UNIVERSIDADE FEDERAL FLUMINENSE

GABRIEL REIS CARRARA

Provendo Confiabilidade para Mecanismos de Consenso Através do Uso de Redes Definidas por Software

NITERÓI

2020

UNIVERSIDADE FEDERAL FLUMINENSE

GABRIEL REIS CARRARA

Provendo Confiabilidade para Mecanismos de Consenso Através do Uso de Redes Definidas por Software

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Sistemas de Computação

Orientador:
Prof. Célio Vinicius Neves de Albuquerque, Ph.D.

Co-orientador:
Prof. Diogo Menezes Ferrazani Mattos, D.Sc.

NITERÓI

2020

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

C313 Carrara, Gabriel Reis
Provendo confiabilidade para mecanismos de consenso através do uso de redes definidas por software / Gabriel Reis Carrara. – Niterói, RJ : [s.n.], 2020.
64 f.

Dissertação (Mestrado em Computação) - Universidade Federal Fluminense, 2020.
Orientadores: Célio Vinicius Neves de Albuquerque, Diogo Menezes Ferrazani Mattos.

1. Redes de computadores. 2. Processamento distribuído. 3. Confiabilidade (Sistema de computação). 4. Qualidade de serviço. I. Título.

CDD 004.6

GABRIEL REIS CARRARA

Provendo Confiabilidade para Mecanismos de Consenso Através do Uso de Redes Definidas por Software

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Sistemas de Computação

Aprovada em Janeiro de 2020.

BANCA EXAMINADORA

Prof. Célio Vinícius Neves de Albuquerque, Ph.D. - Orientador, UFF

Prof. Diogo Menezes Ferrazani Mattos, D.Sc. - Orientador, UFF

Prof. Igor Monteiro Moraes, D.Sc., UFF

Prof. Miguel Elias Mitre Campista, D.Sc., UFRJ

Niterói

2020

Dedico este trabalho a Deus, aos meus pais, Edna e Wilton
e a todos que acreditaram em mim, familiares e amigos.

Agradecimentos

Gostaria de agradecer a Deus, por estar comigo em todos os momentos de minha vida, por ter me permitido chegar até aqui e por ter me abençoado com meus pais, Edna e Wilton, aos quais agradeço pelo amor incondicional e pelo apoio em todos os momentos de dificuldade.

Aos meus amigos e colegas pelo apoio e incentivo que me permitiram não desanimar ao longo dessa jornada. Agradeço aos meus professores da UFF e ao Laboratório MídiaCom e seus membros que contribuíram bastante para meu crescimento pessoal e profissional.

Agradeço ao meus orientadores, Célio Albuquerque e Diogo Mattos por sua dedicação e enorme paciência ao me orientar no desenvolvimento deste trabalho.

Por último, agradeço aos órgãos de fomento de pesquisa CNPq, CAPES, CGI/-FAPESP e FAPERJ, à empresa TAESA e ao programa de P&D ANEEL (PD-07130-0053/2018) pelo suporte financeiro recebido.

Resumo

A cadeia de blocos é uma tecnologia que permite a criação de históricos globais e imutáveis. Essa tecnologia se baseia em redes par-a-par para permitir a conexão de diversos usuários. Por se tratar de um sistema distribuído as aplicações de cadeia de blocos necessitam de protocolos de consenso capazes de replicar de maneira consistente seus dados para todos os participantes. Protocolos de consenso podem ser classificados em probabilísticos e determinísticos sendo respectivamente aplicados em plataformas de cadeia de blocos públicas e privadas. As aplicações privadas de cadeia de blocos simplificam o consenso da rede, pois executam mecanismos de consenso baseados em votação. O consenso baseado em votação é adequado a redes em que todos os nós são conhecidos, embora exija uma grande quantidade de mensagens trocadas, proporcional ao número de nós participantes. A principal contribuição deste trabalho é uma estratégia de aplicação de técnicas de Qualidade de Serviço (QoS) para garantir o funcionamento confiável dos mecanismos de consenso baseados em votação através da criação de filas. A estratégia proposta é baseada em Redes Definidas por Software (SDN) e tem como objetivo reduzir o tempo de terminação dos protocolos de consenso baseados em votação. A avaliação da proposta se baseia em um ambiente emulado, utilizando o Mininet, em que são executados os mecanismos de consenso Raft e BFT-SMaRt. Os resultados mostram que a proposta garante a terminação dos protocolos em tempo reduzido e a redução da carga de controle para a eleição dos líderes, quando comparados com o cenário sem reserva de recursos, já que garante uma qualidade mínima de serviço na taxa de transmissão para os fluxos de consenso da rede. A estratégia proposta aumenta até 100% das terminações de consenso no Raft e, para o BFT-SMaRt(*Byzantine Fault-Tolerant State Machine Replication*), assegura a operação mesmo em cenários de contenção, nos quais o consenso era anteriormente inviável. Os resultados também mostram uma redução de 80% nas mensagens trocadas para alcançar o consenso para o BFT-SMaRt.

Palavras-chave: Redes Definidas por Software, Consenso, Cadeia de Blocos, Qualidade de Serviço.

Abstract

The blockchain is a technology that allows the creation of global and immutable records. This technology is based on peer-to-peer networks to allow the connection of several users. Because it is a distributed system, blockchain applications require consensus protocols capable of consistently replicating their data to all participants. Consensus protocols can be classified into probabilistic and deterministic and applied to public and private blockchain platforms respectively. Private blockchain applications simplify network consensus by implementing voting-based consensus mechanisms. Voting-based consensus is suitable for networks where all nodes are known, although it requires a large amount of exchanged messages, proportional to the number of participating nodes. The main contribution of this work is a strategy for applying Quality of Service (QoS) techniques to ensure the reliable functioning of voting-based consensus mechanisms. The proposed strategy is based on Software Defined Networks (SDN) and aims to reduce the time of termination of voting-based consensus protocols through the creation of queues. The evaluation of the proposal is based on an emulated environment, using Mininet, where Raft and BFT-SMaRt consensus mechanisms are executed. The results show that the proposal guarantees the termination of the protocols in reduced time and the reduction of the control load for the election of leaders, in comparison with the no reservation scenario, since it guarantees a minimum quality of service in the transmission rate for the network consensus flows. The proposed strategy increases up to 100% of the consensus terminations in Raft and, for the BFT-SMaRt, ensures operation even in contention scenarios where consensus was previously not feasible. The results also show a 80% reduction in messages exchanged to reach consensus for the BFT-SMaRt.

Keywords: Software-defined Networking, Consensus, Blockchain, Quality of Service.

Lista de Figuras

Sumário

Capítulo 1

Introdução

A cadeia de blocos é uma tecnologia disruptiva que se baseia na natureza distribuída da rede par-a-par ao executar mecanismo de acordo, ou consenso para gerar um registro imutável, global e consistente de todas as transações. Aplicações comuns de cadeias de blocos, como a Bitcoin [34] e a Ethereum [46], são redes abertas ao público e amplamente utilizadas para transações financeiras. Outras aplicações de destaque, como AdEx[1] e Blockdaemon[2], executam em redes privadas, em que empresas criam e administram seus próprios ativos e redes para trocar informações ou realizar transações e, assim, interagir de maneira segura, controlada e com confiança distribuída [36]. Alguns exemplos de plataformas que permitem a criação de redes privadas são a Hyperledger Fabric[3] e a R3 Corda[4].

Segundo Gartner [2], a tecnologia da cadeia de blocos está passando por um momento em que empresas e centros de pesquisa estão investindo no desenvolvimento de soluções voltadas para cadeias permissionadas. Porém muitas dessas soluções irão falhar em cumprir seus requisitos, esse período é denominado de "Vale da Desilusão" ("*Trough of Disillusionment*"), Figura 1.1. Quando concluídas, essas soluções terão a finalidade de serem aplicadas no ambiente das redes privadas como bancos, centro de investimentos financeiros e na indústria de jogos eletrônicos.

Gartner [1] também afirma que as primeiras aplicações confiáveis de cadeia de blocos ainda estão de 5 a 10 anos de serem desenvolvidas. Muita pesquisa está em desenvolvimento por parte de instituições privadas que possuem interesses na tecnologia. Porém, ainda existem muitas contribuições e melhorias a serem realizadas antes que a tecnologia

[1]Disponível em https://www.adex.network/
[2]Disponível em https://blockdaemon.com/
[3]Disponível em http://www.hyperledger.org.
[4]Disponível em http://www.corda.net/.

da cadeia de blocos se estabeleça no mercado.

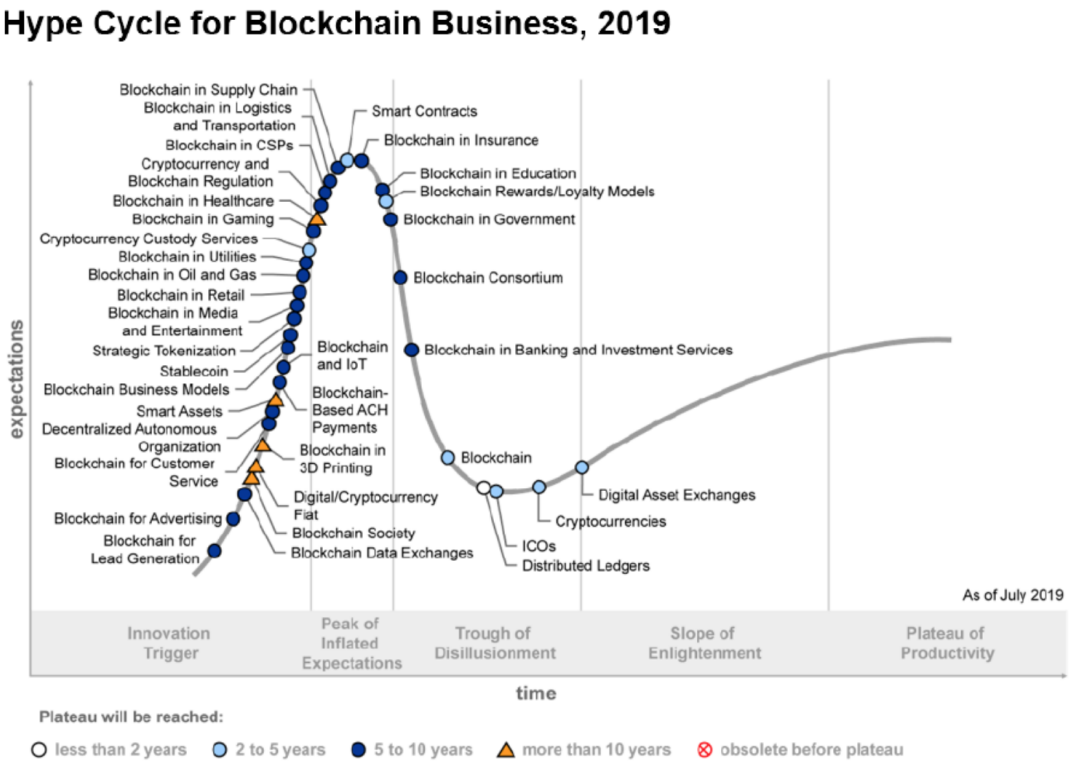


Figura 1.1: Ciclo de "*Hype*" da tecnologia de cadeia de blocos. Segundo Gartner as aplicações de cadeias de blocos ainda estão de 5 a 10 anos de causar o impacto esperado no mercado.

Devido à natureza distribuída, a cadeia de blocos necessita de mecanismos de consenso capazes de manter as informações atualizadas em todos os participantes da rede. Assim, um desafio chave no desenvolvimento de soluções de cadeia de blocos é projetar o mecanismo de consenso adequadamente. Como consequência de ser uma aplicação distribuída, qualquer mecanismo de consenso é restrito à limitações do Teorema *CAP*.

O Teorema *CAP* [9] (*Consistency, Availability and Partition Tolerance*) define a impossibilidade de um sistema distribuído oferecer as três propriedades: consistência, disponibilidade e tolerância de partição. Sendo assim, o sistema apenas é capaz de garantir no máximo duas dessas propriedades. Como sistemas distribuídos operam sobre redes de computadores passíveis de falhas, eles devem optar por garantir seu funcionamento em caso de partições. Dessa maneira os sistemas, em caso de partições, devem optar entre garantir disponibilidade ou consistência.

1.1 Motivação

Um problema encontrado ao se projetar um mecanismo de consenso é o aumento do número de participantes, assim como o aumento do alcance da rede que pode causar um aumento no atraso de encaminhamento das mensagens da rede prejudicando a terminação dos mecanismos de consenso. Em redes privadas, o uso de mecanismos de consenso baseados em votação as torna mais sensíveis a seu próprio crescimento, já que exige a troca de um número de mensagens elevado entre os participantes.

Os mecanismos de consenso podem ser classificados em baseados em competição e baseados em votação [35]. A primeira categoria é normalmente aplicada em redes públicas em que não se conhece o número total de participantes, ou o número de participantes é altamente variável, sendo impossível se obter confiança nos participantes. Os mecanismos baseados em votação são aplicados em ambientes de redes privadas e colaborativos. Nestes ambientes os participantes da rede são conhecidos e a rede é administrada pelas partes interessadas em sua utilização. Dessa maneira, os nós na rede têm sua identidade garantida e a entrada de novos participantes é controlada.

O uso de mecanismos de consenso baseados em votação impõe uso adicional da comunicação entre os nós da rede. Nesses mecanismos é necessária troca frequente de mensagens para a realização de tarefas como a eleição de um líder ou a atualização do estado da rede. A necessidade frequente de troca de mensagens aumenta a sensibilidade desses mecanismos de consenso ao estado de contenção da rede. Problemas como congestionamento ou gargalos em certos pontos da rede podem resultar na degradação do funcionamento desses mecanismos prejudicando sua terminação.

1.2 Contribuições

A principal contribuição deste trabalho é a proposta de uma estratégia de aplicação de técnicas de Qualidade de Serviço (QoS) para garantir o funcionamento de maneira confiável dos mecanismos de consenso baseados em votação. Essa estratégia faz uso de redes definidas por *software* (*Software Defined Networking*, SDN) [23] aplicadas como uma maneira eficiente e flexível para reduzir o tempo de terminação e garantir o funcionamento dos mecanismos de consenso através da criação de filas e fornecimento de banda mínimo. O paradigma de redes definidas por *software* desacopla o plano de dados distribuído do plano de controle logicamente centralizado. Nesse paradigma, a definição de políticas

restringe-se ao plano de controle e, posteriormente, são replicadas no plano de dados de acordo com os recursos disponíveis, como filas e limitação de banda [41].

Embora trabalhos anteriores utilizem as SDN para fornecer qualidade de serviço para diferentes aplicações, estes trabalhos não visam integrar aplicações SDN e plataformas de cadeias de blocos para melhorar o funcionamento do mecanismo de consenso [18, 24, 40]. A proposta foi avaliada no emulador Mininet [29], em que o ambiente de teste foi criado. O protótipo da proposta conta com o controlador Ryu SDN [38], que traduz as políticas de qualidade de serviço em regras de fluxo no plano de dados. A avaliação da proposta foi realizada em duas etapas. Primeiro, regras na forma de limites de banda máximo e garantia de banda mínima e a operação do controlador foram avaliadas e, em seguida, foram analisados diferentes aspectos dos protocolos, como número e custo das terminações em cenários com variados graus de congestionamento. O cenário de testes se baseia em um centro de dados onde a tecnologia de cadeia de blocos é utilizada para garantir auditoria distribuída e imutabilidade para informações. Nesse ambiente, múltiplas instituições compartilham recursos e o uso de cadeia de blocos elimina a necessidade de uma entidade confiável.

1.3 Estrutura do Texto

O trabalho está organizado da seguinte maneira. O Capítulo 2 elenca os trabalhos relacionados estudados durante o desenvolvimento deste trabalho. O Capítulo 3 apresenta o Teorema *CAP* e as propriedades necessárias para o desenvolvimento de mecanismos de consenso. O Capítulo 4 apresenta uma discussão dos principais mecanismos de consenso encontrados na literatura e como eles lidam com as limitações impostas pelo Teorema *CAP*. Em seguida o Capítulo 5 introduz a estratégia utilizada neste trabalho para prover confiabilidade a mecanismos de consenso. No Capitulo 6 são discutidos os resultados da avaliação experimental da proposta e, por fim, no Capítulo 7 é apresentada a conclusão e a discussão dos trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

Ao longo deste trabalho realizou-se um estudo sobre propostas de aplicações envolvendo redes definidas por *software* (*Software Defined Networking*, SDN) encontrados na literatura. As redes SDN representam um novo paradigma de rede onde o controle e o encaminhamento dos dados da rede é fisicamente separado em diferentes planos. Essa separação permite a flexibilização das soluções desenvolvidas para redes.

Parte das propostas levam em conta o uso de planos de dados distribuídos. Um desafio da aplicação de controladores distribuídos é manter a coordenação entre eles. Essa coordenação é essencial para garantir uma visão global do estado da rede. Algumas propostas implementam mecanismos de consenso ou cadeias de blocos para esse fim. O estudo de mecanismos de consenso e cadeia de blocos pode auxiliar no desenvolvimento das SDN, pois o entendimento dessas tecnologias auxilia a criação de novas soluções. O restante deste Capítulo apresenta a discussão dos trabalhos estudados.

2.1 Plano de Controle Distribuído

O DistBlockNet [42], proposto por Sharma *et al.*, consiste na utilização das redes SDN para permitir a criação de regras de controle flexíveis e facilitar o monitoramento das redes de dispositivos de Internet das coisas(*Internet of Things*, IoT). Para facilitar a instalação e disseminação de novas regras através de controladores distribuídos pela rede é utilizada uma cadeia de blocos onde as informações sobre regras de fluxos são mantidas atualizadas, dessa maneira, sempre que um participante da rede necessitar verificar a integridade ou buscar atualizações de regras elas estarão disponíveis na cadeia. Isso permite que essas regras sejam acessadas de maneira rápida e segura, aumentando a velocidade de tomada de decisão e reduzindo o custo de manutenção das regras da rede.

