UNIVERSIDADE FEDERAL DE VIÇOSA

ArchChain: Proposta de um Token de Cessão para Avaliação de Custo e Desempenho em Redes Blockchain

Ronan Dutra Mendonça
Doctor Scientiae

VIÇOSA - MINAS GERAIS
2025

RONAN DUTRA MENDONÇA

ArchChain: Proposta de um Token de Cessão para Avaliação de Custo e Desempenho em Redes Blockchain

Tese apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Doctor Scientiae*.

Orientador: Jose Augusto Miranda Nacif

Coorientador: Alex Borges Vieira

VIÇOSA - MINAS GERAIS
2025

Ficha catalográfica elaborada pela Biblioteca Central da Universidade Federal de Viçosa - Campus Viçosa

T
M539a
2025

Mendonça, Ronan Dutra, 1979-
ArchChain: proposta de um token de cessão para avaliação de custo e desempenho em redes blockchain / Ronan Dutra Mendonça. – Viçosa, MG, 2025.
1 tese eletrônica (137 f.): il. (algumas color.).

Inclui apêndices.
Orientador: José Augusto Miranda Nacif.
Tese (doutorado) - Universidade Federal de Viçosa, Departamento de Informática, 2025.
Referências bibliográficas: f. 112-118.
DOI: https://doi.org/10.47328/ufvbbt.2025.805
Modo de acesso: World Wide Web.

1. Blockchains (Base de dados) - Metodologia - Custos. 2. Desempenho - Avaliação. 3. Tokens. 4. Transferência eletrônica de fundos. I. Nacif, José Augusto Miranda, 1978-. II. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD. 22. ed. 005.74

Bibliotecário(a) responsável: Bruna Silva CRB-6/2552

RONAN DUTRA MENDONÇA

ArchChain: Proposta de um Token de Cessão para Avaliação de Custo e Desempenho em Redes Blockchain

Tese apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Doctor Scientiae*.

APROVADA: 19 de setembro de 2025.

Assentimento:

_____

Ronan Dutra Mendonça
Autor

_____

Jose Augusto Miranda Nacif
Orientador

Dedico este trabalho primeiramente à minha esposa, Maria Geralda, cuja força, paciência e sacrifício silencioso foram o alicerce que sustentou cada passo desta jornada. Sua presença constante, mesmo nos momentos mais difíceis, foi luz e abrigo enviados por Deus. Aos meus filhos, Yasmin e Diego, pela lealdade inabalável e pelo amor que me fez manter firme quando o cansaço ameaçava vencer. Vocês são bênçãos que o Senhor me concedeu, e por isso sou eternamente grato. Dedico também aos meus pais e minhas irmãs, Roque, Leocádia, Geovana e Lissandra, por cada gesto silencioso, cada palavra de incentivo e cada oração feita em meu nome. Que esta conquista represente não apenas um marco pessoal, mas também uma homenagem à dedicação, ao esforço e à confiança que sempre depositaram em mim.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus, fonte de sabedoria e fortaleza, por ter sustentado minha caminhada e renovado minha esperança nos momentos de incerteza. Aos meus orientadores, Nacif e Alex, registro meu reconhecimento pela orientação segura, pela confiança depositada e pela generosidade na partilha do conhecimento. Suas contribuições foram decisivas para o desenvolvimento e aprimoramento deste trabalho. Agradeço, ainda, aos colegas, amigos e demais colaboradores que, direta ou indiretamente, contribuíram para esta pesquisa. Cada gesto de apoio, cada palavra de incentivo e cada crítica construtiva foram fundamentais para a superação dos obstáculos e para o alcance dos objetivos propostos. Expresso minha profunda gratidão à minha esposa, Maria Geralda, pelo apoio e sacrifício incondicional que foram essenciais para que eu pudesse perseverar até aqui. E aos meus filhos, Yasmin e Diego, que me inspiraram diariamente e me motivaram a seguir em frente com determinação. Que este trabalho possa representar o início de novas possibilidades, guiadas pela vontade de Deus e pela fé que me sustenta, pelo compromisso com o conhecimento e pelo desejo de contribuir para a transformação da realidade.

Este trabalho foi realizado com o apoio das seguintes agências de pesquisa brasileiras: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) e Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

"Da soberba só provém a contenda, mas com os que se aconselham se acha a sabedoria" (Provérbios 13:10)

RESUMO

MENDONÇA, Ronan Dutra, D.Sc., Universidade Federal de Viçosa, setembro de 2025. **ArchChain: Proposta de um Token de Cessão para Avaliação de Custo e Desempenho em Redes Blockchain**. Orientador: Jose Augusto Miranda Nacif. Coorientador: Alex Borges Vieira.

Em meio ao crescente uso de Blockchain em aplicações públicas e privadas, uma lacuna persiste na análise sistemática da infraestrutura necessária para a participação eficiente em tais redes, sobretudo no que diz respeito aos nós de acesso a rede. O objetivo é criar um método que permita medir o impacto do uso de endpoints próprios em redes Blockchain. Este trabalho apresenta uma abordagem com a proposta de uma metodologia de avaliação de custo e desempenho para infraestrutura de redes Blockchain. A metodologia criada permite identificar o melhor compromisso entre custo e desempenho computacional na operação de um nó Blockchain em resposta a operações realizadas por uma aplicação descentralizada (Dapp). A arquitetura experimental considerou tanto redes públicas quanto permissionadas, com expansão entre os dois modelos. Os resultados indicaram que infraestruturas computacionais de menor porte podem atender satisfatoriamente a requisitos de desempenho em redes públicas, desde que bem escalonadas. Outro ponto relevante foi a introdução de um modelo para cálculo do compromisso entre custo e desempenho, fornecendo parâmetros concretos para a escolha de configurações ótimas. Os experimentos demonstraram que é viável manter a estabilidade de custo por transação em redes públicas através do ajuste da capacidade computacional. Também foi avaliada a interoperabilidade entre Blockchain, destacando o compromisso em tempo e custo, a partir da aplicação do Token de Cessão e da comparação entre protocolos de comunicação \textit{cross-chain}. Entre as principais contribuições da pesquisa, destacam-se: (i) a formulação de uma metodologia de avaliação de nós Blockchain sob as métricas de custo e desempenho; (ii) a criação de um mecanismo padronizado de cessão temporária de propriedade via tokens; e (iii) a expansão da funcionalidade desses tokens para além de uma única rede, por meio da interoperabilidade. Tais avanços contribuem com subsídios para escolha e adoção de soluções blockchain com maior eficiência e previsibilidade de custos.

Palavras-chave: blockchain; interoperabilidade; cross-chain; token de cessão; avaliação de custo e desempenho

ABSTRACT

MENDONÇA, Ronan Dutra, D.Sc., Universidade Federal de Viçosa, September, 2025. **ArchChain: Cession Token Proposal for Cost and Performance Assessment in Blockchain Networks**. Adviser: Jose Augusto Miranda Nacif. Co-adviser: Alex Borges Vieira.

Amid the growing use of blockchain in public and private applications, a gap remains in the systematic analysis of the infrastructure required for efficient participation in such networks, particularly with regard to access nodes endpoints. The goal is to create a method that allows the measurement of the impact of using proprietary endpoints in blockchain networks. This work presents an approach that proposes a methodology for evaluating cost and performance in blockchain infrastructure. The methodology developed enables the identification of the best trade-off between cost and computational performance in the operation of a blockchain node in response to transactions issued by a decentralized application (DApp). The experimental architecture considered both public and permissioned networks, with a comparative evaluation between the two models. The results indicated that smaller-scale computing infrastructures can satisfactorily meet performance requirements in public networks, provided they are properly scaled. Another relevant point was the introduction of a model for calculating the trade-off between cost and performance, providing concrete parameters for the selection of optimal configurations. The experiments demonstrated that it is feasible to maintain stable transaction costs in public networks through appropriate scaling of computational capacity. Interoperability between blockchains was also evaluated, highlighting the trade-offs in time and cost through the implementation of the Cession Token, as well as through a comparison between cross-chain communication protocols. Among the main contributions of this research are: (i) the formulation of a methodology for evaluating blockchain nodes based on cost and performance metrics; (ii) the creation of a standardized mechanism for temporary property cession via tokens; and (iii) the expansion of the functionality of these tokens beyond a single network through interoperability. These advances provide valuable input for selecting and adopting blockchain solutions with greater efficiency and cost predictability.

Keywords: blockchain; interoperability; cross-chain; cession token; cost and performance assessment

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1

Introdução

A expansão da oferta de dados em plataformas digitais públicas tem impulsionado diversas pesquisas voltadas à administração e ao processamento dessas informações. Os recursos tecnológicos existentes são capazes de oferecer controle, gerenciamento e armazenamento de dados de forma centralizada, tais como informações pessoais, registros médicos, produção e rastreabilidade de suprimentos, seguridade, financeiro e profissional (Farahani et al., 2018). Entretanto, a integridade e a privacidade dos dados constituem preocupações essenciais para assegurar a responsabilidade no tratamento das informações. Em conformidade com as normativas regulatórias, destinadas à proteção da privacidade dos cidadãos e à promoção da responsabilidade no uso de dados pessoais, é um direito que cada indivíduo possua autonomia para gerir e controlar o acesso às suas informações (Vinodhini and Kavitha, 2021). Como exemplo dessas normativas e leis regulatórias temos a *General Data Protection Regulation* (GDPR)[1] e Lei Geral de Proteção de Dados Pessoais (LGPD)[2] que determina o próprio indivíduo como proprietário de seus dados e lhe dá direitos de escolha sobre seu uso.

Nesse sentido, a tecnologia Blockchain repercute como uma solução promissora para o armazenamento e o tratamento de dados, mesmo quando a interação e o compartilhamento ocorrem entre usuários desconhecidos e em ambientes digitais de acesso público. A Blockchain tem como principal proposta manter os dados de forma segura, imutável, íntegra e distribuída. A tecnologia Blockchain foi inicialmente proposta pelo trabalho publicado por Nakamoto et al. (2008). Esse trabalho propôs uma junção de tecnologias, com finalidades iniciais estritamente financeiras, com aplicações para as criptomoedas, e a partir dessa possibilitou a expansão para diversas áreas de pesquisa.

