] Seixas, N. F. S., Ribeiro, A., and Sampaio, L. (2017). Um modelo de rede centrada na informação resiliente a ataques de negação de serviços por inundação de interesses. In *XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, Belém.

[Sharma et al. 2017a] Sharma, P. K., Chen, M. Y., and Park, J. H. (2017a). A software defined fog node based distributed blockchain cloud architecture for iot. *IEEE Access*, PP(99):1–1.

[Sharma et al. 2017b] Sharma, P. K., Singh, S., Jeong, Y. S., and Park, J. H. (2017b). Distblocknet: A distributed blockchains-based secure sdn architecture for iot networks. *IEEE Communications Magazine*, 55(9):78–85.

[Sousa et al. 2018] Sousa, J., Bessani, A., and Vukolic, M. (2018). A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In *DSN'18: The IEEE/IFIP Int. Conf. on Dependable Systems and Networks*.

[Szabo 1997] Szabo, N. (1997). Formalizing and securing relationships on public networks. *First Monday*, 2(9).

[Tourani et al. 2017] Tourani, R., Misra, S., Mick, T., and Panwar, G. (2017). Security, privacy, and access control in information-centric networking: A survey. *IEEE Communications Surveys & Tutorials*.

[Vukolic 2015] Vukolic, M. (2015). The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *Open Problems in Network Security (iNetSeC)*, volume 9591, pages 112–125. Lecture Notes in Computer Science - Springer.

[Vukolić 2017] Vukolić, M. (2017). Rethinking permissioned blockchains: Invited paper. In *1st ACM Workshop on Blockchains, Cryptocurrencies and Contracts (BCC)*.

[Wang et al. 2014] Wang, K., Zhou, H., Qin, Y., and Zhang, H. (2014). Cooperative-filter: countering interest flooding attacks in named data networking. *Soft Computing*, 18(9):1803–1813.

[Xu et al. 2016] Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A. B., and Chen, S. (2016). The blockchain as a software connector. *13th Working IEEE/IFIP Conf. on Software Architecture (WICSA)*, pages 182–191.

[Xylomenos et al. 2014] Xylomenos, G., Ververidis, C. N., Siris, V. A., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K. V., and Polyzos, G. C. (2014). A survey of information-centric networking research. *IEEE Communications Surveys & Tutorials*, 16(2):1024–1049.

[Yang et al. 2017] Yang, H., Zheng, H., Zhang, J., Wu, Y., Lee, Y., and Ji, Y. (2017). Blockchain-based trusted authentication in cloud radio over fiber network for 5g. In *16th Int. Conf. on Optical Communications and Networks (ICOCN)*, pages 1–3.

[Yin et al. 2018] Yin, H., Guo, D., Wang, K., Jiang, Z., Lyu, Y., and Xing, J. (2018). Hyperconnected network: A decentralized trusted computing and networking paradigm. *IEEE Network*, 32(1):112–117.

[Zhang et al. 2014] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Crowley, P., Papadopoulos, C., Wang, L., Zhang, B., et al. (2014). Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73.

[Ølnes et al. 2017] Ølnes, S., Ubacht, J., and Janssen, M. (2017). Blockchain in government: Benefits and implications of distributed ledger technology for information sharing. *Government Information Quarterly*, 34(3):355 – 364.