A proposta foi testada e comparada com uma rede SDN distribuída convencional. Dois tipos de teste foram realizados, o primeiro teste teve o intuito de avaliar o desempenho dos esquemas de atualização das tabelas de fluxo em redes de larga escala e os resultados mostram que o DistBlockNet consegue escalar de maneira linear com a demanda da rede enquanto a solução convencional não. O segundo teste verifica a capacidade de mitigar ataques de saturação feitos contra o plano de controle da rede e os resultados mostram que em cenários de alta demanda de novos fluxos o DistBlockNet foi capaz de identificar as requisições maliciosas e evitar assim a redução de vazão da rede. Testes subsequentes também foram realizados demonstrando a precisão na identificação de ameaças bem como a capacidade de preservar os recursos dos componentes do sistema por parte do DistBlockNet.

O Ravana [21] é uma plataforma que consiste em um módulo de controlador e um de comutador, esses módulos se comunicam em caso de eventos provendo ao controlador capacidades de replicação. A implementação do Ravana segue três princípios para prover as capacidades de replicação, o primeiro é ordenação dos eventos da rede através de um histórico distribuído entre as réplicas, o segundo princípio é a garantia de que os eventos da rede serão processados uma única vez pelo controlador e por fim o terceiro princípio garante que um controlador somente envia comandos referentes a um evento uma vez. O Ravana faz uso de um protocolo de replicação de duas fases onde o controlador ao receber um evento da rede primeiro garante que esse evento será enviado para as réplicas e depois garante que ao processar o evento os componentes da rede sejam devidamente notificados.

Para a validação da proposta foram feitos testes comparando o controlador Ryu padrão e uma versão com o Ravana sendo executado. As métrica avaliadas foram vazão e latência de processamento de eventos com diferentes quantidades de comutadores e os resultados mostram que o Ravana causou pouco impacto no processamento de eventos. Outros testes foram executados para avaliar o tempo de recuperação de falhas do controlador e os resultados mostram que o tempo médio de recuperação foi de 75ms. Esse resultado demonstra, que mesmo ocorrendo uma falha, o funcionamento da rede não é afetado por muito tempo. Por fim, foi avaliado a sobrecarga causada pelo mecanismo na rede e os resultados mostram que no pior cenário houve um aumento de 31% no uso da rede, os autores afirmam que essa quantia pode ser reduzida com a aplicação de otimizações no protótipo.

Para simplificar a utilização de controladores distribuídos apenas contando consistência eventual, Panda *et al.* propõem o SCL (*Simple Coordination Layer*). O SCL é um

mecanismo capaz de fornecer capacidade de coordenação e distribuição a controladores que não possuem essa função como POX, NOX e RYU.

Ele implementa uma camada de controle nos controladores e comutadores responsável por criar e organizar um histórico de eventos da rede, esse histórico é responsável por fornecer uma visão global dos recursos disponíveis na rede. O SCL também realiza periodicamente a sincronização entre os históricos de cada controlador garantindo a consistência eventual do estado da rede além de sondar os comutadores em busca de eventos não notificados. A principal vantagem em relação à consistência eventual vem do fato de que cada controlador é livre para agir em resposta a eventos da rede sem a necessidade de aguardar a confirmação dos demais controladores garantindo assim tempo de resposta reduzido.

A proposta foi avaliada através da comparação do SCL com o controlador ONOS. Os resultados demonstram a viabilidade de se obter um estado consistente da rede com tempo de resposta reduzido através da consistência eventual. Também foi observado que o SCL reduz o número de instâncias de controladores com visões divergentes sobre a rede. Outro experimento mostrou que o SCL foi capaz de encontrar o caminho mais curto na rede mais rapidamente em caso de falhas de enlaces. O impacto das sondagens feitas pelo controlador também foram avaliadas e os resultados mostram que essa abordagem possui pouco impacto no funcionamento da rede consumindo poucos recursos.

Mattos *et al.* [31] propõem um protocolo para o gerenciamento da atualização das políticas de controladores de rede distribuídos. Esse gerenciamento possui dois requisitos, primeiro os controladores devem decidir em que ordem as proposta de atualização das regras deve ser executada e segundo cada controlador deve verificar se a nova regra não causa nenhum conflito com as regras já existentes.

Para executar a atualização consistente das regras é necessário garantir duas propriedades, consistência e composição. Para a assegurar a consistência é importante que os controladores sejam capazes de agendar as atualizações e garantir que seja possível criar um agendamento global dessas atualizações. A composição das regras é a verificação das novas regras contra as antigas garantindo assim que haja conflito entre elas. Para a garantir tais propriedades foi proposta um protocolo de consistência. Esse protocolo executa três passos para garantir a consistência, primeiro o controlador que deseja propor uma nova regra envia uma propostas para os demais, em seguida os demais controladores votam nessa proposta caso ela não cause conflitos com uma regra existente e, por fim, se a proposta foi aceita, cada controlador realiza a instalação nos seus comutadores.

Os resultados experimentais mostram que o número de mensagens necessárias para o protocolo proposto alcançar o consenso é menor que o dos outros protocolos alcançando uma redução de até 66% no melhor caso. O experimento para observar o comportamento da rede quando alguns controladores foram realizados e os resultados mostram que mesmo com uma taxa de falha de 50% dos controladores ainda foi possível a realização do consenso.

Também foram realizadas simulações para verificar o desempenho do protocolo proposto em comparação com o protocolo "Two Phase Commit" e um protocolo idealizado. A comparação com protocolo ideal permitiu avaliar o quão próximo de um resultado ótimo a proposta se encontra. Os resultados dos testes mostram que o protocolo proposto possui desempenho mais próximo ao ideal do que o protocolo "Two Phase Commit". Testes mostraram que o protocolo teve desempenho superior a outros encontrados na literatura e que a sua implantação garante maior robustez da rede.

Pinping *et al.* propõem o WE-Bridge [30], um mecanismo capaz de fornecer capacidades de comunicação inter-domínio para controladores SDN. O WE-Brigde permite que controladores de diferentes domínios se conectem formando uma rede par-a-par para trocar informações sobre os recursos de seus domínios de maneira rica e granular. O mecanismo ainda permite a configuração de como os recurso dos domínios serão representados para os vizinhos, permitindo que informações restritas aos domínios não sejam expostas de maneira desnecessária. Com essa função é possível criar uma visão virtualizada da rede fornecendo apenas a informação das rotas desejadas pelos administradores ou apenas as informações de alcance daquela rede.

Para avaliar a proposta foi utilizada uma rede de testes contendo domínios de diferentes países, foram utilizada as redes CSTNET, Internet2, CERNET e Surfnet. A avaliação levou em conta dois casos de uso, primeiramente foram avaliadas as capacidades de roteamento por múltiplas rotas baseadas no endereço fonte onde foram configuradas diversas rotas através dos múltiplos domínios. Este experimento demonstrou a capacidade da proposta de encontrar múltiplas rotas através de domínios distintos. O segundo experimento teve como objetivo avaliar a capacidade de considerar os atributos das rotas como latência e largura de banda através de diferentes domínios para permitir a aplicação de regras de qualidade de serviço fim-a-fim. O objetivo do teste foi observar se WE-Bridge era capaz de encontrar a rota com maior largura de banda. Resultados mostra que a capacidade de negociação de rotas em diferentes domínios é possível de maneira granular e leve.

2.2 Gerenciamento do Plano de Dados

O OpenNetMon [43] é um módulo para o controlador Pox desenvolvido para permitir o monitoramento de maneira granular e eficiente de fluxos em uma rede SDN. Com o auxílio do OpenNetMon é possível se obter métricas sobre vazão, perda de pacotes e atraso da rede sem a necessidade de introduzir custos adicionais na rede. O OpenNetMon também ajusta sua frequência de verificação das métricas de acordo com a demanda da rede para assim não causar problemas em momentos de grande demanda e ainda assim fornecer informações úteis para aplicações de engenharia de tráfego.

Para a avaliação da proposta foram comparadas as métricas das medidas de vazão e perda de pacotes e atrasos obtidas pelo OpenNetMon, com a ferramenta Tcpstat e com medidas realizadas pela camada de aplicação dos sistemas finais. Os resultados mostram que o OpenNetMon foi capaz de fornecer informações com precisão similar ao esperado. O OpenNetMon ainda foi capaz de extrair tais métricas sem causar grande impacto no desempenho dos dispositivos da redes.

Sakic e Kellerer [39] propõem uma plataforma de avaliação estocástica que permite analisar analiticamente o impacto do uso mecanismo de consenso em controladores SDN distribuídos em diversos cenários de falhas. Outra contribuição feita é a criação de um monitor capaz de detectar o estado de degradação das aplicações dos controladores e agir de maneira a rejuvenescer a rede.

Para permitir a avaliação de maneira analítica os autores utilizaram uma redes de atividades estocásticas (denominadas SAN) para modelar os processos da rede como a comunicação através do RAFT, o processo de eleição do RAFT e seu funcionamento interno. Também foram modelados os processos de falhas dos componentes da rede, permitindo assim a injeção dessas falhas ou seu acontecimento de maneira aleatória. Essas modelagens são traduzidas para a forma de uma Cadeia de Markov de Tempo Contínuo (CTMC) permitindo assim que as características de uma dada configuração de rede predefinida sejam avaliadas de maneira analítica.

Foram realizados testes com diferentes configurações de redes geradas automaticamente e avaliados segundo sua precisão de cálculo e custo computacional. Os resultados das avaliações mostram que os modelos estocásticos propostas são capazes de representar os valores de atraso das redes de maneira bem próxima ao das redes reais e ainda preservarem custos computacionais aceitáveis.

2.3 Conclusão do Capítulo

Neste capítulo foram discutidas algumas propostas de trabalho envolvendo redes definidas por *software*. Ao fim do estudo pode-se observar que a maior parte das propostas envolvia a implementação de planos de controle distribuídos. A distribuição dos planos de controle permite aprimorar a flexibilidade e resiliência das SDN.

Capítulo 3

Consenso na Cadeia de Blocos

A cadeia de blocos é um histórico imutável de transações [48, 4]. Uma transação é a operação atômica e, em um ambiente de cadeia de blocos, ela pode se referir a uma troca de ativos, como no Bitcoin [34], ou uma execução de código, também chamada de contrato inteligente, como no Ethereum [33]. Transações são organizadas em blocos e armazenadas em uma estrutura de dados distribuída em uma rede par-a-par.

Os nós da rede armazenam uma réplica de todos os blocos. Cada nó executa protocolos de consenso para validar transações e agrupar transações nos blocos, que são encadeados usando uma referência ao bloco predecessor [36, 49]. A referência é um resumo criptográfico, obtido através de algoritmos criptográficos unidirecionais. A propriedade unidirecional das funções criptográficas do resumo criptográfico gerado assegura que a recuperação dos dados originais do resumo criptográfico gerado é improvável e, portanto, garante a integridade do conteúdo e a segurança da cadeia de blocos.

Antes de anexar um bloco, os nós processam as transações geradas pelo usuário. O processamento das transações se espalha em diferentes camadas. A Figura 3.1 mostra uma arquitetura em camadas com três camadas: transação, geração de blocos e distribuição. A camada de transação define a linguagem de codificação e os critérios usados na geração de transações e contratos inteligentes. Tanto as transações como os contratos inteligentes precisam ser assinados antes da sua disseminação para a base de transações. O processo de assinatura garante o não repúdio das transações e permite o controle de acesso e autenticação do conteúdo da transação. As assinaturas dependem do uso de um par de chaves assimétricas. Cada usuário é associado a um par de chaves ao entrar na rede. Cada transação emitida é assinada com a chave privada do usuário. A chave pública é responsável por endereçar o usuário na rede.

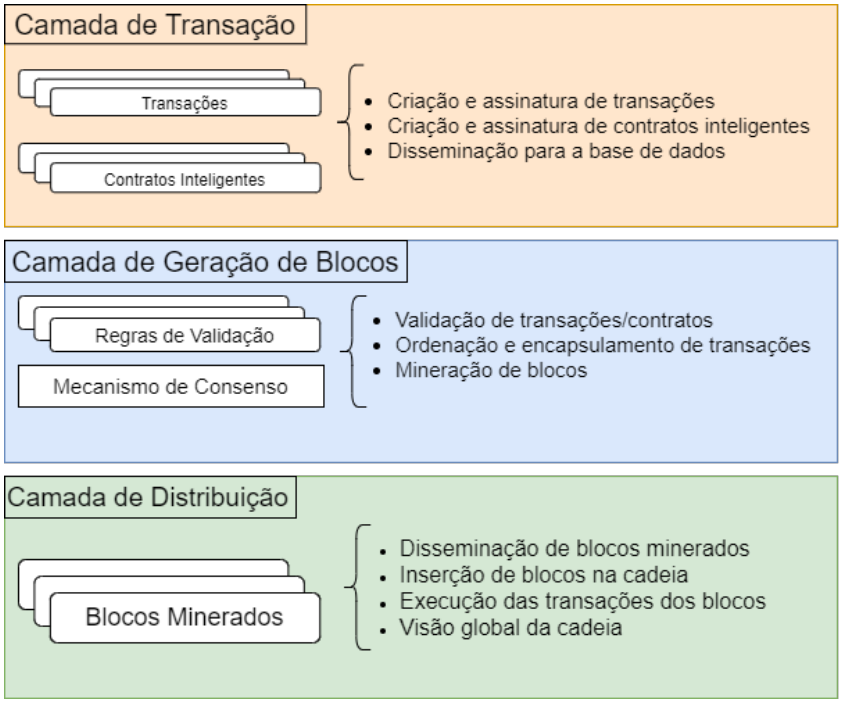


Figura 3.1: As funcionalidades primordiais da tecnologia da cadeia de blocos, separadas em camadas. O usuário gera transações na camada de transação. Na camada de geração de blocos, os pares validam as transações de acordo com as regras da rede, organizam-nas em blocos candidatos e mineram o bloco. A camada de distribuição dissemina os blocos minerados, que são inseridos na cadeia de blocos, ocasionando a execução das transações no bloco.

Os processos de validação de transações e de mineração de blocos residem na camada de geração de blocos. Todas as transações emitidas pelos usuários aguardam a execução em uma base de transações em aberto. Os nós mineradores, nós responsáveis pela validação de transações e criação de blocos, selecionam conjuntos de transações e as inserem em um bloco de candidatos. Antes de adicionar a transação ao bloco, a transação deve ser confrontada com as regras de validação da rede. Um nó minerador adiciona a transação ao bloco somente se ela for válida de acordo com as regras da rede. Caso contrário, a transação é descartada.

O minerador também organiza a ordem de inserção das transações no bloco. O conteúdo restante do bloco depende do mecanismo de consenso implantado na rede. O mecanismo de consenso também rege o processo de mineração. Quando um minerador consegue minerar um bloco, o bloco minerado deve ser disseminado por toda a rede. O processo de distribuição faz parte da camada de distribuição, assim como a inserção do bloco minerado na cadeia de blocos. Tal inserção só é bem sucedida se o resumo criptográfico do bloco minerado estiver correto. Caso contrário, o bloco minerado é descartado.

Assim que o novo bloco for incorporado à cadeia de blocos, as transações armazenadas no bloco são executadas. Após a execução, o estado global da cadeia de blocos muda, e a visão global da cadeia de blocos é atualizada. Se a visão global é a mesma para cada nó da rede, os nós chegaram a um consenso, ou seja, concordam com o estado atual da cadeia de blocos. A rede par-a-par subjacente, responsável pelo processamento das transações através das camadas, consiste em nós distribuídos agindo independentemente. Consequentemente, como qualquer sistema distribuído, o teorema do *CAP* (Consistência, Disponibilidade e Tolerância a Partição) influencia fortemente o seu funcionamento.

3.1 O Teorema *CAP* (Consistência, Disponibilidade e Tolerância a Partição)

O Teorema *CAP*, também conhecido como teorema de Brewer [9], identifica três características como críticas para qualquer sistema distribuído, a consistência, a disponibilidade e a tolerância a partição. Um sistema é dito consistente quando assegura que cada leitura seja a mesma em qualquer nó, ou seja, os nós têm uma visão global comum do estado do sistema. Se não houver tal garantia, o resultado pode retornar valores diferentes para leituras em nós diferentes, ou seja, uma inconsistência. A disponibilidade consiste em um sistema sempre dar uma resposta válida, que não seja uma mensagem de erro, às solicitações feitas a um nó que não esteja falhando. Por último, a tolerância a partições é quando um sistema continua a operar mesmo quando há mensagens perdidas ou atrasadas devido a problemas na rede. O sistema continua operando com qualquer número de falhas, desde que as falhas não resultem em uma falha de toda a rede.

3.2 Impossibilidade FLP

Fischer *et al.* provam a impossibilidade de resolver o consenso deterministicamente em um sistema assíncrono se ao menos um único processo pode falhar. Este resultado é frequentemente referenciado como impossibilidade de FLP, nomeado em homenagem aos seus autores, Fischer, Lynch e Paterson [16, 15]. A prova baseia-se em uma definição fraca do problema de consenso, em que a validade é relaxada, e a terminação só é necessária para um único processo. A incerteza em termos de pontualidade, como é um sistema supostamente assíncrono, somada à incerteza em termos de falhas, leva à conclusão de que nenhum algoritmo determinístico pode garantir a tomada de uma decisão. Há uma impossibilidade de um sistema assíncrono diferenciar um processo bloqueado de outro que

é meramente lento e que eventualmente terminará.