Blockchain é uma tecnologia disruptiva com impactos nas relações entre pessoas, consumo e produção de bens e serviços (Xu et al., 2019). Isso se tornou possível a partir da evolução e unificação de outras tecnologias como criptografia assimétrica e protocolos de consenso distribuído via comunicação par a par, que são a essência de Blockchains (Pilkington, 2016). A junção dessas tecnologias possibilita o registro seguro

[1]GDPR - *https://gdpr-info.eu*
[2]LGPD - *https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm*

e descentralizado de dados ou transações entre entidades (pessoas e/ou organizações) que podem não se conhecer e assim não ter confiança mútua. Logo, os dados e transações entre essas entidades são registrados de forma imutável, com acesso público ou privado para fins de verificação de autenticidade e derivação de novas transações. É possível correlacionar Blockchain a uma tecnologia similar a um sistema de banco de dados distribuído, composto por inúmeros participantes, e que não há necessidade de controle centralizado. Porém, ela se difere dos bancos de dados nas condições de registro e armazenamento dos dados que são impostos e mantidos por um consenso.

Dentre as diversas áreas de pesquisa relacionadas ao desenvolvimento das Blockchains, existe um crescente interesse por novas aplicações no meio corporativo e nos serviços públicos, que abrangem, além das já bastante conhecidas aplicações para criptomoedas, como o Bitcoin e Ethereum (Nakamoto, 2008; Wood, 2014). Estas aplicações englobam um conjunto diversificado de cenários como a validação de dados e propriedade, integridade e interoperabilidade de dados. Isso implica diretamente nos esforços que estão sendo despendidos em adicionar novos recursos a esta tecnologia, gerando novas possibilidades de aplicações.

Dos recursos já oferecidos pela evolução das plataformas blockchain, os contratos inteligentes se destacam por estenderem a diferentes domínios em aplicações pessoais e corporativas (Xu et al., 2019). Os contratos inteligentes (*Smart Contracts*) são compostos por regras executáveis e auto-impositivas que funcionam de acordo com condições previamente acordadas e realizam determinadas operações dentro da Blockchain (Hewa et al., 2021). Esses contratos inteligentes oferecem recursos para soluções em diversos domínios. A solução de representação de ativos em tokens se destaca como um dos principais empregos dos contratos. Um token pode significar um código que representa algo. Ele também pode ser atribuído a dispositivos ou sistemas que geram codificações de acesso e autenticação. No universo Blockchain, o token representa um ativo inteiramente digital ou do mundo real, no qual um uso de aplicações específicas deste domínio visa agregar forma de garantia de posse e/ou representação de bens. Os bens podem ser tanto objetos físicos como digitais e que comportam utilidade variada, podendo ser transferidos, trocados, vendidos, alugados ou cedidos (Fairfield, 2021).

Contudo, a tecnologia Blockchain encontra-se ainda em pleno desenvolvimento e requer avanços em vários pontos, como os estruturais, tecnológicos e de governança. Estes avanços permitirão cada vez mais que sejam expandidas e utilizadas em larga escala. Embora as plataformas Blockchain possam apresentar soluções interessantes, são necessárias ainda ferramentas que permitam de maneira padronizada a integração entre redes distintas, ocasionando a interoperabilidade de ativos. Da mesma forma, a construção de uma forma padronizada e acessível de mensurar e gerenciar os custos dos recursos computacionais (i.e., infraestrutura necessária para prover acesso às redes)

permitirá o aumento da adoção por organizações em setores como indústria, serviços e governos.

No contexto da interoperabilidade, cabe interpretar as demandas de aplicações e apresentar soluções para prover a comunicação efetiva entre plataformas de blockchain distintas. Uma vez que um dado qualquer está alocado em uma determinada rede Blockchain, é de interesse de um grande número de aplicações desenvolver recursos com o propósito de estabelecer o fornecimento de acesso a esse dado em qualquer plataforma desejada. Neste contexto, a integração de soluções uniformes e individuais aos tipos padronizados de tokens estabelece um avanço significativo para as aplicações que almejam prover acesso a dados em mais plataformas.

Já em relação à infraestrutura, entender as características dos recursos computacionais necessários para implantação e o funcionamento de uma rede Blockchain é essencial para orientar o corpo técnico e executivo das organizações a planejarem uma possível adoção dessa tecnologia. Esses atores necessitam avaliar os modelos de rede Blockchain, e o problema em questão é entender requisitos não funcionais essenciais de cada modelo, em especial aspectos de custo e desempenho, assim como o compromisso entre ambos para planejar adequadamente as aplicações que funcionarão no topo da rede Blockchain. Para uma aplicação fazer uso da Blockchain é requerido acesso por meio de um nó pertencente à rede, seja por uma infraestrutura própria ou terceirizada. A aplicação propõe transações através de requisições dos seus usuários aos nós mantenedores da rede após estabelecer o acesso a algum nó. Cada um destes nós provedores é chamado de Ponto de extremidade, Ponto de acesso, *Gateway* ou ainda *Endpoint*. Geralmente, esses *Gateways* são recursos com poder computacional para prover processamento e são interligados a outros dispositivos utilizando redes de acesso público ou privadas, como por exemplo acontece com a Internet. Logo, o custo da utilização da Blockchain por uma aplicação consiste, além dos custos com a moeda da plataforma, no valor aplicado em infraestrutura capaz de prover acesso aos recursos da Blockchain.

As plataformas Blockchains executam diferentes conjuntos de transações por meio de implementações e protocolos distintos e normalmente isolados. Devido a esta heterogeneidade, há uma grande dificuldade em compartilhar informações através das diferentes plataformas. Essa incompatibilidade tem impulsionado o desenvolvimento de tecnologias capazes de promover a comunicação segura e eficiente entre diferentes redes descentralizadas. Esse conceito se refere à capacidade de sistemas independentes operarem de forma conjunta, permitindo a troca de informações e ativos digitais sem barreiras estruturais. Portanto, a interoperabilidade é definida como a capacidade de haver cooperação entre partes envolvidas em uma solução, ainda que utilizem diferentes tecnologias (Wegner, 1996). A tecnologia *cross-chain* é o envolvimento de um par de blockchains em que um tipo de aplicação descentralizada facilita a transferência de

ativos de uma blockchain para outra, promovendo a interoperabilidade. Isso significa que é possível mover ativos, como criptomoedas e tokens, de uma blockchain para outra sem a necessidade de intermediários centralizados.

A maioria das propostas da literatura que lidam com a questão de custos o faz com foco apenas nos custos de transações e desempenho da aplicação específica em redes públicas (Leal et al., 2020; Rouhani and Deters, 2017a; Zhang et al., 2020) e outras em redes permissionadas (Baliga et al., 2018; Thakkar et al., 2018; Wang and Chu, 2020; Xu et al., 2021). Alguns trabalhos ainda focam na análise de uma aplicação comparando ambas as redes (Monrat et al., 2020; Malik et al., 2019). Contudo, nenhuma dessas propostas busca modelar e identificar a infraestrutura, especificamente a arquitetura do *Endpoint*, considerando ao mesmo tempo os fatores de desempenho e custo para uma organização participar como membro de uma rede Blockchain.

A literatura sobre avaliação de desempenho de redes Blockchain tem focado em métricas como transações por segundo (TPS) e latência em plataformas isoladas. No entanto, não foram encontrados estudos que avaliam o comportamento dessas redes sob uma metodologia específica para avaliação da infraestrutura necessária para prover acesso às plataformas, tampouco avaliam os custos dispendidos para a participação por meio de um nó *Endpoint* com infraestrutura própria. O cenário que propõe a utilização de uma infraestrutura própria é comum e indicado pelas plataformas Blockchain. A participação com um nó da rede garante mais autonomia e a descentralização desejada. O diferencial deste trabalho em relação aos demais citados consiste na criação de uma metodologia que estabelece referência para avaliação de nós provedores de acesso às plataformas Blockchain. Por meio dessa metodologia, é possível realizar experimentos padronizados com a submissão de transações a diferentes infraestruturas e plataformas. Os experimentos visam analisar o custo, a resiliência e o desempenho da infraestrutura sob condições de estresse e elevada demanda. Como resultado dos experimentos, a metodologia oferece um panorama sobre a viabilidade prática do uso de nós *Endpoints* próprios.

Este trabalho investiga o impacto da escolha da infraestrutura no custo e desempenho de aplicações descentralizadas. O principal resultado demonstra que, para as redes públicas, o custo por transação pode ser mantido relativamente estável através do escalonamento do poder computacional dos nós em resposta ao aumento da carga de trabalho. Para fundamentar esta análise, foi desenvolvida, como estudo de caso, uma aplicação descentralizada para arrendamento de tokens temporários, denominada Token de Cessão Temporária (CST), incluindo uma extensão para interoperabilidade, o Token de Cessão Temporária Multicadeia (CST_{mc}). Por fim, a pesquisa apresentou métodos para alcançar o custo monetário da infraestrutura necessária para atingir a vazão máxima de transações (TPS) sob diferentes cargas. A validação funcional dos contratos utilizados para os tokens obteve sucesso integral, e a análise de escalabilidade e custo

forneceu uma base empírica para a recomendação de uma infraestrutura ótima, equilibrando custo e desempenho.

1.1 Apresentação do Problema

Um desafio relacionado à Blockchain é quanto à revogação ou anulação de direitos de posse de ativos já concedidos. Após os dados serem registrados em transações e haver o estabelecimento de conexão entre blocos de dados, as inserções não serão mais revertidas. A partir deste desafio surge o problema de que, dada uma permissão por meio de um valor inserido na Blockchain, como revogar essa permissão e ao mesmo tempo manter um histórico para que possa ser realizada a rastreabilidade dos ativos que já foram concedidos.

Outro desafio trata ainda da interoperabilidade de ativos entre Blockchains, uma vez que existem várias plataformas e que não há um meio de comunicação padronizado entre elas. Muitas destas plataformas utilizam variados tipos de tecnologias e protocolos de consenso diferentes, dificultando a interoperabilidade. Embora exista esta dificuldade, é importante superar este obstáculo para que as transações possam ser realizadas e verificadas em múltiplas plataformas.

A partir da definição e avaliação da infraestrutura para uso da tecnologia Blockchain, modelos que permitam analisar benefícios e custo das infraestruturas computacionais necessárias para implantação e o funcionamento de uma aplicação Blockchain são essenciais para orientar o corpo técnico e executivo das organizações a planejarem uma possível adoção da tecnologia Blockchain. Esses atores necessitam avaliar os modelos de opções de rede pública ou permissionada. As plataformas de Blockchain públicas foram as primeiras a serem desenvolvidas e são ainda as mais utilizadas. Estas plataformas permitem o desenvolvimento e execução de contratos inteligentes, sem restrição de acesso ou uso desses recursos e constituem um intricado ecossistema de aplicações. Por sua vez, uma rede Blockchain permissionada (Androulaki and et al., 2018) é uma alternativa atrativa para organizações que possuem infraestrutura e corpo técnico próprios, visando escapar de questões de custos (tarifas) e desempenho instável das plataformas Blockchain públicas. O problema em questão é entender o impacto desses dois modelos no consumo de recursos computacionais e, por conseguinte, identificar a infraestrutura com melhor compromisso entre desempenho e custo para a aplicação Blockchain.