Por um lado, a impossibilidade de FLP indica quando não é possível chegar a um consenso. Por outro lado, Dolev *et al.* identificam cinco parâmetros que afetam a solução do problema de consenso: sincronização entre processos, sincronização de comunicação, ordenação de mensagens, comunicação em difusão ou entre pares, e atomicidade do envio e da recepção [14]. Portanto, para resolver o problema do consenso, o algoritmo deve contornar o resultado da impossibilidade do FLP. O consenso só pode ser alcançado se uma das três propriedades definidoras, validade, acordo e terminação, não for satisfeita. Como resultado, os algoritmos podem resolver o consenso se os processos seguirem fielmente o algoritmo ou, se os processos escaparem ao comportamento esperado, o algoritmo sempre satisfará as propriedades de segurança, mas pode sacrificar a vivacidade. A ideia por trás disso é que, em caso de falha, não há pressuposto de tempo adicional, e o algoritmo pode não terminar, mas a validade e as propriedades de acordo serão sempre aplicáveis.

3.3 Propriedades do Consenso

O teorema *CAP* (Consistência, Disponibilidade e Tolerância de Partição) está relacionado com a confiabilidade de um sistema distribuído e como ele se comporta em caso de falha [9]. Brewer postulou que é impossível garantir essas três propriedades para um sistema distribuído ao mesmo tempo e, então, o sistema só pode garantir duas delas no máximo. Portanto, se um sistema for construído fornecendo duas destas três prerrogativas, a terceira necessariamente não será cumprida.

Um programa de computador é definido como uma máquina de estados infinitos, e a sua execução é uma sequência infinita de estados, chamada histórico. Se a execução do programa termina, seu estado permanecerá infinitamente preso, e o último valor mostrado na execução do programa o definirá. Neste contexto, Lamport define informalmente o conceito de segurança como uma propriedade de um programa para evitar que "algo ruim" ocorra [26]. No mesmo sentido, a vivacidade é uma propriedade que funciona para fazer com que "algo bom" aconteça no programa [26]. O termo "algo ruim" corresponde, na prática, a falhas no software, tais como *deadlocks*. Por sua vez, "algo bom" refere-se às propriedades do software que contribuem para sua correta execução.

O problema de consenso é definido para um conjunto de n processos conhecidos, e, entre os n processos, pode haver um número máximo f de processos defeituosos. As falhas de processos podem ser definidas como *crash-failure*, onde um processo deixa de responder

ou responde em tempo infinito, ou falha bizantina, no qual os processos se comportam de forma inesperada e em desacordo com o protocolo estabelecido [12].

Mecanismos de consenso são algoritmos que alcançam acordo sobre um único dado ou sobre o estado de um sistema distribuído. Lamport [27] define duas propriedades fundamentais para o projeto de mecanismos de consenso, segurança (*Safety*) e vivacidade (*Liviness*). A propriedade de segurança, define que o consenso é responsável por garantir que um sistema distribuído escolha apenas um único valor proposto, e um processo correto somente aprende os valores propostos. Por sua vez, a propriedade da vivacidade pode subdividir os mecanismos de consenso em dois grandes grupos. Os mecanismos de consenso determinístico asseguram o acordo sobre uma transação após a sua aplicação. Os mecanismos de consenso probabilístico criam formas de assegurar que o acordo tende a acontecer, porém sua convergência não é assegurada.

O problema do consenso é formalmente definido em termos de três propriedades, **validade**, **acordo** e **terminação**. A validade assegura que se todos os processos corretos propuserem o mesmo valor v, qualquer processo correto converge para v. A propriedade do acordo define que não há dois processos corretos que decidem de forma diferente. A terminação garante que cada processo correto eventualmente decida. As propriedades de validade e acordo são responsáveis por garantir a segurança de mecanismos de consenso, de acordo com a definição de Lamport, enquanto a terminação é uma propriedade que garante a vivacidade.

Os mecanismos de consenso podem ser binários, multi-valorados, vetoriais ou baseados em Paxos [12]. Um algoritmo de consenso binário visa alcançar consenso sobre um valor binário, verdadeiro ou falso. Cada processo propõe um valor binário inicial e decide sobre um valor de v. O consenso multi-valorado é semelhante ao consenso binário, mas permite que os nós decidam sobre um conjunto de valores de tamanho arbitrário. No entanto, a definição de validade neste tipo de consenso introduz a possibilidade de decidir sobre um valor nulo. O consenso vetorial propõe uma propriedade de validade diferente em relação às versões anteriores do problema. A decisão não é mais um valor único, mas um vetor com alguns dos valores iniciais dos processos e, pelo menos $f + 1$, a maioria, são processos corretos.

Um tipo ligeiramente diferente de definição é a utilizada nos algoritmos Paxos. A ideia é que os processos desempenhem um ou mais dos seguintes papéis: proponentes que propõem valores; aceitadores que escolhem o valor a ser decidido; e aprendizes que aprendem o valor escolhido. A segurança do consenso é definida em termos de que apenas

um valor proposto é escolhido; apenas um único valor é escolhido; um aprendiz correto aprende apenas um valor proposto. A vivacidade é definida em termos de que algum valor proposto seja eventualmente escolhido e, após a escolha desse valor, os aprendizes corretos aprendem-no. Embora as definições conduzam a modelos práticos de sistema, a ocorrência de falhas bizantinas dificulta a implementação de mecanismos de consenso, uma vez que os nós se desviam do comportamento esperado das formas mais diversas possíveis [31].

3.4 Conclusão do Capítulo

Os conceitos apresentados anteriormente permitem classificar as propriedades específicas de um mecanismo de consenso, facilitando assim a sua compreensão, pois a combinação destas características pode caracterizar cada programa. A distinção entre vivacidade e segurança torna-se relevante na avaliação do modo de operação da cadeia de blocos, especialmente no estudo dos mecanismos de consenso. A existência de artifícios que impedem a existência de duas versões do mesmo bloco na cadeia de blocos, ou a presença de dois blocos simultâneos diferentes, está relacionada à propriedade de segurança. Um exemplo da aplicação da vivacidade no contexto da cadeia de blocos é o consenso probabilístico, que garante que em algum momento, as rodadas de consenso terminam, e os nós eventualmente chegam a consenso.

Para os mecanismos de consenso, vivacidade e segurança têm uma correlação direta com o Teorema *CAP*. A propriedade de vivacidade de um mecanismo de consenso garante que ele sempre termina suas rodadas de consenso. Mesmo que não tenha sido capaz de alcançar o consenso, o mecanismo de consenso não deve esperar indefinidamente, garantindo a sua disponibilidade. A propriedade de segurança de um mecanismo de consenso garante que seus participantes ativos estejam no mesmo estado após uma rodada de consenso, garantindo assim sua consistência. A segurança também garante que a rede continue operando mesmo na presença de atrasos ou perda de pacotes, garantindo assim sua tolerância a partições.

Capítulo 4

Mecanismos de consenso para Cadeias de Blocos

As plataformas de cadeia de blocos, assim como qualquer sistema distribuído, estão restritas às limitações impostas pelo Teorema CAP e pela impossibilidade FLP. Ao se projetar um mecanismos de consenso deve-se considerar maneiras de contornar essas limitações para garantir que o mecanismo projetado cumpra a função desejada.

Este Capítulo elenca os mecanismos de consenso encontrados na literatura atualmente. Para cada mecanismo é avaliada a maneira como seu projeto contorna as limitações mencionadas.

4.1 Mecanismos de Consenso Probabilísticos

A impossibilidade FLP define que não é possível obter o consenso em uma rede assíncrona se ao menos um processo falhar. No entanto, o modelo de consenso apresentado em sua prova pode ser modificada para permitir que o consenso seja alcançado ao custo de alguma de suas propriedades fundamentais. Com isso, diversos mecanismos de consenso apresentam técnicas para garantir consenso.

Algumas técnicas para contornar a limitação FLP são técnicas que sacrificam o determinismo, levando a algoritmos probabilísticos; técnicas que adicionam sincronização ao modelo do problema; técnicas que usam hibridização; técnicas que enriquecem a definição do problema [12]. Portanto, mecanismos determinísticos de consenso criam modelos de sincronização, muitas vezes baseados em protocolos de rede e transporte.

Os mecanismos de consenso probabilísticos para a cadeia de blocos, como os baseados

em provas de trabalho e autoridade, sacrificam o acordo. Desta forma, eles trocam a sua convergência determinística pela convergência probabilística. Assim, uma rodada de mecanismos de consenso probabilístico pode alcançar um consenso com uma certa probabilidade. Considerando o Teorema da CAP, plataformas de cadeias de blocos baseadas em um mecanismo de consenso probabilístico normalmente sacrificam a consistência em favor da tolerância a partições e da disponibilidade [36].

4.1.1 Prova de Trabalho

O mecanismo de consenso da Prova de Trabalho (*Proof-of-Worf*, PoW) [34] é utilizado no contexto da cadeia de blocos públicas, ou seja, cadeias sem controle de acesso. Em tal cenário, o número de nós é geralmente grande e cada nó é anônimo. Não há confiança entre os participantes da cadeia de blocos, o que dificulta o funcionamento do mecanismo de consenso. Além disso, ainda existe a possibilidade de nós maliciosos e também ataques como o Sybil, nos quais um usuário pode ter duas ou mais identidades, o que pode influenciar ou mesmo manipular o processo de construção de consenso [6].

O funcionamento da Prova de Trabalho é baseado na resolução de um desafio criptográfico. Os nós responsáveis por resolver este desafio são chamados mineradores, e o processo de resolução de problemas é chamado de mineração. É basicamente um método de tentativa e erro ou força bruta. A mineração consiste em encontrar um valor numérico particular, *nonce* criptográfico, a ser inserido no final de um bloco candidato para ser incorporado à cadeia de blocos. O desafio criptográfico requer que o valor gerado pela função resumo criptográfico do bloco tenha um certo número pré-definido de zeros no seu início. Uma vez resolvido o problema, o valor encontrado passa a fazer parte do conteúdo do bloco candidato, que depois é divulgado por toda a rede para ser validado pelos pares. Após mais da metade dos pares validarem o bloco, ele é inserido na cadeia de blocos. A fim de incentivar a mineração, uma recompensa é oferecida ao minerador, ou grupo de mineradores, que encontrar primeiro a resposta ao desafio.

O processo de validação envolve o cálculo do resumo criptográfico do bloco e a verificação de que ele tem o número necessário de zeros no início. Por um lado, é essencial que o processo de validação seja relativamente rápido para melhorar o desempenho da cadeia de bloqueio, e também para evitar possíveis ataques DDoS. Em tais ataques, numerosos nós maliciosos poderiam falsamente afirmar ter resolvido o desafio criptográfico, e com um processo de validação lento, causar congestionamento ou mesmo uma queda de serviço em certos nós da rede da cadeia de blocos. Por outro lado, o tempo para a

mineração de blocos não pode ser muito curto para evitar que ramificações apareçam com mais frequência na cadeia de blocos. Quando ramificações surgem, a cadeia de blocos deve chegar a um consenso sobre qual caminho seguir e qual caminho descartar, o que requer um certo período de tempo. Se ramificações tiverem um tempo mais curto para emergir do que o seu tempo de resolução, o sistema seria inviável. A PoW pode contornar este problema ajustando o nível de dificuldade do desafio criptográfico. A probabilidade de dois mineradores serem capazes de resolver o desafio criptográfico e disseminar suas diferentes versões do bloco a ser inserido na cadeia de blocos aumenta à medida que o tempo para resolver o problema diminui. Na PoW, o tempo médio levado para minerar um bloco pode ser ajustado de acordo com o número de zeros necessários no início do resumo criptográfico do bloco: quanto maior o número de zeros, mais difícil será o problema e, consequentemente, mais tempo será necessário para resolve-lo.

A Prova de Trabalho é computacionalmente cara e leva tempo para iniciar a execução das transações. Este é o custo para agir em cadeia de blocos não permissionadas. Tais inconvenientes são principalmente devidos à necessidade de pelo menos metade da rede validar o bloco minerado e ao tempo necessário para resolver o desafio criptográfico, que não pode ser curto para evitar ramificações. Além disso, os custos de energia associados à mineração são altos porque a crescente dificuldade do desafio requer cada vez mais potência computacional. Na prática, o que acontece é que a maioria dos blocos são minerados por conglomerados mineradores, chamados *pools*, nos quais vários nós trabalham em associação, compartilhando recursos para resolver o desafio. A formação de *pools* vai contra a premissa fundamental da cadeia de blocos de descentralização.

4.1.2 Prova de Participação

A Prova de Participação (*Proof-of-Stake*, PoS) [45, 13, 36] foi criada para superar as dificuldades de eficiência que a Prova de Trabalho (PoW) enfrenta. A ideia por trás da PoW é que um minerador líder emerge através da competição e é responsável pelo próximo bloco a ser inserido na cadeia de blocos. Os inconvenientes aparecem com a escalabilidade da rede, mais pares tentando resolver o desafio implica um aumento na dificuldade do problema e, como o tempo médio para minerar um bloco deve permanecer aproximadamente constante, o aumento no uso de energia para minerar um bloco é inevitável. Nas plataformas de cadeia de blocos baseadas em PoW, o crescimento da rede é diretamente proporcional ao aumento do uso de energia para se chegar a um consenso.

Ao invés de eleger o líder minerador definido como uma corrida computacional, o

mecanismo de Prova de Participação sugere que o líder minerador deve ser escolhido aleatoriamente, mas ao invés de recursos computacionais, a eleição deve levar em conta a quantidade de participação que cada minerador tem de acordo com o livro razão atual da cadeia de blocos. A participação é o número de moedas que um utilizador detém para participar no processo de consenso. Esta mudança de paradigma tornou a PoS consideravelmente mais eficiente em termos de consumo de energia que a sua predecessora.

A aplicação mais conhecida que usa PoS é a criptomoeda Ethereum. Nessa aplicação, os nós dispostos a participar do consenso, os validadores, depositam uma certa quantidade de moedas na rede. Em seguida, um algoritmo seleciona o validador responsável por forjar o novo bloco, de forma parcialmente aleatória. A quantidade de moedas que cada validador deposita é a participação do validador. A participação de cada validador é proporcional à chance que ele tem de ser selecionado para organizar as transações para forjar um bloco. A aleatoriedade impede que os nós mais ricos monopolizem o processo de consenso. Para incentivar um validador a se comportar bem, a rede considera sua participação compartilhada como um depósito de segurança, ou seja, se o validador se comportar de forma maliciosa incluindo um viés no processo de validação, sua participação é retida. Em contraste, se o validador selecionado para forjar o bloco se comportar como esperado, ele é recompensado com moedas.

4.1.3 Prova de Participação Delegada

A Prova de Participação Delegada (*Delegated Proof-of-Stake*, DPoS) leva adiante os aumentos de velocidade e escalabilidade que o consenso tradicional que PoS proporciona. Os objetivos da DPoS são melhorar o desempenho, diminuindo o tempo de transação e de geração de blocos, e a flexibilidade, sem comprometer a estrutura descentralizada. Algumas das plataformas de cadeia de blocos que utilizam um mecanismo de consenso baseado em DPoS são Bitshares[1], Ark[2], EOS[3], Lisk[4], e Steem[5].

A DPoS é baseado na aprovação da votação, em que cada eleitor pode selecionar qualquer número de candidatos, e o vencedor é o mais aprovado (votado). Diferentemente do PoS, que seleciona pseudo aleatoriamente um nó para ser responsável pela geração de um novo bloco e pela verificação das transações, as partes interessadas de um sistema DPoS

[1]Disponível em https://bitshares.org/
[2]Disponível em https://ark.io/
[3]Disponível em https://eos.io/
[4]Disponível em https://lisk.io/
[5]Disponível em https://steem.com/

concordam em um certo número de nós, denominados testemunhas, que irão realizar essas tarefas. Assim, os únicos nós responsáveis pela validação das transações são as testemunhas eleitas pelas partes interessadas. O número de testemunhas é definido de forma que pelo menos 50% dos participantes votantes concorrem no nível de descentralização, e uma vez concluída essa tarefa, a votação é realizada. Cada voto tem um peso que é proporcional à quantidade de participação do eleitor. Portanto, um usuário não precisa ter uma quantidade significativa de participação para ter grandes chances de ser o gerador do bloco. Reduzir o número de validações feitas por bloco inserido, sem perder o fator de segurança que envolve através da votação de aprovação, é o elemento chave por trás do aumento substancial do desempenho do DPoS quando comparado ao PoS.

O DPoS também proporciona flexibilidade à rede. As regras e parâmetros, tais como tamanho da transação e programação de taxas, podem ser ajustados democraticamente mesmo após a criação do bloco de gênese. Isto pode ser feito através de comitês, que são compostos por delegados eleitos, responsáveis pela manutenção e gestão da rede, o que inclui a organização do processo de seleção das testemunhas. Ainda que o DPoS melhore o desempenho e a flexibilidade, a utilização de um sistema eleitoral ponderado baseado na participação do eleitor pode levar a problemas. Por exemplo, os usuários com menos ativos têm uma pequena influência sobre o processo de votação. Outra desvantagem da DPoS é garantir que todos os delegados estejam bem informados para selecionar as testemunhas adequadas e proporcionar suficiente descentralização.

4.1.4 Prova de Tempo Decorrido

O objetivo da Prova de Tempo Decorrido (*Proof-of-Elapsed-Time*, PoET) é substituir o elevado desperdício de recursos imposto pela PoW através da inclusão de um tempo de espera específico gerado por hardware confiável [11]. O hardware proposto para esta tarefa é o *Intel Software Guard Extension* (SGX), disponível em muitos processadores Intel modernos. A Hyperledger Sawtooth Lake[6], é uma plataforma de cadeia de blocos que implementa o PoET.