O presente trabalho trata de analisar os desafios de mapeamento dos custos necessários para manter uma infraestrutura participante de uma rede Blockchain e os avanços atuais entre os cenários que trabalham com a interoperabilidade de tokens e a tecnologia de Blockchain. Para tanto, o problema da pesquisa inclui a análise sobre as formas de utilização da tecnologia Blockchain, bem como as implicações provenientes dos requisitos de custo e recursos computacionais de infraestrutura desses cenários.

1.2 Hipóteses

As hipóteses para esta pesquisa são as seguintes:

- Uma infraestrutura computacional básica, com baixos custos e poucos recursos computacionais, é capaz de participar como um nó em uma rede Blockchain pública, baseada em padrões Ethereum, permitindo:
  - O uso de tokens para o arrendamento e validação de propriedade temporária;
  - Um meio com escalabilidade e disponibilidade para uso de tokens temporários de forma correlacionada a outras aplicações externas;
  - A interoperabilidade da propriedade temporária de tokens entre Blockchains.

1.3 Objetivos

Objetivo Geral

O objetivo geral deste trabalho é investigar a utilização de Blockchains nos aspectos de levantamento de custo de infraestrutura e verificação de propriedade e interoperabilidade de tokens disponibilizados na rede. Assim, o trabalho realiza um direcionamento de pesquisa e propõe um método para avaliação de custo monetário e desempenho no uso da infraestrutura computacional por tecnologia Blockchain. O intuito é tratar e analisar as possíveis implicações entre o consumo de recursos da infraestrutura pela tecnologia Blockchain e suas implicações no controle de verificação de propriedade de tokens, considerando suas características de imutabilidade e interoperabilidade.

Objetivos Específicos

Os objetivos específicos que envolvem esta pesquisa são detalhados a seguir:

1. Infraestrutura e Requisitos de Blockchain
   - Identificar parâmetros e definir requisitos necessários ao desenvolvimento de um método que contemple soluções de levantamento de custo e desempenho de infraestrutura Blockchain;
   - Delinear um padrão de infraestrutura para a análise de custos e desempenho;
   - Avaliar os custos de infraestrutura para provimento de aplicações Blockchain;
   - Avaliar os custos de utilização do ambiente proposto em plataformas Blockchain.

2. Metodologia e Validação Experimental

    - Produzir cenários de experimentação da metodologia desenvolvida;
    - Avaliar a possibilidade de expansão da metodologia desenvolvida;
    - Disponibilizar protótipos de prova de conceitos que possam contribuir para o entendimento do ambiente e token propostos;

3. Interoperabilidade e Gestão de Token

    - Identificar quais requisitos para a criação, cessão e controle de um token específico para validação externa e temporária de propriedade;
    - Propor ambiente de interoperabilidade e controle do token de cessão temporária;
    - Desenvolver um ambiente que permita caracterizar a interoperabilidade do token de cessão temporária;

1.4 Contribuições

As contribuições deste trabalho são o desenvolvimento de um método para avaliação de infraestrutura, por meio de custo monetário e desempenho de recursos computacionais, necessários para prover acesso e participação junto a uma rede Blockchain pública por uma determinada aplicação. Em conjunto, foi planejado e construído um token com características de posse temporária, bem como conceitos e procedimentos para prover a interoperabilidade desse token, utilizando padrões de tokens dispostos na tecnologia Blockchain. À partir dessa contribuição geral, as seguintes contribuições e resultados de pesquisa alcançados são apresentados:

1. Criou-se uma estrutura padrão de infraestrutura computacional para redes Blockchains públicas e permissionadas, capaz de oferecer expansão para diferentes plataformas, com o intuito de fornecer meios de avaliação de recursos computacionais presentes em nó *Endpoint* Blockchain.

    Foi realizada uma avaliação das implementações de plataformas blockchain existentes. Essa avaliação permitiu a identificação dos modelos de arquitetura para direcionar as pesquisas ao redor do nó *Endpoint* da infraestrutura utilizada. Por meio dos direcionamentos realizados pela pesquisa, desenvolveu-se uma estrutura que permite oferecer uma arquitetura geral para identificação dos nós a serem avaliados. Espera-se que, a partir da divulgação dos resultados apresentados no Capítulo 5 e da publicação do trabalho de pesquisa (Mendonça et al., 2022a), seja estimulado o uso de nós próprios para participar de uma rede Blockchain. Dessa

forma, seja compreendida a necessidade de avaliação e utilização de nós próprios pertencentes à plataforma Blockchain utilizada no contexto de posse de tokens e validações de dados por aplicações de diversos setores.

As seguintes perguntas são respondidas com esses resultados apresentados: Qual a importância de manter um nó *Endpoint* em redes Blockchain? É possível avaliar o custo e o desempenho dos recursos computacionais desse nó?

2. Avaliação do custo e desempenho de infraestrutura para provimento e utilização de aplicações em plataformas Blockchain.

   Buscou-se apresentar uma análise de utilização da aplicação, proposta como solução para a validação de dados e tokens, visando identificar a infraestrutura que leva ao melhor benefício, considerando ao mesmo tempo os fatores de custo e desempenho para padrão de redes públicas. Foi apresentada uma avaliação para representar as características que interferem nos custos da infraestrutura por transação confirmada na Blockchain. A avaliação apresentada na Sessão 5.1.4 considerou a relação entre o custo monetário do recurso computacional necessário para executar a aplicação descentralizada e o desempenho, medido pela vazão máxima obtida por esse recurso em números de transações e publicada em Mendonça et al. (2023).

   Com estes resultados apresentados, são respondidas as seguintes perguntas: Quais elementos devem ser avaliados na escolha da infraestrutura utilizada por Blockchains? Qual o impacto da adoção de cada uma das infraestruturas Blockchain avaliadas em relação ao custo e ao desempenho?

3. Projeto de instrumento, por meio de padrões de tokens, para conceder um controle de propriedade temporária no uso específico de validação de tokens cedidos por terceiros.

   A pesquisa despertou a especificação e criação de instrumento para realizar a validação de posse, projetando e implantando um processo padronizado, por meio de tokens, visando identificar avanços e entraves na adesão aos padrões da tecnologia Blockchain. Para alcançar este resultado, foi desenvolvida uma solução que viabiliza a possibilidade de atribuição de posse de tokens a uma cessão apresentada na Sessão 4.2. A cessão ocorre por meio de transferência do token para um determinado endereço de usuário e determina um tempo em que a cessão será automaticamente revogada. Por meio de verificação de desempenho e aplicabilidade, foi possível responder aos questionamentos de que é possível escalar o controle dos tokens em um ambiente de blockchain pública. Os resultados foram divulgados e apresentados em Mendonça et al. (2022b).

   Com os resultados apresentados na Sessão 5.2.3, são respondidas as seguintes perguntas: Como ceder e revogar permissões em Blockchain de forma temporária?

Blockchain possibilita um meio com escalabilidade e disponibilidade para armazenar e controlar meios de controle de posse através de tokens temporários?

4. Ampliação dos requisitos do CST para torná-lo capaz de interoperar com a solução de concessão de tokens temporários entre blockchains distintas.

   Em Mendonça et al. (2024), comparamos os protocolos de interoperabilidade Notarial e Bloqueio de Hash. A comparação focou no monitoramento das métricas de desempenho e custos dos protocolos em todas as fases do processo para interoperar um token. Avaliamos o impacto para prover a interoperabilidade de tokens com o intuito de fornecer a possibilidade de verificação da propriedade de tokens a partir de redes blockchain distintas. Expandimos a comparação com o protocolo da Chainlink (Chainlink, 2023), que é provido via infraestrutura proprietária, popularmente utilizado atualmente. Conduzimos experimentos com várias operações entre redes distintas e, por meio dessa comparação, ficou evidenciado o amplo espaço para inovação em interoperabilidade, já que o protocolo Chainlink obteve resultados de tempos e custos maiores que os dos outros protocolos (Muniz et al., 2025).

   Com estes resultados apresentados, respondemos às seguintes perguntas: Qual o impacto de prover a interoperabilidade entre Blockchains para permitir a validação de propriedade utilizando tokens temporários?

1.5 Organização do Trabalho

O restante do texto está organizado da seguinte forma:

O capítulo 2 relaciona os temas centrais envolvidos na formulação da pesquisa, como as Blockchains, contratos inteligentes, rastreabilidade, tokens e a interoperabilidade. O capítulo 3 dedica-se também a promover um levantamento bibliográfico acerca do estado da arte sobre os temas envolvidos, como avaliação de desempenho, custo e interoperabilidade associados à tecnologia Blockchain. O capítulo 4 apresenta um modelo arquitetural geral de nós proposto para a solução do problema de avaliação e análise de custo e desempenho de infraestrutura pública para Blockchain. É descrito o papel de cada um dos componentes da arquitetura e é apresentado o formato de cálculo do compromisso entre custo e desempenho por nó em função da carga de trabalho. Esse capítulo também apresenta a abordagem do token de cessão, projetado como parte da solução de validação de propriedade de forma temporária, e a sua interoperabilidade. No capítulo 5 são descritos os procedimentos utilizados nos experimentos realizados e os resultados obtidos. Finalmente, as considerações finais são apresentadas no capítulo 6. A última seção deste documento apresenta as referências bibliográficas consultadas durante as fases de desenvolvimento desta pesquisa. Nos apêndices A, B, C, D e E são

apresentadas as contribuições em publicações referentes ao tema de tese, dados em tabelas de resultados de medições, códigos e descrições citadas e referenciados ao longo do texto, as codificações de testes dos contratos inteligentes e amostras de tabelas de dados capturados dos experimentos, respectivamente.

Capítulo 2

Fundamentos

Este capítulo descreve os principais conceitos relacionados às redes Blockchains, a qual é a área de pesquisa relacionada a esta tese. Estes conceitos servirão como referencial para o entendimento de que esta pesquisa está constituída. Inicialmente, será apresentado o conceito de Sistemas Distribuídos, além dos temas que se relacionam com as questões de tolerância a falhas e protocolos de consenso em redes de blockchains. Em seguida será introduzida a definição de Blockchain, bem como a utilização dos meios tecnológicos para manter esta rede, como Contratos Inteligentes, Carteiras e tokens. Por meio desses meios tecnológicos, o capítulo procura demonstrar a importância dos tokens para aplicações blockchain como um aspecto-chave para a implementação de aplicações fundamentais dessas redes. Por fim, o capítulo apresenta trabalhos relacionados aos temas de pesquisa.