Na PoET, em cada rodada, cada nó realiza uma etapa de espera cujo valor é gerado pelo SGX. Cada nó convoca um enclave dentro do SGX para gerar um atraso aleatório. Após esperar o tempo necessário, um nó pode se declarar líder em uma rodada de consenso e gerar um novo bloco para a cadeia de blocos. O módulo também cria um certificado que pode ser usado por qualquer nó para verificar se o líder esperou corretamente pelo

[6]Disponível em https://www.hyperledger.org/projects/sawtooth

tempo aleatório apropriado. Assumindo que um atacante não pode subverter o módulo de hardware, a PoET fornece um mecanismo de consenso com as mesmas propriedades do PoW, mas sem o desperdício de recursos causado pela mineração. Entretanto, o investimento econômico ainda influencia o protocolo, porque a probabilidade de um nó se tornar o líder é proporcional ao número de módulos de hardware sob seu controle.

Embora apresente as mesmas propriedades do PoW, os resultados experimentais mostram que o PoET proporciona uma menor tolerância a ataques [11]. Se um atacante pode subverter pelo menos $\Theta(\frac{\log \log n}{logn})$ de nós em que n é o número de nós na rede, ele pode simular o comportamento do nó honesto mais rápido sem ser detectado. Os limites do teorema CAP impostos ao PoET são os mesmos que no PoW.

4.1.5 *Ripple Protocol Consensus Algorithm*

Ripple é um projeto de código aberto projetado para funcionar como uma criptomoeda e como uma rede de pagamento para transações financeiras. Ambas as aplicações têm subjacente o Algoritmo de Consenso do Protocolo Ripple, que pode suportar $(n-1)/5$ falhas em que n é o número de servidores na rede. As principais vantagens do Protocolo Ripple são as melhorias fornecidas na utilidade, também referidas como *"usefulness"*. Tais melhorias estão relacionadas à conveniência e à latência do sistema.

O Mecanismo de Consenso do Ripple tem sete componentes principais, sendo alguns deles definições próprias de componentes comuns para cadeias de blocos. O termo livro-razão, que normalmente define a estrutura da cadeia de blocos, é utilizado para definir a estrutura de um bloco. A descrição de cada componente é a seguinte:

- *Servidor*: Todas as entidades que participam ativamente no processo de consenso são chamadas de servidores. Elas executam o programa de Servidor do Ripple;
- *Livro-razão*: Este é o bloco contendo um registro das transações válidas que passaram pelo processo de consenso. No Ripple o consenso é realizado sobre cada transação individualmente. Cada transação aprovada e incluída no livro razão. O livro razão também é atualizado com a quantidade de cripto-moeda que cada usuário possui;
- *Livro-razão em Aberto*: Novas transações propostas pelos usuários são armazenadas em blocos candidatos. Assim, os blocos candidatos têm transações que ainda não foram validadas pelo mecanismo de consenso. Cada nó tem a sua própria versão de

blocos candidatos;

- *Último Livro-razão Fechado*: Uma vez que um bloco candidato passa pelo consenso, torna-se o último livro-razão fechado. Portanto, este é o mais recente bloco possuído por todos os nós da rede.

- *Lista Única de Nós*: A Lista Única de Nós (UNL) é um registro mantido por um servidor que contém uma lista de outros servidores confiáveis que podem não conspirar contra transações justas propostas. Somente membros do UNL do servidor que propôs a transação podem ter votos válidos no processo de consenso. É importante notar que não é necessário que todos os membros da rede tenham um comportamento imparcial, mas a UNL como um todo deve ter.

- *Proponente*: Os proponentes são servidores que transmitem transações para serem incluídas no processo de consenso. O consenso é dividido em rodadas. Em cada rodada, qualquer servidor pode se tornar um proponente, tentando incluir transações no livro-razão.

Cada rodada do consenso tem as mesmas fases. Em primeiro lugar, cada servidor reúne todas as transações armazenadas em um bloco candidato, formando uma lista. A lista é o conjunto de transações candidatas a serem validados na fase seguinte pelos membros da UNL. A segunda fase é a votação pelos membros da UNL. Com base nos votos acumulados, cada nó refina seu conjunto de candidatos, e as transações que recebem o maior número de votos são passadas para a próxima rodada [5]. As transações que recebem 80% ou mais dos votos concordando com a sua validade são então aplicadas ao bloco. Caso contrário, as transações são descartadas ou incluídas na lista de candidatos para a próxima rodada de consenso.

4.2 Mecanismos de Consenso Determinísticos

Sistemas de computador confiáveis devem lidar com componentes maliciosos ou com problemas de funcionamento, o que causa informações conflitantes em diferentes partes do sistema. Lamport define este problema como o Problema dos Generais Bizantinos [28]. O problema é apresentado como uma analogia, na qual várias divisões do exército bizantino, cada uma comandada pelo seu general, fazem um cerco a uma cidade inimiga e planejam um ataque. Devido à distância, os generais só podem fazer contato entre si através de um mensageiro e assim elaborar um plano de invasão. No entanto, alguns generais podem

ser traidores e tentar boicotar o acordo. Portanto, para que a missão tenha sucesso, deve haver um algoritmo que garanta o seguinte. (i) Todos os generais leais concordam com a mesma estratégia de invasão. Generais comprometidos com a invasão da cidade devem concordar sobre o melhor plano a seguir e, após a decisão, devem segui-lo à risca, independentemente dos traidores. Assim, o algoritmo acordado pelos generais deve ser bem sucedido, não importa o que os traidores façam. (ii) Um pequeno número de traidores não deve influenciar os generais leais ao império bizantino a adotar um mau plano. Esta condição é um pouco mais difícil de descrever já que o termo mau plano pode não ser fácil de identificar dependendo do contexto. Se as alternativas ao problema mencionado são apenas atacar ou recuar, o acordo explicado em (i) pode ser feito através de uma votação, baseada em uma maioria simples. Entretanto, se houver um número quase igual de generais traidores e generais leais, é impossível definir qual das opções pode ser chamada de "ruim".

Na analogia apresentada, os generais representam os computadores em uma rede, e a invasão da cidade representa uma tarefa que deve ser realizada em conjunto por esses computadores. As plataformas de cadeia de blocos devem ser capazes de lidar com este tipo de problema, e é papel dos mecanismos de consenso encontrar a solução mais adequada para cada ambiente em que a plataforma atua. A formulação do problema é fundamental para o estudo dos mecanismos de consenso em sistemas distribuídos, pois estabelece um limite teórico mínimo do número de nós bem comportados que devem existir em uma rede com um número fixo de nós maliciosos, sem comprometer o seu correto funcionamento. Kwon [25] mostra que não é possível ter um sistema tolerante a este tipo de falhas, conhecidas como falhas bizantinas, se o número total de generais for inferior a $3f + 1$, onde f indica o número de generais traidores.

As plataformas da cadeia de blocos atuais baseadas em mecanismos determinísticos de consenso tolerante a falhas bizantinas tendem a sacrificar a disponibilidade em favor de uma forte consistência [7, 10, 44]. Os mecanismos de consenso determinísticos que não toleram falhas bizantinas tendem a adotar um modelo de consistência frouxa [37, 31].

Propostas recentes de consenso em cadeia de blocos introduzem as ideias de (i) executar o consenso BFT em uma janela deslizante entre os nós [19]; e (ii) adicionar um caminho rápido e limiares de assinaturas em grupo para reduzir a carga de rede dos protocolos BFT [22, 17]. Estas propostas sacrificam parcialmente a tolerância à partição das plataformas e visam garantir algum grau de sincronização. A sincronização é utilizada para contornar as limitações da impossibilidade FLP. Nesse caso, é possível garantir que

o consenso será alcançado pelo mecanismo caso os nós não se desviem do protocolo.

4.2.1 Paxos

O nome Paxos refere-se a um sistema de consenso fictício inspirado na ilha grega de Paxos. É um mecanismo tolerante a falhas, já que suporta menos da metade dos nós da rede em falha, mas assume a inexistência de conluio ou mensagens corrompidas que tentam subverter o protocolo. Portanto, ele não suporta falhas bizantinas [27, 8].

Paxos é composto por cinco tipos de nós com diferentes papéis no protocolo. (i) *Clientes* são responsáveis por escrever pedidos para o sistema distribuído. (ii) *Aceitadores* agem como a memória do sistema, garantindo a tolerância a falhas. Eles formam grupos chamados *Quorum*. Qualquer mensagem que um membro do *Quorum* aceite receber deve ser recebida por todos os outros membros do *Quorum* também, ou então a mensagem deve ser descartada. O mesmo se aplica às mensagens recebidas pelos membros do *Quorum*; eles devem ignorá-la a menos que todos eles recebam a mesma mensagem. (iii) *Proponentes* agem ativamente para buscar consenso. Estes nós pressionarão os aceitantes a concordar com um pedido e tentarão resolver quaisquer conflitos quando eles ocorrerem, atuando como coordenadores. (iv) Os *Aprendizes* são responsáveis por responder aos pedidos dos clientes. Uma vez que todos os aceitantes validem uma solicitação, os aprendedores a executarão e enviarão de volta os resultados aos clientes. (v) *Líderes* são um tipo especial de proponente, que tentam garantir a propriedade de segurança do sistema. O correto funcionamento do Paxos requer a presença de um líder. Se mais de um proponente atuar como líder, o protocolo poderá ter atualizações conflitantes e, portanto, só garantirá o progresso se somente um líder estiver presente.

O consenso acontece em duas fases. A primeira fase consiste na preparação do pedido, e a segunda fase está relacionada com a sua aceitação. Na fase um, um proponente envia ao *Quorum* uma mensagem de preparação com um identificador. Em seguida, cada aceitador verifica o número do identificador. Se for maior que o último valor, ele retorna uma mensagem de promessa ao proponente com o valor do identificador da última solicitação comprometida. Se o identificador tiver valor menor na versão original do Paxos, o aceitador não enviará nenhuma mensagem, não validando a solicitação. Quando um proponente recebe uma mensagem de promessa de todos os membros do *Quorum*, a segunda fase começa. O proponente prepara e envia uma mensagem de aceitação, com um número maior que o maior identificador da última solicitação comprometida enviada pelos aceitadores, forçando-os a concordar com o compromisso. Assim, o *Quorum* valida

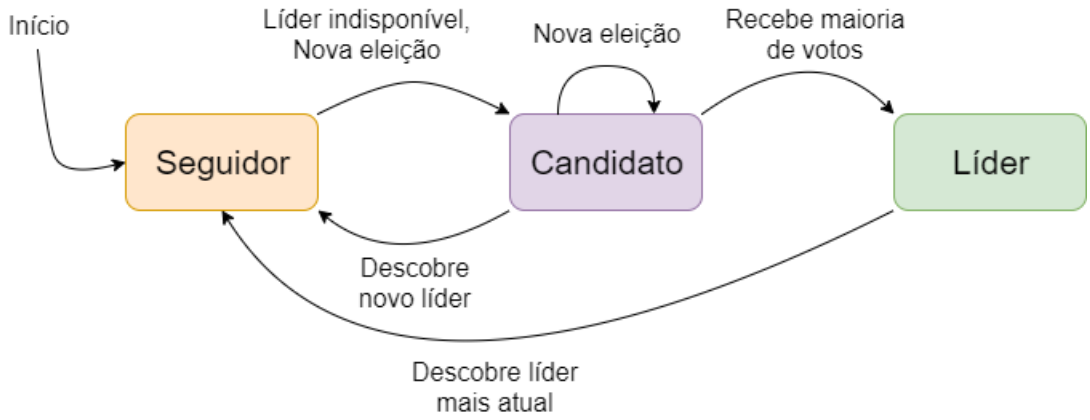


Figura 4.1: Máquina de estados do mecanismo Raft. Se um seguidor não receber nenhuma comunicação, torna-se um candidato e inicia uma nova eleição. Um candidato que recebe votos da maioria do grupo completo torna-se o novo líder. Os líderes normalmente operam até falharem.

a mensagem e responde uma mensagem aceita para o aceitador.

4.2.2 *Raft*

O *Raft* é um mecanismo de consenso desenvolvido como uma alternativa mais simples de estudo que o Paxos. Sua implementação é baseada no modelo líder-seguidor que suporta a perda dos dois tipos nós, sendo portanto, considerado um mecanismo tolerante a falhas [37, 47].

O consenso no *Raft* é alcançado de uma forma relativamente simples, tem um desempenho semelhante e a mesma tolerância a falhas que o Paxos. A principal diferença entre estes dois mecanismos é a forma como subdividem o problema do consenso e como abordam cada tarefa. No *Raft*, o consenso é dividido em três etapas: eleição do líder, a replicação do histórico e a segurança. A máquina de replicação de estado do *Raft* pode assumir três estados: candidato, seguidor e líder. Todos os nós devem estar, necessariamente, em um desses estados. O estado inicial padrão de um nó é como um seguidor. Se nenhum líder for notado em um determinado período, uma eleição ocorre, e os seguidores atualizam seu estado para candidato. Nesse momento, a votação começa entre os candidatos, e através da Chamada de Procedimento Remoto *RequestVotes*, eles começam a reunir votos durante a eleição. Uma vez alcançada a maioria simples, a eleição termina, e um nó candidato é promovido a líder. Se houver uma falha no nó líder, outro nó é eleito. O líder utiliza um *RPC* de *AppendEntries* com duas intenções principais:

- *Replicação de entradas do Histórico*: Os registos das transações são armazenadas

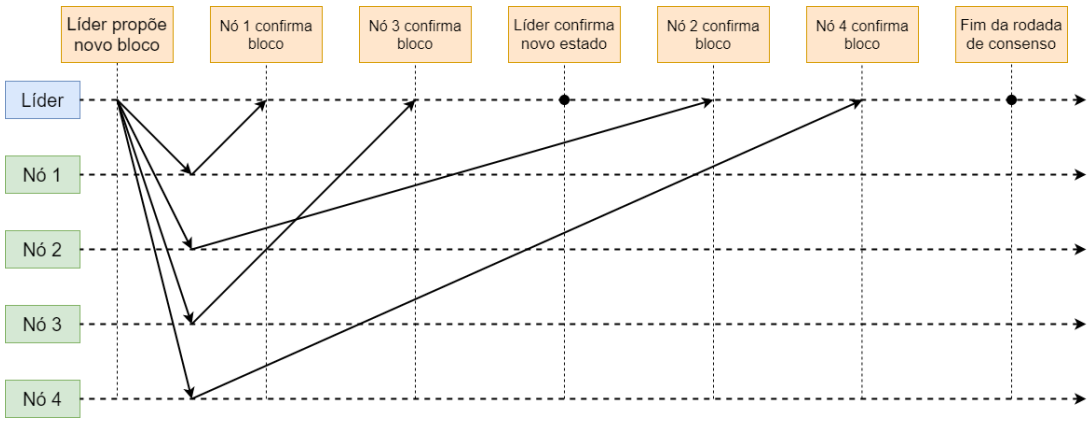


Figura 4.2: Diagrama de sequência representando a troca de mensagens necessária para realização do consenso no mecanismo Raft. No Raft as requisições são enviadas para o líder que, após validar, envia a proposta para os demais participantes. São necessários $\frac{n}{2}+1$ confirmações para validar um novo bloco.

em cada líder, sob a forma de filas, e indexadas por um número inteiro. O líder então decide quando é seguro inserir as transações no livro razão e então começa a enviar o comando de inserção para todos os seus seguidores até que todos eles tenham o mesmo estado. A replicação do registro é feita de forma a forçar outros registros a concordarem com ele. Após esta tarefa estar concluída, é enviada uma resposta para o cliente que solicitou o registro com os resultados.

- *Validação do estado de Seguidor*: Funciona como um batimento cardíaco, ao enviar um pedido, se o líder não receber resposta, o nó de destino é considerado em falha.

Na presença de um líder ativo os seguidores podem enviar as requisições de alteração de estado da rede que são armazenados pelo líder. Quando um certo número de requisições é atingido o líder decide quais e em qual ordem elas serão executadas. Após a decisão o líder envia aos seguidores essa informação. Cada seguidor por sua vez executa as operações segundo a ordem definida atualizando assim sua visão do estado da rede, a Figura 4.2 apresenta as mensagens trocadas durante uma rodada de consenso. Para que todos os seguidores possuam a mesma visão da rede ao final desta operação é necessário que apenas operações deterministas sejam executadas.

O mecanismo não garante que as mesmas transações sejam inseridas na mesma ordem em todos os nós. Por exemplo, um seguidor que estava momentaneamente indisponível e, portanto, com um registro desatualizado, eleito como líder, substituiria os registros do líder anterior. Para evitar isso, o mecanismo de segurança do Raft garante que cada líder eleito tenha a última versão dos registros e, portanto, restringe os seguidores que podem participar de uma possível eleição.

As vantagens do Raft são: (i) fácil implementação, possuindo versões diversas linguagens de programação com C++, Go, Java e Python;(ii) Possibilidade de cada nó pode se tornar líder, garantindo um grau de justiça para os participantes; (iii) um sistema de eleição eficiente. Por outro lado, o Raft requer uma capacidade de armazenamento significativa, e algumas premissas adotadas reduzem sua aplicabilidade, por exemplo, a ausência de falhas bizantinas.

4.2.3 *Tendermint*

O Tendermint [25] é um protocolo de consenso capaz de punir os participantes que contribuem para as ramificações da cadeia de blocos. Ele pode tolerar até $\frac{N}{3}$ nós bizantinos. Os nós responsáveis pela validação das transações são chamados de validadores. Um nó precisa comprometer uma parte do seu saldo atual para se tornar um validador. Enquanto ele quiser participar do processo de validação, este montante deve permanecer comprometido.

O Tendermint propõe resolver os problemas encontrados na PoW adicionando uma forma de punição aos nós maliciosos e prevenindo mais efetivamente o ataque de gasto duplo. Esta punição evita a criação de ramificações e o problema conhecido como "*nothing-at-stake*", onde os participantes não recebem nenhuma punição se forem encontrados agindo maliciosamente. O Tendermint não requer o uso de alto poder computacional, impedindo que a validação das transações seja restrita a grupos específicos na rede. Outro fator que permite uma melhor distribuição de lucro é a divisão das taxas de validação entre os participantes de uma rodada de consenso.

O funcionamento do protocolo consiste de três etapas primárias e duas adicionais. As etapas primárias são: proposta, pré-voto e pré-compromisso. Nestes passos, um validador, escolhido através de um esquema de alternância cíclica ponderado pela capacidade de voto dos candidatos, é responsável por criar e propor um novo bloco a ser validado, ou terminar a validação de um bloco anterior. Se a rede aceitar um bloco durante a fase de pré-compromisso, todos os validadores que o aceitaram ficam bloqueados até sua confirmação. O processo de bloqueio de cada nó garante a segurança do consenso.

Se a rede rejeita um bloco, uma prova de rejeição, formada por uma lista com a identificação dos nós que rejeitaram o bloco, é criada e utilizada para desbloquear os validadores. A prova de rejeição evita que os nós permaneçam bloqueados indefinidamente, garantindo assim a vivacidade da rede.

Durante a etapa de validação, se um nó detectar a proposição de dois blocos distintos, é possível descobrir quais validadores são responsáveis por tentar ramificar a cadeia de blocos com base na chave pública presente no bloco proposto. O protocolo então pune esses nós com a perda de um terço dos valores comprometidos com a participação do consenso. Cada validador monitora o comportamento um do outro para lidar com as partições na rede. Um validador que não participa de um determinado número de rodadas de consenso é considerado desconectado da rede. Um validador desconectado tem seu *status* revogado e seu saldo liberado. Ao reingressar na rede, o nó pode se tornar um validador novamente, comprometendo parte de seu saldo.

4.2.4 *Practical Byzantine Fault Tolerance*

O *Practical Byzantine Fault Tolerance* (PBFT) [10] atua em um cenário padrão em sistemas distribuídos. Ele considera a existência de nós conectados por uma rede que pode não entregar todas as mensagens, atrasar sua entrega, duplicá-las ou entregá-las fora de ordem. O PBFT também assume que as falhas são independentes e que os nós dependem parcialmente um do outro, ou seja, um ambiente de cadeia de blocos privada.

Como indicado pela sua denominação, o objetivo do PBFT é agir como uma solução prática para o problema dos generais bizantinos. Portanto, o PBFT garante segurança e vivacidade, mesmo com a existência $(n-1)/3$ de nós maliciosos, em que n é o total de nós na rede. Desta forma, procura-se evitar falhas gerais, mitigando o efeito dos nós maliciosos que agem sobre o sistema.

O algoritmo consiste em quatro partes, que seguem o formato Geral-Tenente, uma variação do modelo original em que só existiam Generais. Neste contexto, os nós seguem uma hierarquia representada pela existência de um nó líder. Os quatro passos são os seguintes.

1. Um cliente envia uma solicitação de transação para o nó principal;
2. O líder então transmite o pedido através da rede para outros nós;
3. Os demais nós executam o pedido e enviam uma resposta ao cliente;
4. O cliente aguarda a chegada de $2f + 1$ respostas com o mesmo resultado, onde f representa o número máximo de nós que podem estar faltando.

Os nós devem ser determinísticos, e todos eles partem do mesmo estado inicial. Os

resultados do algoritmo são que nós honestos concordam na ordem das transações e se eles devem aceitá-las. Desta forma, a maioria dos nós honestos evitaria a ação dos maliciosos através da comunicação estabelecida entre eles.

O PBFT resolve o problema da falha dos nós líderes com um modelo de troca baseado em alternância cíclica. A troca de liderança é acionada quando o líder passa um determinado período sem solicitações. Outra forma de mudar a liderança ocorre quando a grande maioria dos nós decide que o líder está com falhas. Neste caso, o próximo na linha assume a posição do líder.

Algumas das vantagens do PBFT estão relacionadas com o fato de ter um baixo custo energético, e também o mecanismo de consenso tem um tempo de execução abaixo da média para sistemas assíncronos com um pequeno aumento na latência. No entanto, na cadeia de blocos com um grande número de pares, as limitações do mecanismo são evidentes. Um grande número de mensagens trocadas para obter consenso implica uma perda significativa de desempenho em redes maiores. Outra desvantagem é que o PBFT é suscetível a ataques de Sybil.

4.2.5 BFT-SMaRt

O BFT-SMaRt é um mecanismo de máquinas de estado replicadas tolerante a falhas bizantinas, utilizado para alcançar um consenso sobre plataformas de cadeia de bloqueios permissionadas [7]. Seu funcionamento é baseado na eleição de um líder responsável pela validação dos pedidos de transação recebidos de outros nós. Não há processo de eleição neste mecanismo, pois uma lista identifica os nós responsáveis pelo consenso durante a criação da rede. Sempre que um líder não se comunica com o resto da rede, o próximo nó dessa lista assume automaticamente a liderança.

No BFT-SMaRt, um nó que queira propor uma transação deve enviar essa transação para todos os outros nós da rede. No início de uma rodada de votação, o líder envia uma lista de transações para cada nó da rede. Ao receber essa lista, os nós verificam se a transação está em sua base local e a validam enviando seu voto para o líder. O envio de votos a todos os nós permite verificar se um nó enviou dois votos diferentes para a mesma transação. A Figura 4.3 apresenta a troca de mensagens realizada durante uma rodada do mecanismo BFT-SMaRt, ao receber o número mínimo de confirmações o líder considera o novo estado como válido.

Ao receber todos os votos, o líder informa aos nós quais transações foram aprovadas,

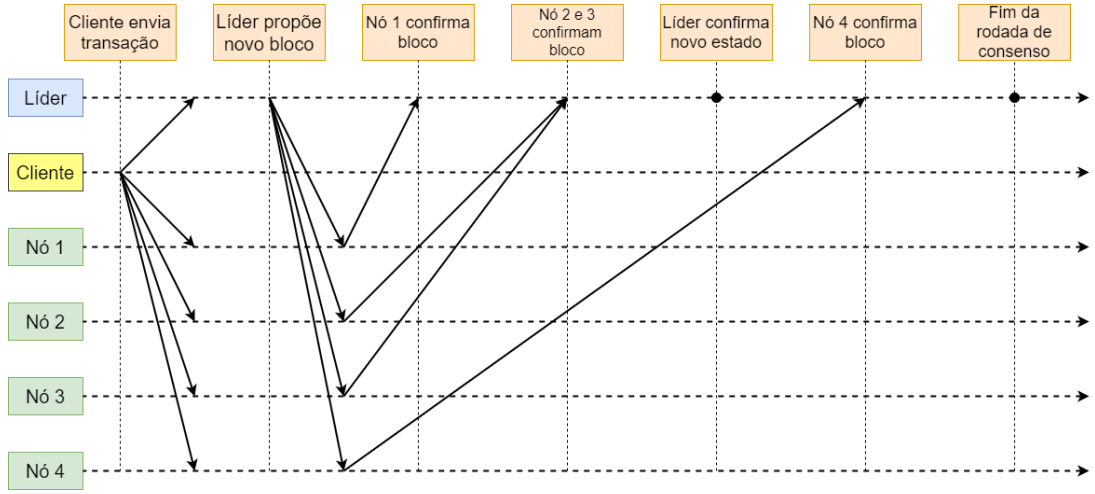


Figura 4.3: Diagrama de sequência representando a troca de mensagens necessária para realização do consenso no mecanismo BFT-SMaRt. No BFT-SMaRt as requisições são enviadas para todas os participantes do consenso, gerando assim um alto número de troca de mensagens. São necessários $\frac{2n}{3}+1$ confirmações para validar um novo bloco.

e anexadas ao histórico local. O líder também envia a lista de votos que recebeu a todos os nós da rede. Isto lhes permite verificar se outros nós votaram de forma diferente para o mesmo conjunto de transações, garantindo a segurança do mecanismo. Se uma transação permanecer por muito tempo na base de dados local de um nó sem validação, ou se o nó receber uma solicitação de voto para uma transação que não possui, ele solicita imediatamente a última versão das informações da rede ao líder.

Na presença de partições de rede, o BFT-SMaRt pode continuar a receber pedidos desde que as partições resultantes tenham nós suficientes para realizar a votação. Neste caso, cada partição irá eventualmente convergir para diferentes versões da rede, comprometendo a consistência de toda a rede em troca de manter a sua disponibilidade.

4.3 Conclusão do Capítulo

Os mecanismos de consenso para cadeias de blocos utilizam diferentes técnicas para contornar as limitações do Teorema CAP e da impossibilidade FLP. Essas técnicas permitem a criação de mecanismos com diferentes requisitos de uso e cenários de aplicação. Mecanismos probabilísticos sacrificam determinismo em troca de uma convergência eventual.

Uma consequência dessa escolha é o fato de que, nas cadeias de blocos que implementam tais mecanismos, é necessária a utilização de técnicas capazes de garantir a segurança dos dados da cadeia de blocos como uso de provas ou empenho de recursos. Os mecanis-

mos de consenso determinístico tendem a aplicar técnicas de sincronização para garantir que o consenso seja alcançado. Essas técnicas utilizam uma grande quantidade de troca de mensagens entre os nós para garantir que seus estados permaneçam sincronizados. Como consequência, esses mecanismos são mais sensíveis ao mal funcionamento da rede do que os mecanismos probabilísticos.

Perdas ou atrasos de pacotes, assim como partições na rede podem afetar a vivacidade e segurança desses mecanismos. Uma partição da rede pode se tornar incapaz de realizar o consenso por possuir um número de nós insuficiente ou duas partições podem operar de maneira independente e convergir para estados distintos. A aplicação de técnicas para aprimorar o funcionamento da rede pode auxiliar o funcionamento destes mecanismos de consenso.

Capítulo 5

Proposta de uma Estratégia Leve para Confiabilidade de Mecanismos de Consenso

Mecanismos de consenso são responsáveis por replicar e manter a visão do estado da rede em sistemas distribuídos. No ambiente da cadeia de blocos, os mecanismos de consenso desempenham um papel crucial para garantir que o estado atual da cadeia de blocos seja acessível a todos os pares participantes. As redes da cadeia de blocos podem ser classificadas baseadas na forma de acesso de seus participantes. Com isso elas podem ser dividas em duas categorias, públicas e privadas, cada uma das quais tem requisitos diferentes para alcançar o consenso [35].

Neste trabalho os mecanismos de consenso para cadeia de blocos foram divididos em dois conjuntos de acordo com a maneira como os nós da rede interagem entre si para alcançar o consenso. O primeiro conjunto é formado pelos os mecanismos de consenso baseados em prova. Os mecanismos pertencentes a este conjunto trabalham de maneira competitiva para forjar um novo bloco e assim obter uma recompensa pelo seu investimento.

Na cadeias de blocos públicas, não há controle de acesso para rede nem confiança entre os participantes. Por este motivo, há necessidade de implementar mecanismos de consenso dependentes de recursos, nos quais os participantes do consenso comprometem grandes quantidades de recursos computacionais para desencorajar a realização de fraudes, evitando a criação de identidades falsas e permitindo que o resto da rede valide a resposta rapidamente.

Cadeias de blocos privadas geralmente incluem participantes de alguma organização empresarial ou consórcio. Nesses casos, há o controle de acesso dos participantes da rede.

Os interessados também gerenciam o funcionamento da rede, mantendo a integridade das transações realizadas na cadeia de blocos. Como as cadeias de blocos privadas muitas vezes dependem da infraestrutura de recuperação de falhas para ajudar na resolução de disputas. Estas cadeias de blocos utilizam mecanismos de consenso baseados em votação que não requerem o comprometimento de recursos computacionais, ao custo de demandar um maior número de mensagens a serem trocadas.

O elevado número de mensagens trocadas torna os mecanismos de consenso baseados em votação mais sensíveis ao estado de contenção da rede. Problemas como perda ou atrasos de pacotes podem impedir que esses mecanismos funcionem corretamente. Mensagens atrasadas ou perdidas prejudicam a propriedade de vivacidade, pois dessa maneira, a votação pode nunca convergir. A segurança dos mecanismos também pode ser prejudicada, pois a impossibilidade de comunicação entre processos pode levar os nós a considerarem que uma partição ocorreu na rede. Nesse caso, os nós podem convergir para valores diferentes.

As plataformas de cadeias de blocos dependem do uso de mecanismos de consenso para replicar seus dados entre todos os participantes corretamente [3]. Os mecanismos de consenso garantem a terminação e a segurança. A terminação consiste em prever que o mecanismo de consenso não entre em um impasse. A segurança refere-se a evitar que dois nós convirjam para valores de consenso distintos. Uma falha no mecanismo pode comprometer a informação na cadeia de blocos. Neste trabalho são avaliados o desempenho dos mecanismos de consenso Raft e BFT-SMaRt e o impacto causado pela aplicação da estratégia proposta em seu funcionamento. Esse mecanismos foram escolhidos por serem implementados pelas principais plataformas de desenvolvimento de aplicações baseadas em cadeia de blocos, R3 Corda e Hyperldger Fabric.

5.1 Redes Definidas por *Software*

O paradigma de redes definidas por *software* (*Software Defined Networking*, SDN) separa fisicamente o controle e o encaminhamento de dados das redes de computadores tradicionais em diferentes planos [23]. Um plano de controle, logicamente centralizado, é responsável pelo gerenciamento das funções da redes. Nesse plano regras podem ser definidas e instalados na rede através do controlador SDN. O controlador é responsável pela comunicação com os dispositivos de encaminhamento presentes no plano de dados. Por sua vez, os dispositivos de encaminhamento possuem somente a função de encaminhar

os dados da rede segundo as regras definidas pelo plano de controle.

A Figura 5.1 representa um topologia típica de redes SDN, nela os dispositivos de encaminhamento, parte do plano de dados, se comunicam com o controlador SDN através da *API Southbound* e o controlador se comunica com as aplicações da rede através da *API Northbound*. Esta API permite que as regras de funcionamento da rede sejam definidas no controlador.

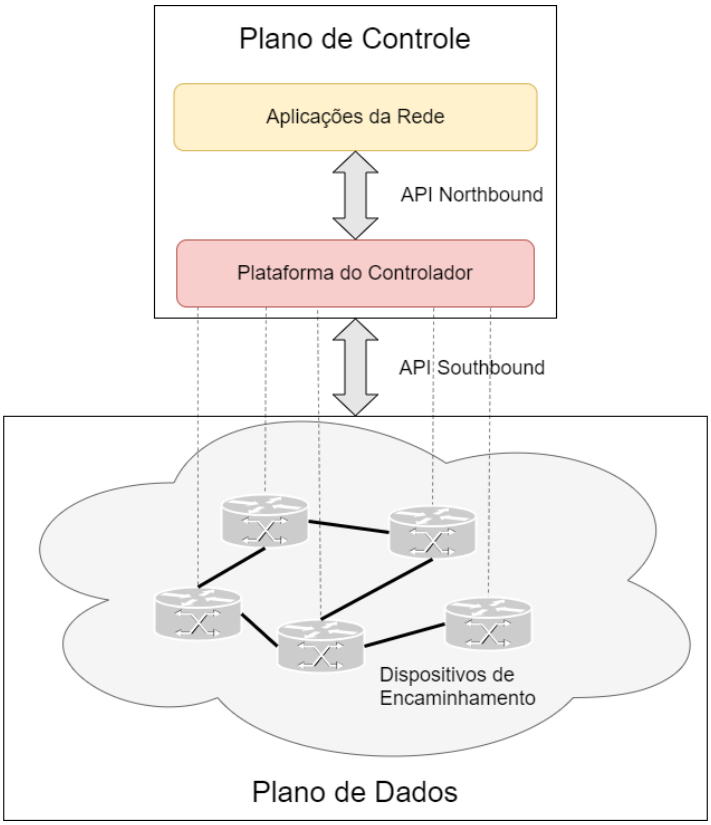


Figura 5.1: Visão simplificada da arquitetura SDN. Adaptado de [23]

O uso de SDN simplifica o gerenciamento das redes de computadores, pois elimina a necessidade de configurar cada dispositivo de encaminhamento do plano de dados individualmente, como nas redes de computadores tradicionais. Ao invés disso o controlador é responsável por configurar todos os dispositivos de acordo com as regras definidas pleas aplicações de rede. A redução da complexidade de configuração e gerenciamento da rede traz consigo economia de tempo e redução de custos.

Entre os protocolos de comunicação para *API Southbound* o mais implementado em controladores e dispositivos de encaminhamento é o OpenFlow [23]. O OpenFlow [32] é um protocolo de comunicação baseado na criação de tabelas de fluxos no plano de dados. Ao receber um pacote, um dispositivo de encaminhamento deve verificar se existe em suas tabelas de encaminhamento uma regra compatível com ele. Caso tal regra exista ele

simplesmente encaminha o pacote de acordo com a regra estabelecida. Caso contrário, o dispositivo pode enviar um requisição para o plano de controle para que uma decisão seja tomada. O plano de controle ao receber o pacote avalia suas informações e, então pode definir uma nova regra. A nova regra definida é então instalada nos dispositivos relevantes.

Neste trabalho as redes SDN foram utilizadas para simplificar o processo de criação de filas de qualidade de serviço. A versão 1.3 do OpenFlow fornece mecanismos que permitem que filas sejam definidas com atributos como limites de banda. O controlador ao processar novos pacotes envia comandos para os dispositivos de encaminhamento alocando esses pacotes em uma dessas filas.