2.1 Sistemas Distribuídos

Os sistemas distribuídos consistem em um conjunto de computadores, denominados nós, independentes que se denominam funcionando como um sistema único. Esses sistemas são projetados para compartilhar recursos, executar tarefas de forma colaborativa e oferecer maior eficiência, escalabilidade e disponibilidade. Os nós são espalhados por diferentes locais, comunicando-se por meio de uma rede para realizar tarefas. Esses sistemas são usados em diversas aplicações, como computação em nuvem, bancos de dados distribuídos e redes Blockchain (Tanenbaum and van Steen, 2002).

As principais características dos Sistemas Distribuídos são a transparência, escalabilidade e tolerância a falhas. Os sistemas distribuídos buscam ocultar as complexidades de sua distribuição, para que os usuários interajam com o sistema como se fosse único. Isso inclui transparência de acesso, localização e replicação. Um dos principais benefícios dos sistemas distribuídos é escalabilidade. A escalabilidade é a capacidade de lidar com o aumento do número de usuários, demandas por transações ou volume de dados. A tolerância a falhas é um quesito primordial dos sistemas distribuídos que são projetados para continuar seu funcionamento mesmo que parte do sistema falhe, sendo essencial em aplicações de nível crítico.

Como uma das vantagens dos sistemas distribuídos tem-se a melhora no desempenho

ao permitir o uso paralelo de múltiplos processadores, acelerando as operações e melhorando a eficiência. A redundância e alta disponibilidade também são vantagens dos sistemas distribuídos. Onde, uma vez que os dados são replicados em diferentes locais, o sistema pode permanecer em funcionamento mesmo em caso de falhas. A flexibilidade e escalabilidade com que os sistemas distribuídos são projetados facilitam o incremento de novos recursos sem interrompimento dos serviços.

Apesar dos vários benefícios e vantagens que os sistemas distribuídos trazem com suas características tecnológicas, existem ainda vários desafios a serem superados. Dentre esses desafios encontram-se a sincronização, segurança, complexidade e interoperabilidade. Então, para que as diversas aplicações dos sistemas distribuídos possam usufruir de todos os seus benefícios, devem implementar e garantir a sincronização correta de diferentes nós, gerenciar e proteger os dados, mantendo a comunicação e interoperabilidade em ambientes amplamente distribuídos. Por fim, é possível encontrar vários exemplos de sistemas distribuídos que aplicam de forma eficiente os conceitos como a computação em nuvem, redes *peer-to-peer* e aplicações baseadas em blockchain. Esses exemplos são fundamentais para muitas das tecnologias que são utilizadas atualmente, desde serviços de transmissão de vídeos, plataformas de comércio eletrônico até as criptomoedas.

2.1.1 Tolerância a falhas e Protocolos de Consenso

A tolerância à falha é um conceito fundamental em sistemas distribuídos. Refere-se à capacidade de um sistema continuar funcionando corretamente, mesmo quando ocorrem falhas em alguns de seus componentes. A tolerância é alcançada por meio de estratégias como redundância e protocolos de consenso que mitigam os impactos de falhas. Os sistemas com tolerância a falhas devem garantir a alta disponibilidade de serviços, mesmo em casos de falhas de hardware ou software (Lamport et al., 2019).

Os protocolos ou algoritmos de consenso são mecanismos extremamente necessários em alguns sistemas tolerantes a falhas. Eles permitem que os nós de um sistema concordem sobre um único valor ou estado, mesmo em cenários de falhas ou comunicações não confiáveis. Esses algoritmos são essenciais para garantir a consistência e confiabilidade em sistemas onde múltiplos agentes trabalham de forma colaborativa e independente, como as redes blockchains. Nesse caso, os nós de uma rede podem falhar ou apresentar comportamentos inconsistentes e, para que o sistema opere como uma unidade coesa, os nós precisam concordar em decisões, como a ordem de processamento de transações.

Algoritmos de consenso, em geral, trabalham com as propriedades: Acordo, onde todos os nós honestos devem decidir sobre o mesmo valor; Validade, onde o valor decidido deve ser um valor válido dentre os valores iniciais propostos; Término, onde todos os nós honestos chegam a uma mesma decisão. Lamport (2001) propôs o algoritmo nomeado

Paxos. Ele é um algoritmo amplamente conhecido e baseia-se em três agentes principais: os propositores, aceitadores e aprendizes. Teoricamente é bastante eficiente, mas com uma grande complexidade prática. O algoritmo de consenso *Raft* é uma alternativa ao Paxos. Ele é conhecido por ser mais compreensível e de fácil implementação prática. Ele utiliza um líder eleito para coordenar as operações, garantindo a consistência entre os nós (Huang et al., 2020).

O algoritmo *Byzantine Fault Tolerance* (BFT) foi projetado para sistemas onde alguns nós podem agir de forma maliciosa dentro da rede. Essas ações dos nós são conhecidas como falhas bizantinas (Lamport et al., 2019). Um exemplo de implementação desse protocolo é o *Practical Byzantine Fault Tolerance* (PBFT), usado em blockchains como Hyperledger Fabric. Esses algoritmos são usados principalmente em transações distribuídas que requerem a garantia de que todas as partes do sistema concordem em concluir ou reverter uma transação.

As plataformas blockchain públicas empregam algoritmos de consenso como Proof of Work (PoW) e Proof of Stake (PoS) para alcançar consenso em redes distribuídas e abertas. Os desafios enfrentados por esses algoritmos de consenso são latência, falhas bizantinas e escalabilidade. A latência é afetada porque a comunicação entre nós pode ser lenta, especialmente em redes geograficamente distribuídas. As falhas bizantinas ocorrem quando há o ingresso de nós maliciosos ou comprometidos que tornam o consenso ainda mais desafiador. Já a escalabilidade é impactada por se tratar de um sistema complexo e custoso que deve estar empenhado em garantir o consenso em redes com milhares de nós.

Os algoritmos de consenso são os fundamentos principais em áreas que utilizam os sistemas distribuídos como a computação em nuvem e blockchains. A implementação dos algoritmos de BFT apresenta desafios especialmente ligados à complexidade dos cenários em que operaram. Dentre os desafios, têm-se a comunicação entre nós, falhas, latência, segurança, escalabilidade, consistência e disponibilidade. Em sistemas distribuídos com muitos nós, a comunicação necessária para que todos os nós troquem mensagens pode crescer exponencialmente. Isso torna os algoritmos de BFT difíceis de escalar, especialmente em redes grandes como blockchains.

A comunicação em larga escala pode ocasionar uma sobrecarga, impactando a eficiência do sistema. Além disso, os algoritmos precisam lidar com nós que podem agir de forma maliciosa ou imprevisível. Essa ação, chamada de falha bizantina, insere um comportamento ilegítimo, que causa desde o envio de informações incorretas até a negação de participação no processo de consenso. A complexidade dessa falha dificulta a detecção e redução dessas ações maliciosas. Um sistema pode enfrentar ataques em cenários reais, como a corrupção de um grande número de nós. Os algoritmos BFT precisam garantir a segurança mesmo se 1/3 dos nós apresentarem falhas bizantinas.

A tolerância à latência é uma característica importante dos algoritmos de consenso. A latência dificulta o processo, gerando atrasos ou perda de mensagens que podem levar

a inconsistências do serviço, em especial nos sistemas geograficamente distribuídos. Portanto, proteger a comunicação entre os nós é essencial para evitar ataques, como a falsificação de mensagens. Adicionar criptografia aumenta a segurança, porém aumenta a complexidade computacional, causando maior latência. O compromisso com a escalabilidade vai de encontro à limitação de vários algoritmos BFT, que não escalam com facilidade em redes com milhares de nós, pois sobrecarregam rapidamente a rede com um crescente número de mensagens. Algoritmos atuais tentam lidar com a escalabilidade, porém há um compromisso entre a escalabilidade, a eficiência e a segurança que contrapõe a conquista da solução dessa limitação. Portanto, atingir o equilíbrio entre a consistência, disponibilidade e particionamento tolerante a falhas é um grande desafio aos algoritmos de consenso. Algoritmos BFT têm sido aplicados em diversos contextos práticos, como blockchains, e apesar desses desafios, eles são essenciais em sistemas distribuídos.

2.2 Blockchain

Blockchain é uma tecnologia que armazena registros de transações de forma distribuída, composta por inúmeros participantes, e não há necessidade de um controle centralizado. O termo Blockchain foi inserido inicialmente por Nakamoto et al. (2008), que o implementou para servir como livro-razão da sua moeda virtual. O objetivo da implementação era criar uma tecnologia inviolável para evitar que uma certa quantidade de moeda fosse gasta duas vezes em situações distintas. Desde então, uma imensa variação deste conceito vem sendo desenvolvida e muitas outras ainda estão sendo idealizadas. Cada nó conectado à rede de uma Blockchain contém uma cópia completa destes dados e realiza operações de validar e transmitir as transações aos demais nós. A característica mais marcante é que uma vez que uma informação é registrada, não poderá sofrer alterações, o que resulta em confiabilidade e permite uma fácil auditoria.

A aplicabilidade da tecnologia Blockchain é diversificada e facilita o suporte para inúmeras soluções de problemas como a validação de acesso, integridade, registros de operações financeiras e interoperabilidade de dados. Em relação a estas aplicações, a Blockchain oferece maior transparência, rastreabilidade e segurança no registro de suas transações (Gordon and Catalini, 2018). Uma vez que a Blockchain é aplicada como parte da solução de uma aplicação, esta leva o nome de uma aplicação descentralizada (Decentralized applications - dApps). As dApps geralmente interagem com componentes internos e externos às blockchains de acordo com seus requisitos. Os componentes que executam transações diretamente na Blockchain são chamados de ***on-chain***. Por esta razão, os componentes on-chain oferecem maior transparência e segurança nas transações. Já os componentes que executam transações fora da Blockchain são denominados ***off-chain***. Estes, por sua vez, são menos transparentes e mais suscetíveis a vulnerabilidades.