5.2 Estratégia para Prover Confiabilidade ao Consenso

A principal contribuição deste trabalho consiste em uma estratégia leve que visa a distribuição de recursos entre os diferentes fluxos da rede. A ideia chave é definir políticas de reserva dirigidas aos fluxos de consenso. Estas políticas foram empiricamente definidas e traduzidas na criação de filas com reservas de recursos em cada dispositivo de roteamento. Assim, quando o controlador cria um novo fluxo, ele decide, com base na política de encaminhamento de pacotes definida em alto nível, a qual fila ele deve atribuir o fluxo. Portanto, para fornecer qualidade de serviço aos fluxos do mecanismo de consenso, o controlador da rede aloca o fluxo de consenso em filas de maior prioridade.

A estratégia utiliza o mecanismo de criação de filas disponibilizado pelo protocolo *OpenFlow* a partir da versão 1.3. Esse mecanismo possibilita que filas sejam criadas nas portas de saída de cada comutador da rede. Para cada fila criada podem ser definidas restrições de recursos, como por exemplo alocação de banda, na forma de limites máximos e mínimos. A Figura 5.2 mostra a arquitetura da estratégia proposta. As aplicações de rede fornecem uma *API REST* para a definição de políticas de qualidade de serviço. O controlador, por sua vez, traduz as políticas e as lança no plano de dados através da *API southbound*.

A reserva de banda para os fluxos de consenso tem como objetivo garantir a propriedade de vivacidade dos mecanismos de consenso. Entre as propriedades importantes para o projeto de mecanismos de consenso, a vivacidade está relacionada ao funcionamento da rede. Se as mensagens trocadas pelos nós participantes do consenso forem perdidas ou entregues com atraso o funcionamento dos mecanismos pode ser prejudicado ou até

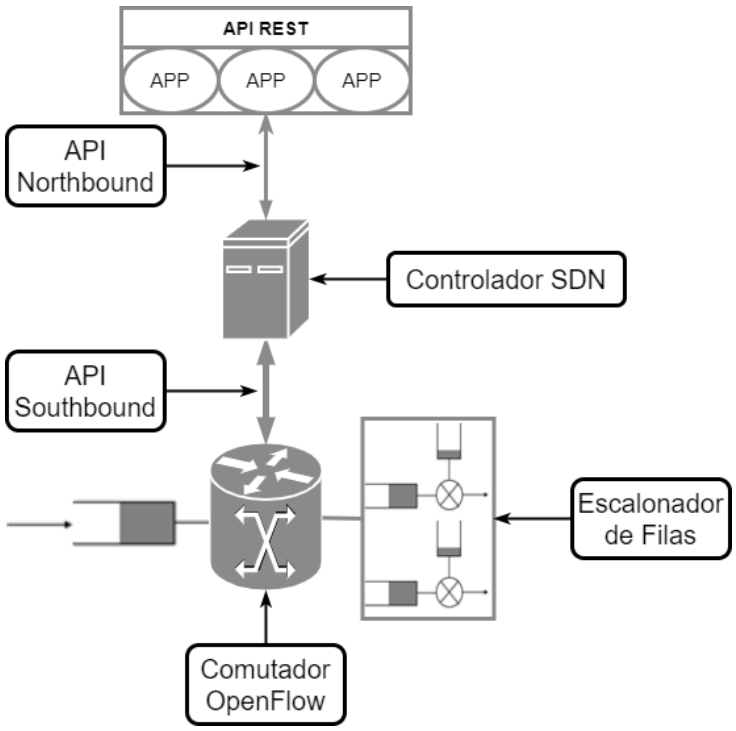


Figura 5.2: Diagrama da estratégia proposta. As políticas são instaladas no controlador através da *API REST* das aplicações. O controlador traduz as políticas em filas do plano de dados através do *API southbound*.

mesmo impedido. Com isso, a aplicação de técnicas de qualidade de serviço pode ter grande impacto no funcionamento desses mecanismos e na manutenção das propriedades necessárias para seu correto funcionamento.

Para garantir o funcionamento dos mecanismos de consenso são criadas em cada comutador da rede duas filas. Na primeira fila são alocados fluxos que não pertençam ao mecanismo de consenso e ela possui banda máxima limitada porém não possui banda mínima garantida. Essa configuração é aplicada na forma de uma limitação na taxa de transferência da fila designada.

O Algoritmo 4.1 apresenta o funcionamento da seleção das filas executado pela aplicação implementada no controlador. Primeiro as filas são configuradas com os limites de banda designadas. Em seguida, o controlador aguarda uma requisição de novo fluxo. Ao receber as informações do fluxo ele verifica se os campos do cabeçalho TCP, como a porta, são compatíveis com os fluxos dos mecanismos de consenso. Por fim, o controlador retorna uma resposta indicando a qual fila o fluxo de ser designado.

A segunda fila destina-se para a alocação de fluxos pertencentes ao mecanismo de consenso e para essa fila é definido um limite mínimo de uso de banda. Assim, uma parte da taxa de transferência de cada enlace da rede é reservado para os fluxos do consenso

mesmo em cenários de contenção. Dessa maneira, caso haja congestionamento ou gargalos na rede, os comutadores serão capazes de encaminhar os pacotes pertencentes aos fluxos do mecanismo de consenso.

```
1  Algoritmo Encaminhamento por Fila
2  Entrada banda_consenso
3      Fila_A->banda_min = banda_consenso \\reservada para o consenso
4      Fila_A->porta = porta_consenso \\porta do consenso
5      Fila_B->banda_max = (banda_rede - banda_consenso)
6      Enquanto sistema_executando
7          Aguarda Fluxo \\packet_in
8          se Fluxo->Porta_Dst == Fila_A->Porta
9              resposta<-Fila_A //Fluxo do consenso
10         senão
11             resposta<-Fila_B //Outros Fluxos
12         retorna resposta
```

Algoritmo 5.1: Algoritmo de encaminhamento de fluxos para filas designadas.

Do ponto de vista de rede, o funcionamento dos mecanismos de consenso avaliados é baseada em protocolos da camada de aplicação. Para ambos os mecanismos avaliados, ao se iniciar sua execução, cada um dos nós inicia uma conexão TCP com os demais. O estado dessa conexão é utilizada para verificar o estado dos nós vizinhos. Quando uma conexão com um nó é interrompida, aquele nó é considerável inoperante e seu voto é desconsiderado durante a rodada de consenso. É necessário uma quantidade mínima de conexões com outros nós para que a execução do mecanismos seja iniciada.

5.3 Conclusão do Capítulo

Mecanismos de consenso baseados em votação necessitam trocar um grande número de mensagens para garantir sua convergência. O estado de contenção da rede tem grande impacto nesses mecanismos podendo impedir sua terminação. Esses mecanismos são normalmente utilizados em redes privadas onde é possível gerenciar seus recursos.

A aplicação de técnicas de qualidade de serviço (QoS) pode reduzir problemas como perda ou atraso de pacotes na rede. Com o auxílio dessas técnicas é possível prover confiabilidade para mecanismos de consenso baseados em votação e assim garantir suas propriedades de vivacidade e segurança. A vivacidade é garantida pelo fato de que as mensagens serão entregues sem perdas ou atrasos e, como consequência, a segurança também é mantida pois o consenso será capaz de convergir.

Capítulo 6

Resultados e Discussão

Para realizar a avaliação da estratégia proposta foi criada, em um ambiente emulado com o auxílio do emulador *Mininet* a topologia apresentada na Seção 6.1. O ambiente de emulação foi uma máquina virtual configurada com 4 núcleos de processamento e 8GB de memória RAM com o sistema operacional Linux Ubuntu 16.04 LTS. A ferramenta *tcpdump*[1] foi utilizada para realizar a captura dos pacotes da rede.

O arcabouço Ryu[2] foi empregado como o controlador de rede. Ryu traduz as políticas de reserva de recursos em regras no plano de dados e permite a criação das filas de QoS. Para isso, a estratégia proposta se comunica com o controlador através da *API REST Northbound* disponível no controlador. O controlador exporta interfaces de programação que permitem a criação de filas gerenciadas pelo algoritmo *Hierarchical Token Bucket* (HTB) nas portas de saída dos comutadores da rede.

O algoritmo HTB possibilita a criação de filas nas portas de saída dos comutadores da rede. Ele permite que essas filas sejam separadas em classes e subclasses com diferentes quantidades de recursos alocados. A Figura 6.1 apresenta um exemplo de criação de filas usando o HTB. No topo é definida uma classe que possui a vazão total da rede. Essa vazão é calculada na forma de *tokens*. Um reservatório, o balde (*bucket*), é preenchido na taxa da vazão da rede. Os recursos da classe principal são então repassados para duas subclasses e, assim, preenchem seus baldes. Cada subclasse é responsável por gerenciar uma fila por onde passam fluxos.

Sempre que um pacote de um fluxo é encaminhado por uma fila, o tamanho desse pacote e removido do balde. A estrutura do balde permite armazenar uma certa quantidade de *tokens* de maneira que em caso da chegada de uma rajada de pacotes as filas sejam

[1]Disponível em https://www.tcpdump.org/.
[2]Disponível em https://osrg.github.io/ryu/.

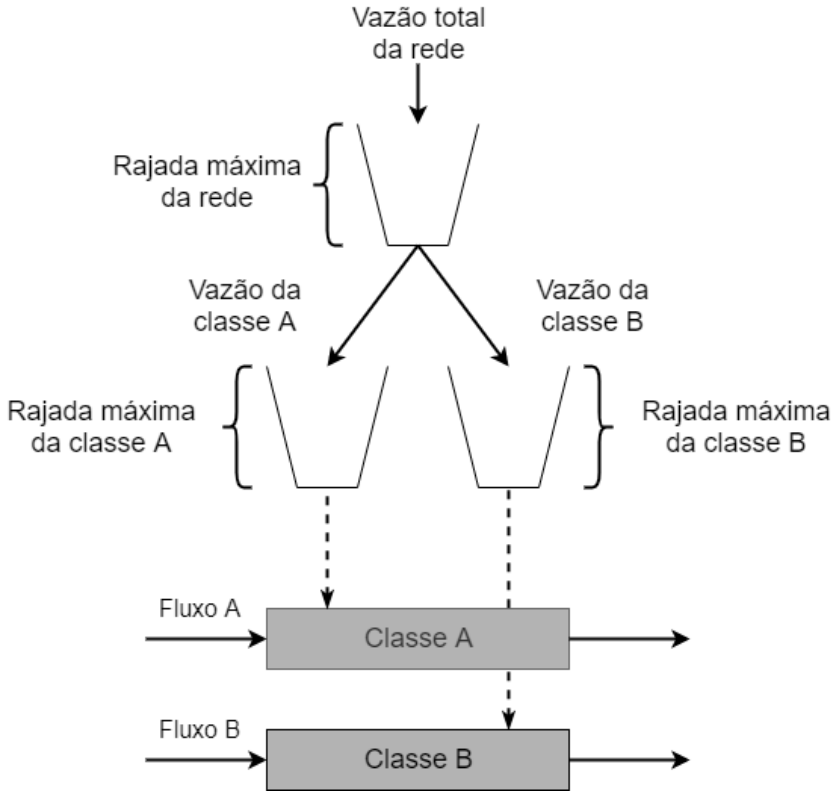


Figura 6.1: Exemplo de classes criadas utilizando o algoritmo HTB.

capazes de atendê-las. Ao esvaziar seu balde, uma fila só é capaz de encaminhar pacotes a taxa média da vazão permitida. Dentre os algoritmos de encaminhamento disponíveis no núcleo (*kernel*) do sistema operacional Linux, o HTB é a que possui a implementação mais leve e versátil permitindo assim que a estratégia implementada com ele não utilize muitos recursos computacionais.

O critério utilizado para discriminar fluxos em filas foram os campos do cabeçalho TCP utilizados por cada mecanismo de consenso, uma vez que é definida estaticamente. Assim, em cada cenário, foram criadas duas filas, a primeira para pacotes pertencentes aos mecanismos de consenso com uma vazão mínima garantida e a outra para pacotes de melhor esforço sem largura de banda garantida.

Foram escolhidos cenários em que a rede apresenta diferentes níveis de congestionamento para avaliar o impacto da estratégia sobre os mecanismos de consenso. Os cenários avaliados consideram enlaces com uma capacidade de 100Mb/s, o valor escolhido foi limitados pela capacidade de encaminhamento do ambiente emulado. Cada estação executa o mecanismo de consenso e tráfego de fundo, criado com a ferramenta iPerf. O tráfego de fundo é composto por fluxos UDP, variando de 10Mb/s a 50Mb/s de taxas de transferência. Cada cenário foi testado com e sem o uso da aplicação de políticas. Além disso,

definimos diferentes valores de reserva de recursos para determinar a melhor configuração de reserva de recursos para cada mecanismo.

A avaliação experimental da proposta foi realizada em duas etapas: (i) primeiro o funcionamento do mecanismo de criação de filas do controlador foi avaliada, (ii) foram realizados experimentos com o protótipo da proposta em cenários com diferentes graus de congestionamento.

Cada rodada de teste seguiu os seguintes passos. Primeiro, a topologia foi criada com o Mininet, depois as políticas de qualidade de serviço foram instaladas no controlador e imediatamente traduzidas para os comutadores de rede. Após a instalação das regras, cada hospedeiro ligado a uma folha de árvore começa a transmitir o seu tráfego de fundo. O tráfego segue o modelo de tráfego interno *Scatter-Gather* encontrado em centros de dados, onde cada estações envia ou recebe quantidades significativas de dados de outros estações em pontos distantes da rede [20].

Ao mesmo tempo, à medida que o tráfego de fundo é transmitido, as instâncias de mecanismos de consenso são iniciadas nos estações. Foi utilizado um modelo simplificado de cadeia de blocos onde a estrutura da cadeia foi abstraído. Quando lançadas, cada instância de mecanismo de consenso procura fazer conexões com seus vizinhos para formar uma rede par-a-par. Quando os participantes da rede conectada atingem um valor mínimo, os mecanismos começam a enviar seus pedidos de mudança de estado.

Para a geração de novas transações, foi empregado no experimento um modelo estatístico baseado em dados reais segue o intervalo entre transações da rede Bitcoin [35]. Cada cenário do experimento foi executado 10 vezes e os resultados são apresentados com um intervalo de confiança de 95%.

6.1 Cenário Avaliado

Como a proposta aborda o uso da cadeia de blocos privada, considera-se a topologia típica de rede de um centro de dados, onde os estações trocam informações continuamente entre si, e algumas aplicações têm a exigência de manter seus dados sincronizados e consistentes. Para isso, eles empregam mecanismos de consenso. Como exemplo dessa aplicação pode-se considerar o uso da plataforma R3 Corda nesse ambiente. Nela diversos usuários podem trocar transações em pequena porções da cadeia de blocos e por isso é necessário reservar um parte dos recursos para garantir o funcionamento do mecanismo de consenso.

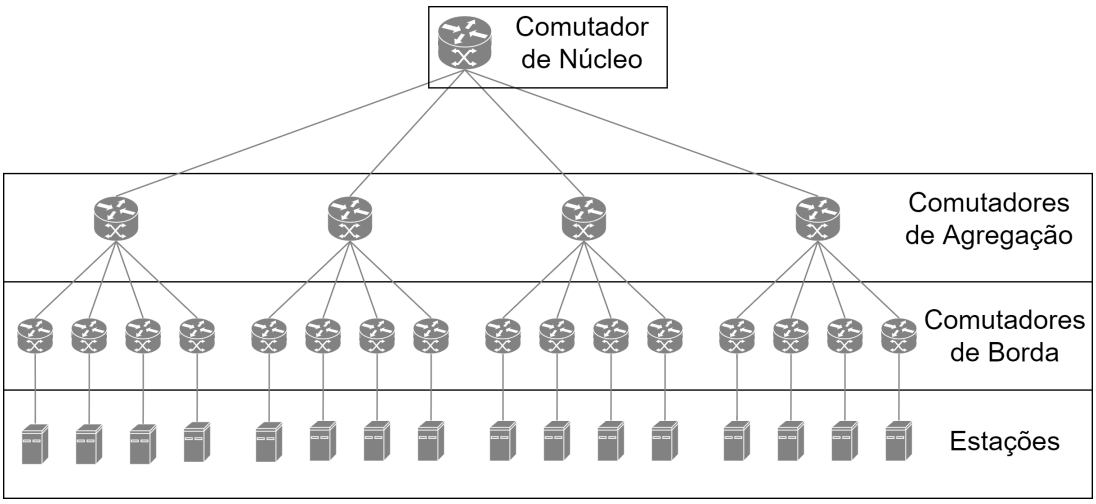


Figura 6.2: Topologia típica de árvores de centros de dados. As folhas contêm os nós hospedeiros responsáveis pela execução de aplicações como, por exemplo, protocolos de consenso.

A topologia considerada é uma topologia em árvore. A Figura 6.2 mostra a topologia, que é dividida em três níveis hierárquicos. O primeiro nível contém os hospedeiros responsáveis pela execução das aplicações do centro de dados. No segundo nível encontram-se os comutadores de borda responsáveis pela ligação dos hospedeiros ao resto da rede. No nível superior, encontram-se os comutadores de agregação que reúnem os dados da borda e definem para onde enviar os dados. Finalmente, o comutador de núcleo é responsável por conectar a topologia e atuar como um *gateway* para a rede. Neste trabalho, as funções de *gateway* comutador de núcleo foram desconsideradas pois o modelo de comunicação utilizado é apenas entre nós.