As informações ficam organizadas em blocos encadeados por meio de *hashes* criptográficos. Esses blocos são uma estrutura para inclusão de dados na blockchain e contêm cabeçalho e uma lista de transações conforme ilustra a Figura 2.1 (Nakamoto et al., 2008). O primeiro bloco criado é denominado "*genesis block*". Cada bloco é identificado por um *hash* criptográfico estampado no cabeçalho do bloco e faz referência a um bloco anterior por meio do campo "*previous block hash*". No cabeçalho do bloco, ainda há um conjunto de metadados que consiste em "*timestamp*" e "*nonce*", e outro conjunto de metadados consiste na "*Merkle tree root*". O "*timestamp*" e o "*nonce*" estão relacionados ao processo de mineração. A "*Merkle tree root*" é uma estrutura de dados usada para organizar todas as transações no bloco.

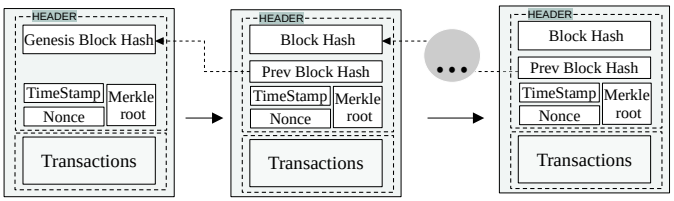


Figura 2.1: Cadeia de blocos. Fonte: elaborado pelo autor

Os tipos de plataformas blockchain são classificados basicamente de acordo com a forma de acesso e proteção das transações armazenadas. Elas são encontradas na forma pública, privada e de consórcio. As especificações de protocolo de consenso, proteção de acesso às informações e controle da forma de distribuição diferenciam os tipos de Blockchain e são apresentadas na Tabela 2.1.

Tabela 2.1: Tipos de Blockchains.

	Centralização	**Consenso**	**Acesso a Dados**
Pública	Nenhuma	Livre	Públicos
Privada	Totalmente	Restrito	Públicos ou restritos
Consórcio	Parcial	Restrito	Públicos ou restritos

Em uma blockchain privada, as respostas são mais rápidas e seguras, porém o controle é exercido por um proprietário específico e os nós precisam de permissão para ingressar na rede. A Blockchain privada é mais rápida, eficiente e segura.

Na Blockchain pública, por sua vez, a rede é totalmente descentralizada e pode conter vários nós e qualquer nó pode se inscrever na rede. Porém, apenas nós sincronizados são utilizados para consenso. Uma Blockchain de consórcio é composta por nós de organizações específicas que se unem e controlam quem pode ter acesso à rede. A rede resultante do consórcio é parcialmente descentralizada (Wang and Feng, 2018; Rouhani and Deters, 2017a).

2.2.1 Plataformas

As plataformas blockchain são fundamentais para a implementação de soluções descentralizadas e transparentes em diversos setores. Algumas plataformas que implementam soluções de blockchains públicas e permissionadas são: Ethereum, Hyperledger Fabric, Binance Smart Chain, Solana, Polkadot e Cardano. A tabela 2.2 apresenta uma comparação de algumas das características dessa plataformas:

Tabela 2.2: Plataformas Blockchains.

Plataforma	**Algoritmo de Consenso**	**Contrato Inteligente**	**Foco**	**Características e uso**
Ethereum	PoS	x	Pública	Criptomoeda amplamente utilizada
Hyperledger Fabric	Raft e outros	x	Permissionada	Aplicações em cadeias de suprimentos, finanças e saúde
Binance	PoS/PoA - PoSA	x	Pública	Ecossistema de finanças descentralizadas (DeFi)
Solana	PoH e PoS	x	Pública	Aplicativos que exigem alto desempenho, como jogos e NFTs
Polkadot	PoS	x	Pública	Interoperabilidade entre diferentes blockchains
Cardano	PoS	x	Pública	Aplicações em áreas como finanças, saúde, logística e até mesmo governança

Todas estas plataformas são ferramentas poderosas em seus próprios contextos. As plataformas Ethereum e Hyperledger são projetos blockchain amplamente usados, mas com propósitos, arquiteturas e públicos-alvo distintos. A plataforma Ethereum[1] implementa uma blockchain pública e descentralizada. Sua principal característica é o uso de contratos inteligentes, que são programas autônomos que podem executar ações predeterminadas de forma automática quando condições específicas são atendidas. O uso dos contratos inteligentes expande as barreiras das blockchains e dá suporte às aplicações descentralizadas (dApps). Ethereum utiliza uma máquina virtual Ethereum (EVM) com sua criptomoeda nativa, o Ether (ETH). Esta criptomoeda é utilizada para pagar transações, taxas de rede e como meio de incentivo para os participantes colaboradores da rede. Após a atualização para a versão Ethereum 2.0, a rede passou a utilizar o mecanismo de consenso Proof of Stake (PoS), que substituiu o anterior Proof of Work (PoW). Essa mudança reduziu o consumo de energia e aumentou a eficiência da rede, uma vez que não são mais necessários os recursos de processamento em resolução de cálculos para minerar um bloco. Existem aplicações importantes no ecossistema Ethereum como os protocolos de finanças descentralizadas, tokens não fungíveis, jogos baseados em blockchain e organizações autônomas descentralizadas (DAOs).

Os benefícios de uma plataforma de rede pública, como a Ethereum, são a descentralização, flexibilidade de soluções e abertura à comunidade de desenvolvimento. A descentralização proporciona o aumento da confiabilidade e da transparência da rede por não depender de um controle centralizado. Por meio de contribuições ao ecossistema Ethereum, os desenvolvedores aprimoram a plataforma e criam formas de compatibilidade com aplicações complexas e soluções individualizadas. Porém, existem

[1] https://ethereum.org/

ainda muitos desafios a serem superados pelas redes públicas. O alto volume de transações que são processadas pela plataforma Ethereum desafia a escalabilidade da plataforma por risco de comprometer a velocidade de processamento das transações. As taxas cobradas por transações também são elevadas durante os períodos em que a demanda aumenta. Portanto, a plataforma Ethereum é um exemplo de tecnologia blockchain que proporciona uma oportunidade de inovação em redes distribuídas.

A plataforma Hyperledger também é uma iniciativa de código aberto, porém voltada para o desenvolvimento de tecnologias blockchain permissionadas. Foi desenvolvida inicialmente pela Linux Foundation, em colaboração com a IBM e a Intel, e seu foco é atender às necessidades específicas de negócios e indústrias que requerem soluções personalizáveis e com controle de privacidade dos dados. Sua principal característica é permitir a criação de redes distribuídas e permissionadas, onde os participantes são conhecidos e confiáveis por meio de permissão e autenticação. Aos desenvolvedores são permitidas as opções de escolha de componentes como algoritmos de consenso, privacidade e modelos de identidade de acordo com a necessidade do projeto. Com o código acessível para uso e desenvolvimento, vários projetos são desenvolvidos de forma coletiva e colaborativa. Seus principais projetos são o Hyperledger Fabric, Besu e Indy. O Hyperledger Fabric [2] foi desenvolvido para criar redes de blockchains permissionadas. Conta com suporte a contratos inteligentes ou Chaincode, autenticação para privacidade de dados e ambiente propício à escalabilidade. O Hyperledger Besu é uma implementação compatível com a plataforma Ethereum que permite o seu uso tanto em redes públicas como privadas. O Hyperledger Indy tem seu foco em soluções para identidade descentralizada. Oferece controle de dados pessoais para autenticação e privacidade.

Uma comparação geral entre as principais tecnologias blockchain utilizadas atualmente, Ethereum e Hyperledger Fabric, pode ser observada na Tabela 2.3. A comparação leva a concluir que Ethereum é mais indicada para aplicações que exigem uma maior descentralização, transparência das transações e acesso público, como as DeFi e NFTs. Já a Hyperledger Fabric é mais apropriada para quem busca redes altamente controláveis, como cadeias de suprimentos.

2.2.2 Endpoints

As redes blockchain públicas são redes distribuídas de computadores, e esses computadores fazem parte dessa rede de forma independente e sem restrições. Esses computadores são cada um dos nós da rede que recebem as propostas de transações e/ou executam o software que verifica os dados das transações e os organiza em blocos. Portanto, existem dois tipos de nós que podem possuir funções separadas por diferentes

[2] https://hyperledger-fabric.readthedocs.io

Tabela 2.3: Comparação entre Ethereum e Hyperledger Fabric

Aspecto	Ethereum	Hyperledger Fabric
Propósito	Aplicativos descentralizados, contratos inteligentes e redes públicas.	Soluções empresariais, com redes permissionadas e personalizáveis.
Arquitetura	Rede pública. Aberta a todo usuário.	Rede permissionada. Participantes são conhecidos e confiáveis.
Consenso	Utiliza PoS eficiente em redes descentralizadas.	Permite escolha, como PBFT.
Moeda Nativa	Ether. Usado para pagar taxas de transação e como incentivo.	Geralmente não utiliza criptomoedas. Aplicada a negócios que não dependem.
Escalabilidade	Enfrenta desafios de escalabilidade em momentos de alta demanda.	Altamente escalável devido ao uso em redes permissionadas controladas.
Privacidade	Todas as transações são públicas e podem ser rastreadas.	Projetada para oferecer privacidade. Permite transações privadas.
Flexibilidade	Foco em contratos inteligentes e aplicativos descentralizados.	Possibilita personalizações específicas
Casos de Uso	Finanças descentralizadas, NFTs, jogos, governança descentralizada.	Cadeia de suprimentos, saúde, finanças, redes corporativas e aplicações industriais.

software. Esses tipos diferentes de nós que executam softwares são nomeados como "Clientes" e cada um desses Clientes é necessário para formar um nó da rede. Portanto, uma rede Blockchain é constituída a partir do instante em que os computadores instanciam os softwares destes clientes e se conectam aos outros. Para as redes blockchains públicas baseadas na plataforma Ethereum, os softwares clientes são apontados como o cliente de execução e o cliente de consenso. Esses softwares clientes devem verificar as condições e regras impostas pelo protocolo da rede e mantê-la segura. A Figura 2.2 apresenta um esboço de um nó cliente que executa simultaneamente os clientes de execução e consenso.