6.2 Avaliação do Mecanismo de Criação de Filas

A avaliação preliminar, teste de sanidade, valida a aplicação das políticas através das filas. A ferramenta iPerf[3] foi utilizada para transmitir dois fluxos UDP de 100Mb/s sobre um caminho de gargalo de 100Mb/s. A Figura 6.3 apresenta os resultados da avaliação. No instante 30s, as políticas são instaladas no controlador, que as traduz imediatamente em configurações de filas com limites 40Mb/s e 60Mb/s de largura de banda. Dos 30s a 36s, os fluxos ajustam-se aos limites impostos, e até os 60s, os fluxos permanecem dentro dos limites. Observa-se também que os fluxos ficam um pouco abaixo dos limites máximos estabelecidos, isso se deve a limitações presentes no meio de transmissão. Este resultado mostra a capacidade do controlador de criar e os comutadores de respeitar as regras de

[3]Disponível em https://iperf.fr/.

qualidade de serviço, validando as ferramentas utilizadas nos experimentos da proposta.

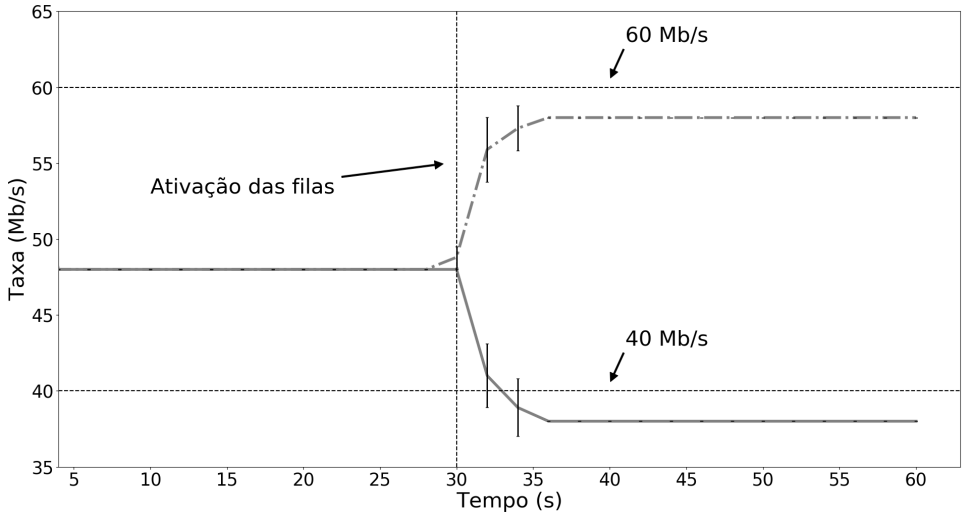
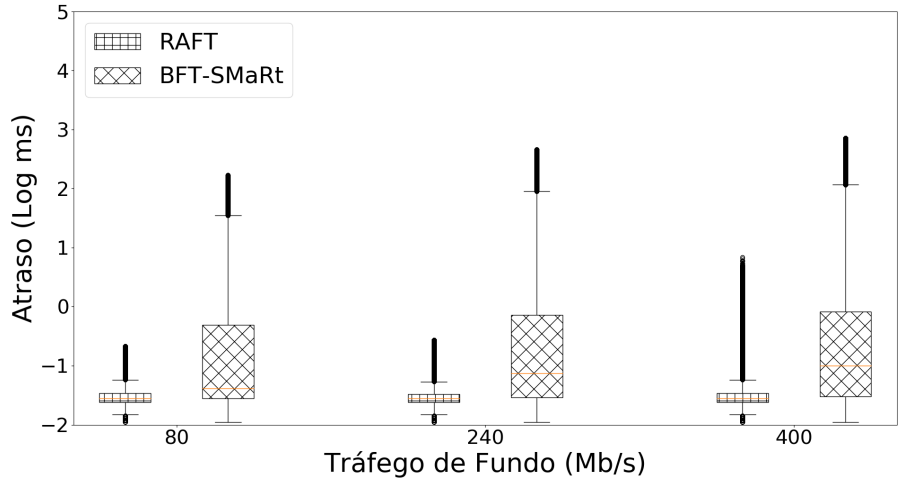


Figura 6.3: Teste de validação das regras criadas pelo controlador. Fluxos distintos são transmitidos em uma rede composta por dois hospedeiros e um comutador. Nos primeiros 30 segundos os fluxos disputam os recursos da rede. Aos 30s políticas são instaladas no controlador criando filas distintas para os fluxos.

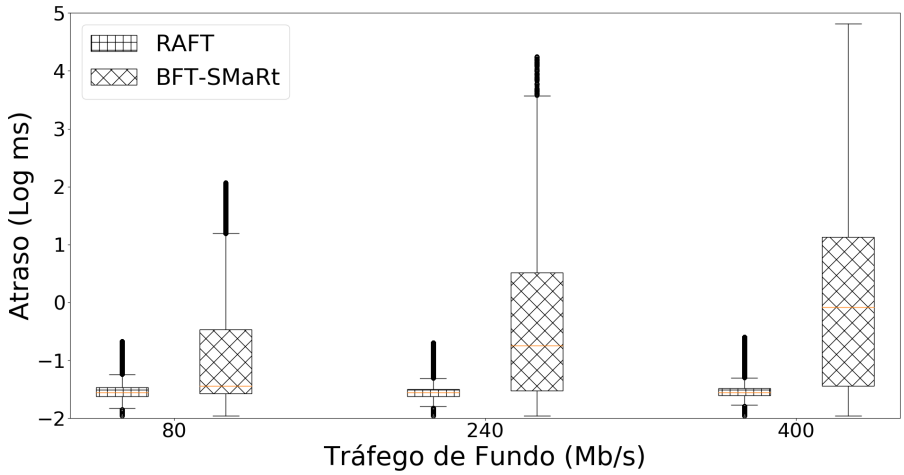
6.3 Atraso de Entrega de Pacotes

O objetivo deste experimento é avaliar o impacto causado no atraso de transmissão dos mecanismos de consenso pelo tempo de espera adicional nas filas. A Figura 6.4 mostra os tempos de espera de ambos os mecanismos para cenários sem reservas, Figura 6.4(a), com 1Mb/s, Figura 6.4(b), e com 20Mb/s de reserva, Figura 6.4(c). Para o consenso do Raft, todos os cenários mostram distribuições semelhantes no atraso da entrega de pacotes, uma vez que o Raft requer a troca de menos mensagens para alcançar o consenso. Entretanto, para o BFT-SMaRt, há um aumento nos tempos de fila em cada cenário de congestionamento. Esse aumento é proporcional ao tráfego de fundo devido ao limite de recursos da máquina virtual.

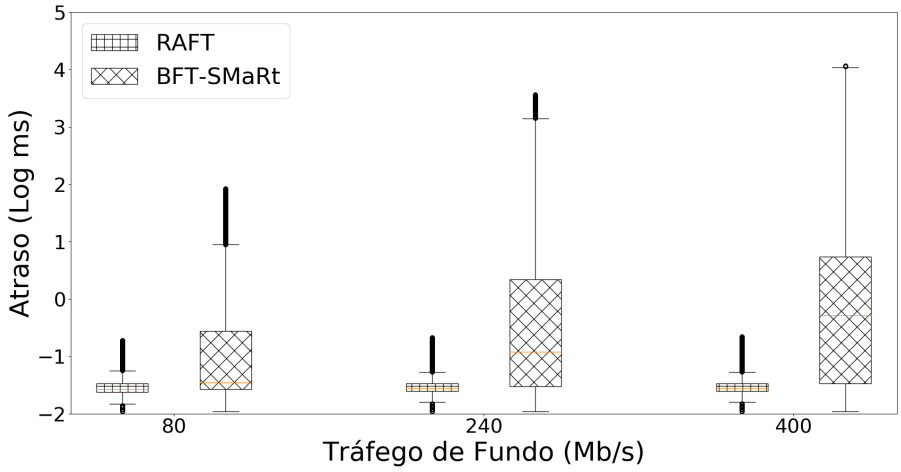
Nos cenários com 400Mb/s de tráfego de fundo pode-se observar um aumento do atraso. Este aumento é devido ao impacto da geração de tráfego de fundo adicional, já que todos as estações e os comutadores compartilham os mesmos recursos da máquina virtual. Entretanto, observa-se que para ambos os mecanismos, o cenário sem reservas, apresentou um aumento de *outliers* em relação aos outros casos. Assim, a estratégia proposta é eficaz na redução do atraso de transmissão de pacotes para os mecanismos.



(a) Sem reserva de banda.



(b) Reserva de 1 Mb/s.



(c) Reserva de 20 Mb/s.

Figura 6.4: Resultados dos testes para medir o tempo de fila dos pacotes de cada mecanismo.

6.4 Número de Terminações

A Figura 6.5 mostra que devido a sua baixa taxa de troca de mensagens, o mecanismo Raft alcança consenso mesmo em um cenário onde a rede está congestionada. Entretanto, sua operação se degrada quando 280Mb/s de tráfego de fundo é coalocado com o mecanismo de consenso em um cenário sem garantia de qualidade de serviço (*Quality of Service*, QoS). Ao ser executado em um cenário com garantia de QoS, o Raft melhora até 100% o número de terminações.

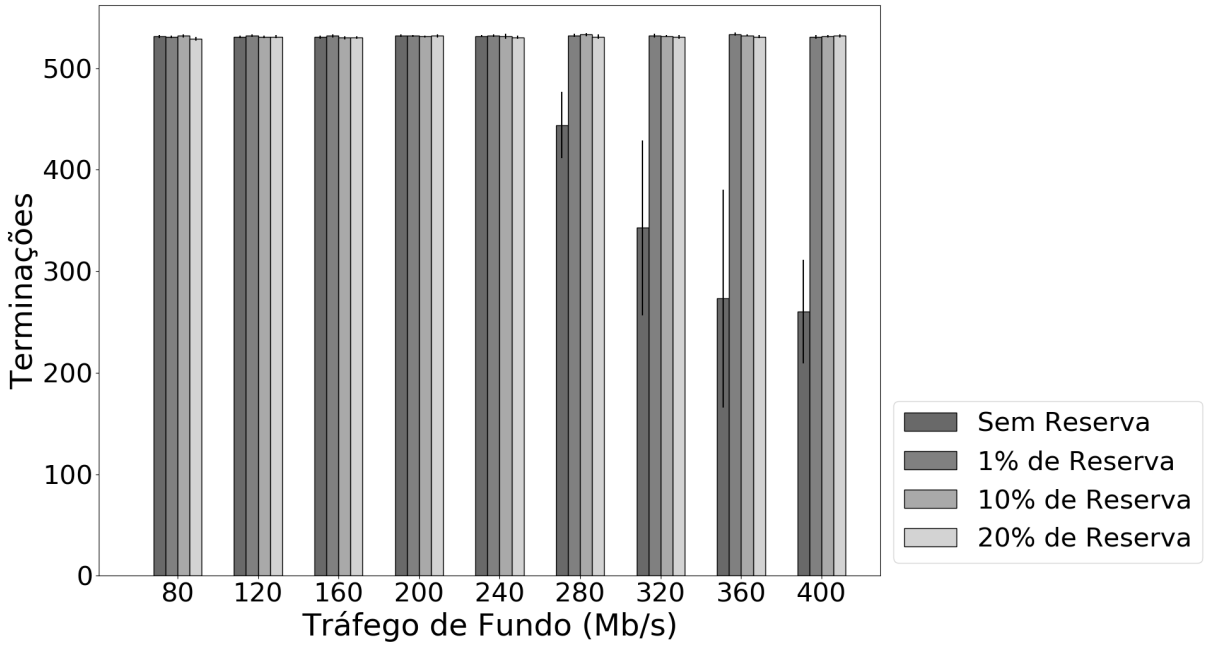


Figura 6.5: Número de terminações para o mecanismo RAFT com diferentes gruas de congestionamento.

Para o BFT-SMaRT, a eficácia da estratégia proposta é ainda mais evidente. A Figura 6.6 apresenta os resultados para o número de terminações do mecanismo do BFT-SMaRT. Os resultados atestam que no cenário sem reservas, o mecanismo não alcança consenso em cenários congestionados, a partir de 200Mb/s de tráfego de fundo. Isso ocorre devido à necessidade de cada nó enviar mensagens para cada outro nó participante. Por outro lado, em cenários com reserva de recursos, é possível observar o funcionamento correto do mecanismo em todos os cenários de congestionamento.

6.5 Custo por Terminação

A avaliação também considera o custo adicional de comunicação para alcançar um consenso para cada mecanismo. Essa métrica é definida na forma da razão do tráfego total

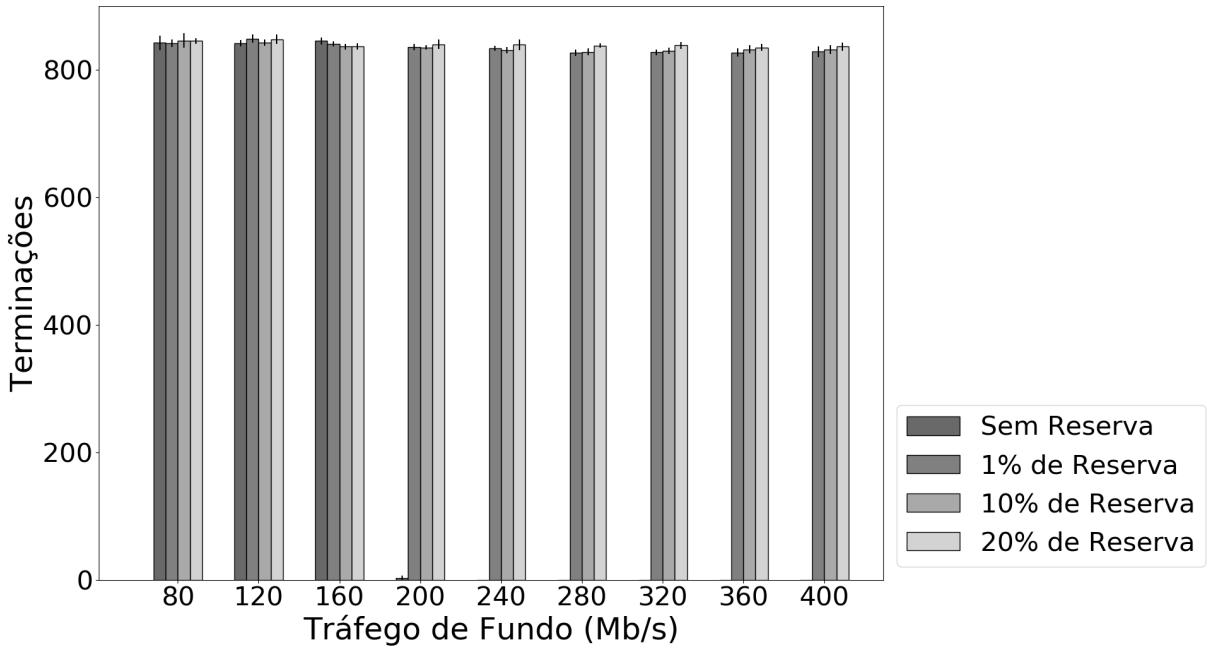


Figura 6.6: Número de terminações para o mecanismo BFT-SMaRT com diferentes gruas de congestionamento.

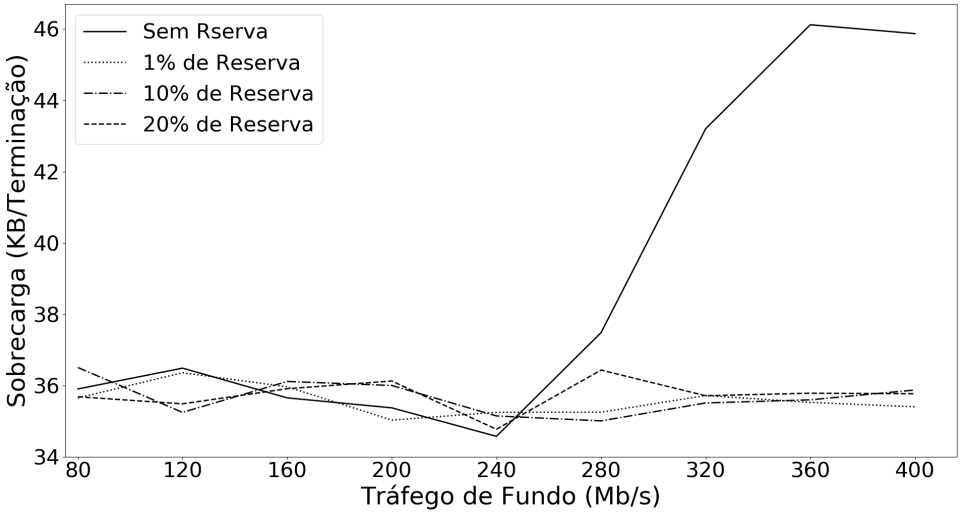
transmitido pelo número de terminações de uma rodada de experimento. Como ambos os mecanismos de consenso utilizam o protocolo TCP para trocar mensagens, a contenção da rede leva a retransmissões. A Figura 6.7 avalia o custo de cada terminação. A Figura 6.7(a) mostra os valores para a avaliação do Raft, nela observa-se um aumento de 30% na sobrecarga de mensagens por terminação para cenários sem reserva de recursos.

Para o BFT-SMaRt, os resultados são mais evidentes. Na Figura 6.7(b), é possível observar que, no cenário de tráfego de fundo de 200Mb/s, a sobrecarga de mensagens é 80% maior por terminação quando não há reserva de recursos. Em casos de tráfego de fundo acima de 200Mb/s, não há valores definidos porque o BFT-SMaRt não é capaz de alcançar consenso.