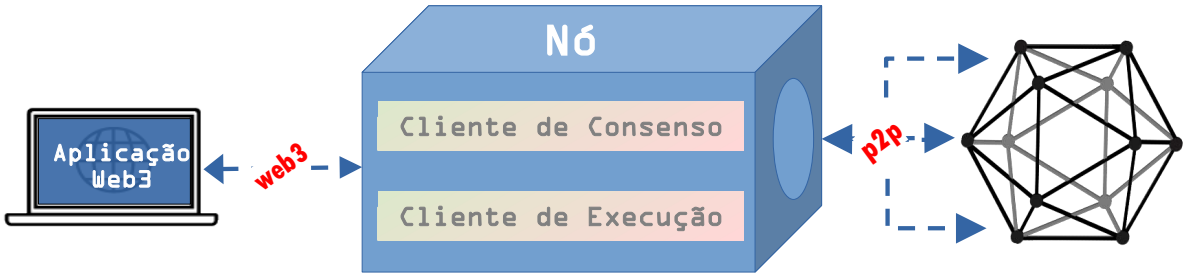


Figura 2.2: Nó. Fonte: elaborado pelo autor

O cliente de execução (também conhecido como Execution Engine, cliente EL ou anteriormente cliente Eth1) recebe novas transações transmitidas na rede, executa-as na EVM e mantém o estado mais recente e o banco de dados de todos os dados atuais do Ethereum. O cliente de consenso (também conhecido como Beacon Node, cliente CL ou anteriormente o cliente Eth2) implementa o algoritmo de consenso proof-of-stake, que permite que a rede alcance um acordo com base em dados validados pelo cliente de execução. Há também uma terceira parte do software, conhecida como "validador", que pode ser adicionada ao cliente de consenso, permitindo que um nó participe da proteção da rede.

Existem provedores de infraestrutura Blockchain que oferecem recursos para execução de um nó *Endpoint*. Esses provedores disponibilizam ferramentas como um serviço para facilitar o desenvolvimento de aplicações. Esses serviços são oferecidos por um determinado valor monetário e de acordo com a demanda requerida pela infraestrutura. Alguns desses provedores são ofertados pela AWS[3], Alchemy[4], QuickNode[5] e Infura[6]. Logo, o custo da aplicação consiste primordialmente no recurso computacional do nó *Endpoint*, ao passo que o usuário geralmente arca com a tarifação das transações. A vantagem desse tipo de oferta diz respeito à facilidade em contratar e utilizar os serviços de um *Endpoint* para o uso de submissão de transações por uma aplicação através de uma Interface de Programação de Aplicações (*Application Programming Interface (API)*. Em contrapartida, existe a desvantagem de ter cada vez mais a centralização da rede pela infraestrutura muitas vezes compartilhada. Em uma rede pública, o ideal seria que cada nó fosse executado por seu proprietário de forma direta, i.e., sem a disponibilização por terceiros. Desta forma, a rede se torna mais segura e descentralizada, permitindo que o processo de verificação de dados seja realizado por seu próprio nó, sem a necessidade de confiar em nós terceirizados.

Ao contrário dos nós *Endpoints* de uma rede pública, os nós de uma rede blockchain permissionada se tornam uma alternativa atrativa para organizações que visam escapar de questões de custos tarifários e desempenho instável das redes blockchain públicas como Ethereum e Bitcoin. O Hyperledger Fabric é uma das plataformas para blockchains permissionadas mais populares atualmente [7]. Conta com recursos para a implantação de uma infraestrutura de rede privada entre organizações e desenvolvimento de aplicações no topo dessa rede. Nesse caso, os participantes da rede formam um consórcio e arcam com o custo da infraestrutura, buscando ganhos no desempenho e custo tarifário em relação às redes blockchain públicas.

Nesse contexto, entender as características das infraestruturas computacionais para implantação e o funcionamento de uma rede blockchain é essencial para orientar o corpo técnico e executivo das organizações a planejarem uma possível adoção dessa tecnologia. Esses atores necessitam avaliar os modelos de rede pública ou permissionada e o problema em questão é entender requisitos não funcionais essenciais de cada modelo, em especial aspectos de custo e desempenho, assim como o compromisso entre ambos para planejar adequadamente as aplicações que funcionarão no topo da rede blockchain.

A maioria das propostas encontradas na literatura que lidam com a questão de desempenho de redes Blockchains foca em aplicações específicas para rede pública (Leal et al., 2020; Rouhani and Deters, 2017a; Zhang et al., 2020) ou rede

[3]https://docs.aws.amazon.com/blockchain-templates/
[4]https://www.alchemy.com/
[5]https://www.quicknode.com/
[6]https://infura.io
[7]https://www.ibm.com/topics/hyperledger

permissionada (Baliga et al., 2018; Thakkar et al., 2018; Wang and Chu, 2020; Xu et al., 2021). Alguns trabalhos ainda focam na análise de uma aplicação típica para ambas as redes (Monrat et al., 2020; Malik et al., 2019). Contudo, nenhuma dessas propostas busca identificar a infraestrutura, especificamente a arquitetura do nó *Endpoint* da rede, considerando ao mesmo tempo os fatores de desempenho e custo de infraestrutura para uma organização participar com um nó de uma rede blockchain.

2.2.3 Contratos Inteligentes

A utilização do conceito de contratos inteligentes pela tecnologia Blockchain aumentou significativamente as suas possibilidades de uso. Contratos Inteligentes (*Smart Contracts*) são programas autoexecutáveis e autoimpositivos que funcionam de acordo com condições previamente acordadas (Hewa et al., 2021). Os contratos são implementados em uma determinada linguagem de programação e de acordo com a plataforma por meio de scripts. As regras criadas para os contratos são armazenadas em uma plataforma e realizadas pela rede da forma como foram estabelecidas. Estes contratos são capazes de cumprir determinadas operações funcionando como aplicações descentralizadas. Entre os benefícios da utilização de contratos inteligentes apontados por (Hewa et al., 2021) estão a eliminação da necessidade de um terceiro confiável para efetuar transações, a integridade e a transparência das transações e a autonomia de execução e a precisão dos contratos, uma vez que são imutáveis e são executados quando as pré-condições são atendidas.

Na Figura 2.3 são apresentadas quatro etapas que envolvem o ciclo de vida de um contrato inteligente. Na primeira etapa são pré-definidas todas as condições que devem ser acordadas por todas as partes envolvidas, implementadas em uma linguagem de computadores específica para a plataforma Blockchain desejada e implantar o contrato pronto na Blockchain. Na segunda etapa, é esperado que os eventos que foram acordados ocorram, e a partir desta ocorrência, a execução do contrato é desencadeada. A execução ocorre na terceira etapa, em que os eventos acionam automaticamente as funções necessárias. Por fim, todos os acordos são executados com rapidez e eficiência sem a necessidade de intervenção por terceiros.

Devido ao caráter complementar que os contratos possuem em relação à Blockchain, eles se tornaram parte essencial das Blockchains que surgiram após o Bitcoin, proposto por (Nakamoto et al., 2008). Com a utilização desses contratos, as Blockchains tiveram sua capacidade de armazenamento aprimorada. Um contrato permite definir um comportamento para um determinado estado e atender necessidades de aplicações diversas. Nesse sentido, os contratos inteligentes são capazes de executar transações muito mais complexas dentro da Blockchain do que simplesmente a troca de moeda. Um exemplo simples de um contrato inteligente é apresentado pelo código da Listagem 2.1,

Figura 2.3: Ciclo de vida em um Contrato Inteligente. Fonte: elaborado pelo autor

onde na linha 1 é especificada a versão do compilador da linguagem *Solidity*. Na linha 2 é definido o nome do contrato chamado de "OlaMundo". Sendo que um contrato é uma coleção de funções e dados, uma vez implantado, o contrato recebe um endereço específico na blockchain para ser invocado. Na linha 3 é declarada uma variável de estado "mensagem" do tipo *string*. As variáveis de estado são variáveis cujos valores são armazenados permanentemente no armazenamento do contrato. A palavra-chave "public" torna as variáveis acessíveis de fora de um contrato e cria uma função que outros contratos ou aplicações clientes podem acessar o seu valor. Na linha 4 é criado um construtor para o contrato que é uma função especial e que só é executada na criação do contrato. Eles são usados para inicializar os dados do contrato. O construtor recebe um argumento do tipo *string* "mensagemInicial" e na linha 5 define o valor na variável de armazenamento "mensagem" do contrato. Por sua vez, a função pública "atualizar" definida na linha 7 recebe um argumento do tipo *string* e na linha 8 atualiza o valor da variável de armazenamento "mensagem".

```
1  pragma solidity 0.8.7;
2  contract OlaMundo {
3     string public mensagem;
4     constructor(string memory mensagemInicial) {
5        mensagem = mensagemInicial;
6     }
7     function atualizar(string memory novaMensagem) public {
8        mensagem = novaMensagem;
9     }
10 }
```

Listagem 2.1: Exemplo de Contrato Inteligente

A partir dessas características, inúmeras aplicações podem ser desenvolvidas baseadas no uso de contratos inteligentes como, por exemplo, aplicações financeiras (gerenciamento de moeda, serviço de garantia, procedimentos de auditoria, empréstimo), aplicações médicas (gestão de informações de saúde, proteção de dados de pesquisa clínica, monitoramento e tratamento automatizado de pacientes, gerenciamento de identidade e controle de acesso, proteção de dados de identidade), aplicações imobiliárias, aplicações de acordos contratuais, aplicações de internet das coisas, aplicações de serviços de telecomunicações, aplicações de gestão de logística, além de aplicações entre diferentes indústrias (Hewa et al., 2021).

2.2.4 Carteiras

As carteiras são ferramentas digitais utilizadas para assinar, armazenar, enviar e receber transações realizadas com as criptomoedas ou outros ativos digitais. Elas funcionam como uma interface entre blockchain e a conta do usuário, permitindo o gerenciamento dos ativos. Essas carteiras são essenciais para realizar transações no universo das criptomoedas. As carteiras se dividem basicamente em dois tipos principais, sendo as carteiras *On* e *Off*. As carteiras *On* sempre estão conectadas às redes e são mais práticas para realizar transações. Os exemplos desse tipo de carteiras são os aplicativos mobile e extensões que rodam em navegador web. Já as carteiras *Off* não são diretamente conectadas à rede e armazenam as chaves offline. Oferecem maior segurança, porém com menor usabilidade. Essas carteiras podem ser tanto carteiras baseadas em dispositivos de hardware quanto escritas e armazenadas em papel.

O funcionamento das carteiras em blockchains, conforme apresentado na Figura 2.4, passa pelo uso de tecnologias que lhes permitem interagir com a blockchain, armazenando, enviando e recebendo transações. A principal tecnologia utilizada pelas carteiras é um par de chaves públicas/privadas. Essas chaves são geradas por meio de algoritmos criptográficos. Sendo a chave privada composta por uma sequência de números aleatórios, é requerida como uma senha. Já a chave pública atua como um endereço, semelhante a um número representando uma conta. Essa chave pode ser compartilhada para que possa receber transações direcionadas a ela. A chave pública é gerada a partir da chave privada que apenas o proprietário deve ter acesso a ela.

A interação das carteiras com as plataformas blockchain ocorre por meio de assinaturas digitais, apresentando a chave privada. Sempre que uma transação é realizada, a carteira utiliza a chave privada para assinar a transação digitalmente. As carteiras não armazenam as transações, limitando-se a gerenciar as chaves e interagir com os nós da blockchain. Ao receber um ativo em uma conta, a blockchain registra a transação no endereço público e a carteira apenas apresenta a posse desses ativos.

O consumo ou transferência dos ativos depende da criação, pela carteira, de uma

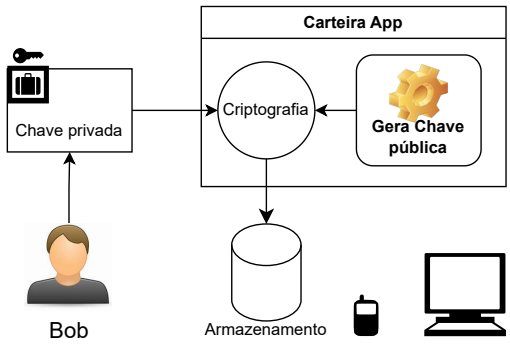


Figura 2.4: Carteiras Blockchain. Fonte: elaborado pelo autor

transação com o endereço de destinatário, valor do ativo e assinatura. Dessa forma, as carteiras blockchain protegem as transações por meio de métodos criptográficos. Alguns exemplos de implementações de carteiras blockchain são a Metamask [8], Trust [9] e Coinbase [10]. Os benefícios do uso de carteiras blockchain para interação com criptomoedas e ativos digitais são ter o controle sobre os ativos sem a necessidade de terceiros, segurança provida pela utilização de criptografia das chaves públicas/privadas e a transparência e imutabilidade, já que as transações são registradas e verificadas na blockchain. Ainda assim, há desvantagens como o risco de perda de acesso às contas e a possibilidade de vulnerabilidade a ataques.

2.3 Tokens

Nesta seção são tratados os conceitos de tokens e os aspectos das estruturas de um token, tais como seus componentes e padrões. Descrevemos os *Ethereum Request for Comments* (ERC), tais como ERC-20, ERC-721 e ERC-1155, que são padrões de contratos criados no nível de aplicação no ecossistema Ethereum para tokens e propostos nos *Ethereum Improvement Proposals* (Wang et al., 2021a). Eles são um fator importante para o sucesso dos NFTs, pois são padrões abertos que descrevem como construir tokens não fungíveis em blockchains e possuem um conjunto de regras que facilitam o trabalho do desenvolvedor. São descritas também as formas em que os padrões podem ser utilizados para troca de propriedade e serem negociados por outros tokens ou criptomoedas.

Token pode ter o significado de uma ficha ou um código que representa algo. Na tecnologia, a palavra token é atribuída a dispositivos ou sistemas que geram codificações de acesso e autenticação. Este termo significa a representação digital de um ativo no universo de uma blockchain. Portanto, ele é utilizado para descrever a criação de um

[8] https://metamask.io/
[9] https://trustwallet.com/
[10] https://www.coinbase.com/

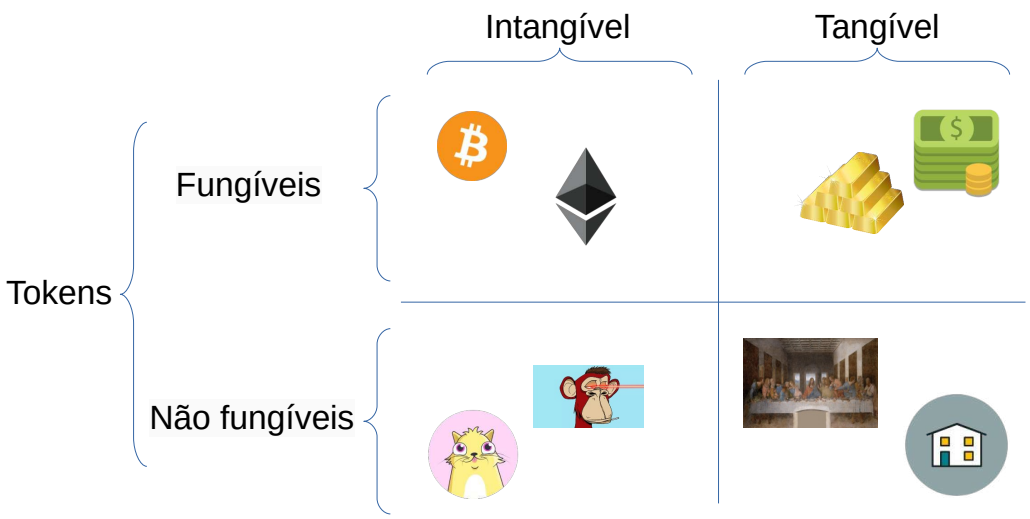


Figura 2.5: Tokens Fungíveis e Não Fungíveis. Fonte: elaborado pelo autor

registro digital de um ativo tangível, como por exemplo o ouro,o dinheiro, as obras de arte,os imóveis, os equipamentos, etc. Ou ainda um ativo intangível como, por exemplo, software, criptomoedas, artes digitais, etc.

Sendo assim, conforme representados na Figura 2.5, os ativos tangíveis ou intangíveis podem ser protegidos com a ajuda da tecnologia Blockchain e representados como um ativo digital por um token fungível ou não. Podendo representar dinheiro, certificados de propriedade e outros ativos. As próprias criptomoedas são consideradas como tokens. Existem, portanto, vários tipos de tokens e se distinguem de acordo com a proposta à qual forem designados. Dentre eles, os principais tipos são: tokens fungíveis ou de pagamentos, utilidades, não fungíveis e de segurança.

- O token fungível (FT) ou de pagamento exerce o papel de dinheiro eletrônico e é operado em transferência e/ou pagamentos. Os exemplos mais conhecidos desse tipo de token são o Bitcoin e o Ether.

  Definição 1 *Uma unidade de um FT é idêntica a todos os outros tokens em valor, para a mesma classe de tokens e é igualmente intercambiável com qualquer outro dentro de uma determinada classe.*

- Os tokens de utilidade são usados no oferecimento de algum benefício ao cliente ou usuário de um serviço ou produto. Eles fornecem utilidades como descontos, acessos a serviços especiais, entre outros.

- Os NFTs são tokens que podem representar algo singular, isto é, não podem ser trocados por outro de mesmo valor, como acontece com o FT, porque é único. Eles correspondem à identificação de um item único e vinculado a um proprietário. Em suma, um NFT é um código que contém o registro do objeto e de seu proprietário.

A maioria dos projetos de NFTs é originada da plataforma Ethereum ou a utiliza para prover garantias. No entanto, existe uma gama de outros projetos baseados em plataformas diferentes e que demonstram a independência da utilização dos NFTs de qualquer plataforma[11].

Definição 2 *Um NFT é um token exclusivo dentro de uma determinada classe de tokens e não é igual a nenhum outro NFT dentro de uma determinada classe.*

- Os tokens de segurança são uma nova modalidade de propriedade de ações, títulos e produtos regulados por instituições financeiras. Representam uma participação externa em um ativo e podem ser emitidos em blockchain públicas ou permissionadas.

Em síntese, os tokens são representações digitais de valores monetários, ativos físicos digitais ou reais. Eles são criados e gerenciados em uma blockchain. Os tokens permitem que ativos como imóveis, obras de arte, commodities ou até mesmo carros sejam "tokenizados", ou seja, convertidos em uma forma digital que pode ser negociada, transferida ou fracionada. Cada unidade de token pode ser vinculada a um ativo físico específico, garantindo que ele tenha um valor subjacente no mundo real. É ainda possível representar um ativo por meio de fracionamento, o que permite dividir um ativo em partes menores. Dessa forma, o ativo se torna mais acessível e os registros das transações dos tokens em blockchain tornam-os transparentes e seguros de acordo com as características da tecnologia. Por fim, quando um ativo é tokenizado e cada token representa uma fração do ativo, a posse da fração permite que diferentes usuários sejam proprietários do ativo sem adquiri-lo integralmente.

A posse de itens raros e exclusivos é comum no espaço físico real. A propriedade de itens digitais de forma exclusiva no espaço online digital se torna possível pela tecnologia blockchain. O "Non-fungible token" (NFT) ou Token não-fungível é um token ou chave de segurança com o objetivo de distinguir cada item com uma identificação única e com garantia de autoria e integridade (Wang et al., 2021a). Pessoas estão comprando itens digitais a valores extremamente altos, uma vez que diversas entidades aderem ao conceito de NFTs e disponibilizam seus produtos no mercado online (Chohan, 2021).

Um NFT é um ativo digital criado e acordado dentro de uma rede blockchain já com uma moeda digital pré-existente. Os NFTs são criados a partir da implantação de contratos inteligentes em uma blockchain e podem interagir uns com os outros. O desenvolvimento de NFTs não necessita da criação de uma rede blockchain exclusiva para ele, podendo utilizar uma rede já existente e a reputação da rede para gerenciar o registro da nova moeda por meio de um contrato inteligente hospedado na blockchain. Como

[11]https://chaindebrief.com/non-ethereum-nft-projects-zilliqa-solana/

exemplo, podemos comparar um NFT a um ingresso de acesso a um evento: o ingresso só tem valor perante o evento e permite o acesso aos seus serviços. Existe um crescente número de projetos registrados em blockchains públicas que emitem seus tokens para oferecer serviços diversificados (Chevet, 2018). Os Tokens derivam de vários padrões para NFTs que permitem aos desenvolvedores criar novos exemplares a partir destes padrões. O padrão Ethereum ERC-721, por exemplo, foi o primeiro a ser usado para a categoria NFT e tem um mecanismo herdável que permite aos desenvolvedores herdar o padrão para a criação de novos contratos compatíveis. (Wang et al., 2021a) descreveu a verificabilidade, transparência, disponibilidade, inviolabilidade, usabilidade, atomicidade e negociabilidade como as propriedades-chave de um NFT. Estas propriedades podem beneficiar no projeto de aplicações, como as do cenário voltado para a privacidade de usuários, no esquema de controle de acesso, rastreabilidade e interoperabilidade de dados. As pesquisas atuais não incluem soluções para o anonimato e a privacidade dos NFTs, uma vez que todas as atividades são registradas em um ambiente público e podem ser observadas. Soluções já propostas para preservação de privacidade , tais como zero-knowledge proof (Wang and Kogan, 2018), ring signature (Noether, 2015) e multiparty computation (Raman et al., 2018), não foram adotadas pelos NFTs devido à complexidade computacional empregada.