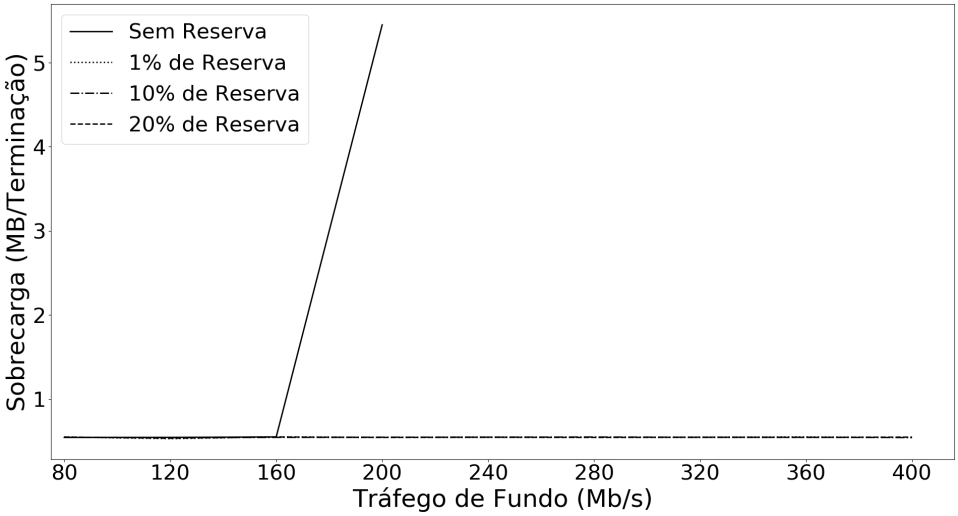
6.6 Conclusão do Capítulo

Neste Capítulo foram apresentados os resultados da avaliação experimental da proposta. Uma avaliação preliminar teve como objetivo validar a capacidade de criação e manutenção das filas. Os resultados demonstram que, embora o meio de transmissão apresente limitações, os comutadores foram capazes de respeitar as regras instaladas no controlador e traduzidas como filas.

Em seguida foram avaliados o atraso, número de terminações e o custo por terminação



(a) O custo estimado em KB de cada terminação para a RAFT.



(b) O custo estimado em MB de cada terminação para o BFT-SMaRt.

Figura 6.7: Custo por terminação para cada mecanismo de consenso. Não foram exibidas barras de erro porque os valores são relativos ao número de terminações, e em cenários e rodadas em que não houve terminação, o custo não é definido.

de cada mecanismo de consenso. Para cada experimento foram comparados os resultados dos cenários sem reservas de recursos com cenários com diferentes graus de reserva. Os resultados mostram que, mesmo nos cenários com apenas 1Mb/s de reserva de recursos, a aplicações de técnicas de qualidade de serviço foi capaz de garantir o funcionamento dos mecanismos de consenso. O impacto da reserva de recursos nos demais fluxos da rede pode ser considerado mínimo pois os mecanismos de consenso trocam mensagens em rajadas de curta duração. Durante o período em que os nós estão se comunicando os recursos reservados ficam disponíveis para os demais fluxos.

Capítulo 7

Conclusão

A tecnologia da cadeia de blocos depende de uma estrutura de dados distribuídos e de uma rede par-a-par de comunicação. Uma propriedade crítica é que a estrutura de dados da cadeia de blocos seja a mesma em todos os nós da rede. A consistência global da cadeia de blocos é dependente da consistência de sua estrutura de dados distribuída.

Como um sistema assíncrono distribuído, a cadeia de blocos depende do mecanismo de consenso para manter a consistência de seus dados compartilhados. No entanto, trabalhos anteriores provaram que o consenso é impossível em um sistema assíncrono na presença de nós defeituosos. Além disso, como qualquer sistema distribuído, o teorema da CAP também se aplica à cadeia de blocos e, portanto, é impossível garantir as propriedades de consistência, disponibilidade e tolerância à partições simultaneamente.

A utilização de mecanismos de consenso baseados em votação permite que os nós trabalhem de maneira cooperativa para alcançar o consenso. Essa abordagem permite a implementação de soluções sem a necessidade de empenho de grandes quantidades de recursos, como energia elétrica. Essa economia faz com que os mecanismos baseados em votação sejam utilizados em cadeias de blocos privadas. Essas cadeias são utilizadas por empresas ou consórcios para realização de negócios de maneira segura e descentralizada.

As plataformas de cadeia de blocos dependem de protocolos de consenso para funcionarem corretamente. Os mecanismos de consenso baseados em votação são especialmente sensíveis a problemas de rede porque precisam trocar um grande número de mensagens para realizar o consenso. Neste trabalho, foi desenvolvida uma estratégia leve para fornecer qualidade de serviço aos mecanismos de consenso baseados em votação através da alocação de recursos de rede pelo plano de controle. A garantia de qualidade de serviço permite que as mensagens do consenso não sofram atrasos ou perdas e assim garantem

sua vivacidade pois permite a convergência do protocolo. Como as mensagens não são perdidos, não há partições na rede oque garante que os nós irão convergir para os mesmos valores, o que garante a segurança dos mecanismos.

A proposta visa aplicar técnicas de qualidade de serviço através do uso de redes definidas por *software* (*SDN*). Através delas políticas de reservas de recursos podem ser definidas e instaladas na rede através do controlador. Em seguida, o controlador traduz as políticas em filas nas portas de saída de cada comutador da rede. Cada fila conta com reservas de recursos na forma de limites máximos e mínimos de utilização de banda. O controlador também decide para qual fila cada fluxo deve ser designado, alocando fluxos dos mecanismos de consenso em filas com maior prioridades de recursos.

Um protótipo da proposta foi testado com os mecanismos de consenso Raft e BFT-SMaRt em cenários com diferentes graus de congestionamento e reserva de recursos. Os resultados mostram que ambos os protocolos se beneficiaram da aplicação da estratégia mesmo no cenário com 1% de reserva de recursos. A estratégia proposta aumentou em até 100% as terminações do consenso no Raft e, para o BFT-SMaRt, garantiu o funcionamento mesmo em cenários de contenção, e reduziu em até 80% o número de mensagens trocadas para alcançar o consenso pois o número de retransmissões do protocolo TCP é reduzido.

O restante deste capítulo destina-se a apresentação das contribuições, uma discussão das limitações encontrados durante a avaliação experimental da proposta e por fim as possibilidades de aprimoramento, assim como, contribuições futuras que podem ser desenvolvidas a partir dos estudos realizados.

7.1 Contribuições

Além do protótipo desenvolvido, este trabalho apresenta outra contribuições visando auxiliar o estudo e desenvolvimento da tecnologia da cadeia de blocos. As demais contribuições são:

1. Uma revisão de como limitações do Teorema CAP e da impossibilidade FLP impactam o desenvolvimento de sistemas distribuídos (Capítulo 3);

2. Apresentação das características dos principais mecanismos de consenso probabilísticos e determinísticos. Além de uma breve analise de seu funcionamento a aplicação (Capítulo 4).

3. Uma discussão de como o projeto dos mecanismos de consenso apresentados lidam com as limitações impostas pelo Teorema CAP e a Impossibilidade FLP (Capítulo 4)

4. Aplicação de uma estratégia leve capaz de garantir o funcionamento e garantir a terminação, mantendo a propriedade de vivacidade, de mecanismos de consenso baseados em votação (Capítulo 5).

A realização deste trabalho também produziu contribuições na forma de publicações de artigos acadêmicos em conferencias e periódicos. As publicações realizadas são as seguintes:

- Um artigo, com o título "Uma Estratégia Leve para a Confiabilidade de Mecanismos de Consenso baseada em Redes Definidas por Software", publicado no XXXVII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBRT 2019).
- Um artigo, com o título "A Lightweight Strategy for Reliability of Consensus Mechanisms based on Software Defined Networks", publicado no Global Information Infrastructure Symposium (GIIS'19).
- Um artigo, com o título "Consistency, Availability and Partition Tolerance in Blockchain: A Survey on the Consensus Mechanism over Peer-to-Peer Networking", aceito na revista Annals of Telecommunications.

Espera-se que este trabalho seja útil tanto para pesquisadores quanto desenvolvedores. Pesquisadores poderão usá-lo como referência para pesquisas sobre mecanismos de consenso aplicados a tecnologia da cadeia de blocos. Para desenvolvedores a contribuição vem na forma de um exemplo de como o funcionamento de mecanismos de consenso pode ser aprimorado através da aplicação de técnicas e qualidade de serviço (QoS) e como o funcionamento da rede pode impactar o desempenho de tais mecanismos.

7.2 Trabalhos Futuros

Como trabalho futuro, pretende-se desenvolver um mecanismo híbrido de consenso baseado em difusão primitiva confirmada, recorrendo a canais de comunicação parcialmente sincronizados, e alcançando o acordo ao considerarmos apenas o voto dos vizinhos de um salto.

Referências

[1] Blockchain technologies are still five to 10 years away from transformational impact. https://www.gartner.com/en/newsroom/press-releases/2019-10-08-gartner-2019-hype-cycle-shows-most-blockchain-technologies-are-still-five-to-10-years-away-from-transformational-impact. Acessado: 2020-01-20.

[2] Gartner 2019 hype cycle for blockchain. https://www.gartner.com/en/newsroom/press-releases/2019-09-12-gartner-2019-hype-cycle-for-blockchain-business-shows. Acessado: 2020-01-20.

[3] Aggarwal, S.; Chaudhary, R.; Aujla, G. S.; Kumar, N.; Choo, K.-K. R.; Zomaya, A. Y. Blockchain for smart communities: Applications, challenges and opportunities. *Journal of Network and Computer Applications 144* (2019), 13 – 48.

[4] Ateniese, G.; Magri, B.; Venturi, D.; Andrade, E. Redactable blockchain–or–rewriting history in bitcoin and friends. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)* (2017), IEEE, pp. 111–126.

[5] Baliga, A. Understanding blockchain consensus models. In *Persistent*. 2017.

[6] Baliga, A.; Solanki, N.; Verekar, S.; Pednekar, A.; Kamat, P.; Chatterjee, S. Performance characterization of hyperledger fabric. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)* (June 2018), pp. 65–74.

[7] Bessani, A.; Sousa, J. a.; Alchieri, E. E. P. State machine replication for the masses with bft-smart. In *Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (Washington, DC, USA, 2014), DSN '14, IEEE Computer Society, pp. 355–362.

[8] Boichat, R.; Dutta, P.; Frølund, S.; Guerraoui, R. Deconstructing paxos. *ACM Sigact News 34*, 1 (2003), 47–67.

[9] Brewer, E. A. Towards robust distributed systems. In *PODC* (2000), vol. 7.

[10] Castro, M.; Liskov, B. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst. 20*, 4 (Nov. 2002), 398–461.

[11] Chen, L.; Xu, L.; Shah, N.; Gao, Z.; Lu, Y.; Shi, W. On security analysis of proof-of-elapsed-time (poet). In *International Symposium on Stabilization, Safety, and Security of Distributed Systems* (2017), Springer, pp. 282–297.

[12] Correia, M.; Veronese, G. S.; Neves, N. F.; Verissimo, P. Byzantine consensus in asynchronous message-passing systems: a survey. *International Journal of Critical Computer-Based Systems 2*, 2 (2011), 141–161.

[13] de Oliveira, M. T.; Reis, L. H. A.; Carrano, R. C.; Seixas, F. L.; Saade, D. C. M.; Albuquerque, C. V.; Fernandes, N. C.; Olabarriaga, S. D.; Medeiros, D. S. V.; Mattos, D. M. F. Towards a blockchain-based secure electronic medical record for healthcare applications. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)* (May 2019), pp. 1–6.

[14] Dolev, D.; Dwork, C.; Stockmeyer, L. On the minimal synchronism needed for distributed consensus. *J. ACM 34*, 1 (Jan. 1987), 77–97.

[15] Fischer, M. J.; Lynch, N. A.; Paterson, M. S. Impossibility of distributed consensus with one faulty process. Tech. rep., Massachusetts Inst of Tech Cambridge lab for Computer Science, 1982.

[16] Fischer, M. J.; Lynch, N. A.; Paterson, M. S. Impossibility of distributed consensus with one faulty process. *J. ACM 32*, 2 (Apr. 1985), 374–382.

[17] Golan Gueta, G.; Abraham, I.; Grossman, S.; Malkhi, D.; Pinkas, B.; Reiter, M.; Seredinschi, D.; Tamir, O.; Tomescu, A. Sbft: A scalable and decentralized trust infrastructure. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (June 2019), pp. 568–580.

[18] Huong-Truong, T.; Thanh, N. H.; Hung, N. T.; Mueller, J.; Magedanz, T. Qoe-aware resource provisioning and adaptation in ims-based iptv using openflow. In *2013 19th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN)* (2013), IEEE, pp. 1–3.

[19] Jalalzai, M. M.; Busch, C. Window based bft blockchain consensus. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (July 2018), pp. 971–979.

[20] Kandula, S.; Sengupta, S.; Greenberg, A.; Patel, P.; Chaiken, R. The nature of data center traffic: measurements & analysis. In *9th ACM SIGCOMM Conference on Internet Measurement* (2009), pp. 202–208.

[21] Katta, N.; Zhang, H.; Freedman, M.; Rexford, J. Ravana: Controller fault-tolerance in software-defined networking. In *Proceedings of the 1st ACM SIGCOMM symposium on software defined networking research* (2015), ACM, p. 4.

[22] Kotla, R.; Alvisi, L.; Dahlin, M.; Clement, A.; Wong, E. Zyzzyva: Speculative byzantine fault tolerance. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles* (New York, NY, USA, 2007), SOSP '07, ACM, pp. 45–58.

[23] Kreutz, D.; Ramos, F.; Verissimo, P.; Rothenberg, C. E.; Azodolmolky, S.; Uhlig, S. Software-defined networking: A comprehensive survey. *arXiv preprint arXiv:1406.0440* (2014).

[24] Kumar, H.; Gharakheili, H. H.; Sivaraman, V. User control of quality of experience in home networks using sdn. In *2013 IEEE International conference on advanced networks and telecommunications systems (ANTS)* (2013), IEEE, pp. 1–6.

[25] Kwon, J. Tendermint: Consensus without mining. *Draft v. 0.6, fall 1* (2014), 11.

[26] Lamport, L. Proving the correctness of multiprocess programs. *IEEE transactions on software engineering*, 2 (1977), 125–143.

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

[28] Lamport, L.; Shostak, R.; Pease, M. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS) 4*, 3 (1982), 382–401.

[29] Lantz, B.; Heller, B.; McKeown, N. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks* (2010), ACM, p. 19.

[30] Lin, P.; Bi, J.; Wolff, S.; Wang, Y.; Xu, A.; Chen, Z.; Hu, H.; Lin, Y. A west-east bridge based sdn inter-domain testbed. *IEEE Communications Magazine 53*, 2 (2015), 190–197.

[31] Mattos, D. M. F.; Duarte, O. C. M. B.; Pujolle, G. A lightweight protocol for consistent policy update on software-defined networking with multiple controllers. *Journal of Network and Computer Applications 122* (2018), 77 – 87.

[32] McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review 38*, 2 (2008), 69–74.

[33] Medeiros, D. S. V.; Fernandes, N. C.; Mattos, D. M. F. Smart contracts and the power grid: A survey. In *2019 1st Blockchain, Robotics and AI for Networking Security Conference (BRAINS)* (2019), BRAINS'19, IEEE, pp. 41–47.

[34] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system.

[35] Oliveira, M. T.; Carrara, G. R.; Fernandes, N. C.; Albuquerque, C.; Carrano, R.; Medeiros, D. S.; Mattos, D. Uma avaliação de desempenho de cadeias de blocos privadas permissionadas através de cargas de trabalho realísticas. In *XVIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais* (2018), pp. 309–322.

[36] Oliveira, M. T.; Carrara, G. R.; Fernandes, N. C.; Albuquerque, C. V. N.; Carrano, R. C.; Medeiros, D. S. V.; Mattos, D. M. F. Towards a performance evaluation of private blockchain frameworks using a realistic workload. In *22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)* (Feb 2019), pp. 180–187.

[37] Ongaro, D.; Ousterhout, J. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)* (Philadelphia, PA, June 2014), USENIX Association, pp. 305–319.

[38] Ryu, S. Framework community: Ryu sdn framework. *Online. http://osrg. github. io/ryu* (2015).

[39] Sakic, E.; Kellerer, W. Response time and availability study of raft consensus in distributed sdn control plane. *IEEE Transactions on Network and Service Management 15*, 1 (2017), 304–318.

[40] Seddiki, M. S.; Shahbaz, M.; Donovan, S.; Grover, S.; Park, M.; Feamster, N.; Song, Y.-Q. Flowqos: Qos for the rest of us. In *Proceedings of the third workshop on Hot topics in software defined networking* (2014), ACM, pp. 207–208.

[41] Sezer, S., et al. Are we ready for sdn? implementation challenges for software-defined networks. *IEEE Communications Magazine 51*, 7 (2013), 36–43.

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

[43] Van Adrichem, N. L.; Doerr, C.; Kuipers, F. A. Opennetmon: Network monitoring in openflow software-defined networks. In *2014 IEEE Network Operations and Management Symposium (NOMS)* (2014), IEEE, pp. 1–8.

[44] Vukolić, M. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *Open Problems in Network Security* (Cham, 2016), J. Camenisch and D. Kesdoğan, Eds., Springer International Publishing, pp. 112–125.

[45] Wang, W.; Hoang, D. T.; Hu, P.; Xiong, Z.; Niyato, D.; Wang, P.; Wen, Y.; Kim, D. I. A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access 7* (2019), 22328–22370.

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

[47] Woos, D.; Wilcox, J. R.; Anton, S.; Tatlock, Z.; Ernst, M. D.; Anderson, T. Planning for change in a formal verification of the raft consensus protocol. In *Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs* (New York, NY, USA, 2016), CPP 2016, ACM, pp. 154–165.

[48] Zhang, P.; Schmidt, D. C.; White, J.; Lenz, G. Blockchain technology use cases in healthcare. In *Advances in Computers*, vol. 111. Elsevier, 2018, pp. 1–41.

[49] Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE International Congress on Big Data (BigData Congress)* (2017), IEEE, pp. 557–564.