2.3.1 Estrutura e Padrões de tokens

A estrutura de um token é realizada fundamentalmente em um contrato inteligente. Esta estrutura contém basicamente um mapa de endereços de contas e seus saldos vinculados. O saldo, chamado de token, simboliza um valor que também pode ser utilizado na representação de objetos, valores monetários e outras finalidades, conforme mencionadas nos itens da seção anterior. As operações sobre esta estrutura discorrem de transferências da propriedade destes tokens. Ao realizar esta transferência, os contratos atualizam o saldo das duas contas envolvidas, creditando-o em uma das contas e debitando na outra.

Os padrões de tokens foram sugeridos inicialmente para a plataforma Ethereum. Na plataforma Ethereum, quando surge uma nova proposta que altera um procedimento, deve-se fazer por meio de uma Proposta de Melhorias no Ethereum ou *Ethereum Improvement Proposal* (EIP). Esta proposta consiste em um documento que detalha as alegações de mudanças e as questões técnicas envolvidas. Por meio dos EIPs é possível submeter proposições de comentários, mudanças em protocolos e outros detalhes da Ethereum. No caso dos tokens, os documentos são submetidos como solicitação de comentários Ethereum ou *Ethereum Request for Comments (ERC)*, que é a forma de definir padrões de uso na Ethereum.

As EIPs submetidas passam por um ciclo de vida e recebem um status em cada etapa, que vai desde o rascunho da ideia até a versão final. As EIPs que tratam de conteúdos

técnicos e padrões referentes aos contratos inteligentes passarão a ser nomeadas com a sigla ERC após a aprovação na fase final. Existem vários EIPs/ERCs que foram submetidos e aprovados como padrões por meio deste ciclo e que são comumente utilizados para definir tipos de tokens. Dentre eles citamos os padrões ERC-20, ERC-721 e ERC-1155 que tratam dos tokens fungíveis e tokens não-fungíveis, respectivamente.

Inicialmente, o padrão ERC-20 foi introduzido como uma experiência de proporcionar recursos para padronização de criação de contratos de tokens. Este padrão trouxe muitos benefícios, permitindo uma integração de vários tokens com uma mesma carteira e listagem em plataformas digitais onde é possível comprar, vender, trocar e guardar os tokens. Estes benefícios avançaram para outros padrões de outros tipos de tokens conforme apresenta a Tabela 2.4.

Tabela 2.4: Padrões de tokens Ethereum com casos de uso

Padrão de token	Definições	Utilidade
ERC 20	Baseado em saldos Valor fungível	Oferta inicial de criptomoedas token de segurança tokens de utilidade
ERC 721	Valor não fungível Cada elemento é único	Tokenização de ativos reais Registro de propriedade de ativos colecionáveis e virtuais
ERC 1155	Combinação dos padrões ERC-20 e ERC-721 Permite que diferentes tokens sejam configurados de um único ponto	Conjunto de ativos heterogêneo

2.3.1.1 Padrão ERC-20

O ERC-20 é um padrão de contratos inteligentes que permite a criação de tokens fungíveis no Ethereum, isto é, tokens que são iguais e possuem o mesmo valor independentemente do índice que o representa. O padrão foi proposto por (Vogelsteller and Buterin, 2015). Por meio da implementação da interface deste padrão é possível utilizar inúmeras das funcionalidades pré-estabelecidas. Tais funcionalidades como as de transferência de tokens entre contas, verificação do saldo da conta, verificação do total de tokens criados e aprovação de tokens para utilização de terceiros possibilitam a reutilização e compatibilidade por aplicações como carteiras e exchanges descentralizadas. Há vários tokens já implantados e seguindo o padrão ERC-20 na rede Ethereum. Podemos encontrar em (Charles, 2013) implementações de exemplo que são utilizadas pelas dApps. A Listagem 2.2 apresenta alguns dos métodos deste padrão que são relevantes para nossa discussão e prática.

```
1 // SPDX-License-Identifier: UNLICENSED
2 pragma solidity ^0.4.20;
3 interface ERC721{
4 event Transfer(address, address, uint256)
5 event Approval(address, address, uint256)
6 function name() public view returns (string)
7 function symbol() public view returns (string)
```

```
8  function decimals() public view returns (uint8)
9  function totalSupply() public view returns (uint256)
10 function balanceOf(address) public view returns (uint256)
11 function transfer(address, uint256) public returns (bool)
12 function transferFrom(address, address, uint256) public returns (bool)
13 function approve(address, uint256) public returns (bool)
14 function allowance(address, address) public view returns (uint256)
15 }
```

Listagem 2.2: Padrão para token fungível: EIP-20

2.3.1.2 Padrão ERC-721

O padrão ERC-721, também conhecido como o padrão de NFTs, é utilizado na identificação de itens únicos e exclusivos. Este tipo de padrão fornece a capacidade ao token de realizar funções como certificação de propriedade, identificação de credenciais, identificação de acesso, ingressos para eventos, itens colecionáveis, etc.

Os NFTs possuem as características básicas do token ERC-20, porém apresentam um identificador exclusivo para cada registro, tornando-os únicos e exclusivos. Uma interface padrão para os NFTs foi apresentada por (Entriken et al., 2018). Este padrão pode ser considerado como um certificado de posse que permite a implementação de uma API padrão para NFTs utilizando os contratos inteligentes. As funcionalidades básicas para rastrear e transferir NFTs são apresentadas na Lista de código 2.3.

```
1  // SPDX-License-Identifier: UNLICENSED
2  pragma solidity ^0.4.20;
3  interface ERC721{
4    event Transfer(address, address, uint256);
5    event Approval(address, address, uint256);
6    event ApprovalForAll(address, address, bool);
7    function balanceOf(address) external view returns (uint256);
8    function ownerOf(uint256) external view returns (address);
9    function safeTransferFrom(address, address, uint256, bytes) external payable;
10   function safeTransferFrom(address, address, uint256) external payable;
11   function transferFrom(address, address, uint256) external payable;
12   function approve(address, uint256) external payable;
13   function setApprovalForAll(address, bool) external;
14   function getApproved(uint256) external view returns (address);
15   function isApprovedForAll(address, address) external view returns (bool);
16 }
```

Listagem 2.3: Padrão para token não fungível: EIP-721

2.3.1.3 Padrão ERC-1155

O padrão ERC-1155 é chamado de padrão multitoken e foi idealizado para controlar um número maior de tokens simultaneamente. Para abranger tokens fungíveis e não fungíveis, o ERC-1155 contém as funções do token ERC-20 e do ERC-721 (Radomski et al., 2018). Por meio de uma interface padrão utilizada por um único contrato inteligente, é possível gerenciar vários tipos de token. As funcionalidades básicas desta interface são apresentadas na Lista de código 2.4.

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.5.9;
interface ERC1155
    event TransferSingle(address, address, address, uint256, uint256);
    event TransferBatch(address, address, address, uint256[], uint256[]);
    event ApprovalForAll(address, address, bool);
    event URI(string, uint256);
    function safeTransferFrom(address, address, uint256, uint256, bytes) external;
    function safeBatchTransferFrom(address, address, uint256[], uint256[], bytes) external;
    function balanceOf(address, uint256) external view returns (uint256);
    function balanceOfBatch(address[], uint256[]) external view returns (uint256[]);
    function setApprovalForAll(address, bool) external;
    function isApprovedForAll(address, address) external view returns (bool);
}
```

Listagem 2.4: Padrão para Multitoken: EIP-1155

Os padrões ERC-20 e ERC-721 requerem a implantação de contratos distintos por cada tipo de token. O padrão ERC-721, por exemplo, contém um ID do token que é um índice que representa um item não fungível desse grupo e é implantado como um único contrato que abrange configurações para todo o grupo de tokens. Já o ERC-1155 permite que cada ID de token represente um novo tipo de token configurável, que pode ter seus próprios metadados, funções e atributos. Tarefas como a transferência de vários tipos de token simultaneamente são possíveis por meio das funcionalidades adicionadas a este padrão, resultando em economia nos custos de transação.

2.3.2 Negociação de tokens

Os tokens ERC-721 são negociados de forma diferente das criptomoedas e outros tipos de tokens. O valor dos NFTs geralmente depende de sua relação fora da Blockchain em que são negociados e também da raridade imposta a eles (Rogers et al., 2022). Alguns NFTs só podem ser comprados com criptomoedas, portanto, é necessário possuir parte dessa criptomoeda e armazená-la em uma carteira digital. A partir daí é possível comprar NFTs por meio de qualquer um dos mercados NFT online, incluindo OpenSea, Rarible e Cryptopunks. Quando ocorre uma transferência de tokens fungíveis, por exemplo, do tipo ERC-20, são submetidos ao consenso da rede o débito e o crédito nos saldos das contas envolvidas na transação. Já a transferência de NFTs, isto é, o token não fungível, ocorre a mudança da propriedade daquele registro individual que não será agregado a nenhum outro token. A Figura 2.6 apresenta duas funções executadas na transferência e obtenção de dados de um NFT em forma sequencial.

Neste caso, os usuários A e B realizam uma transferência da propriedade de um token com um ID específico. Onde o usuário A é o atual proprietário do token e o usuário B é quem receberá a propriedade do token. A transferência do token só se dá por meio da execução da função ***safeTransferFrom*** acionada pelo proprietário do token. Esta função recebe os dados de identificação do atual proprietário, identificação do usuário para quem vai ser transferido o token, juntamente com a identificação do token no contrato.

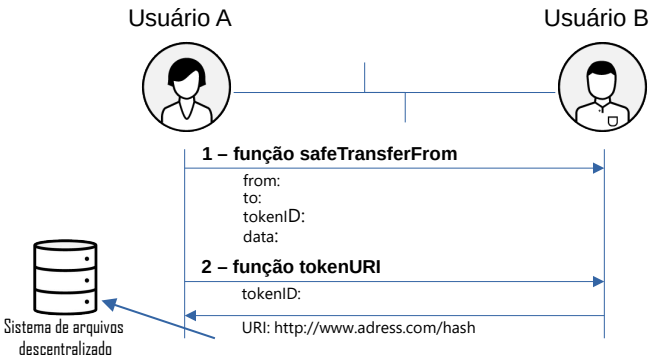


Figura 2.6: Sequência para transação de transferência de NFTs. Fonte: elaborado pelo autor

A função ***tokenURI*** solicita o retorno do metadado URI armazenado ao criar o registro do token.

A compra e venda de NFTs deve ser de comum acordo entre as partes envolvidas, porém a efetivação da transferência só é realizada pelo vendedor. Apesar da iniciativa da compra ser comumente feita pelo comprador nos mercados de tokens, o comprador tem simplesmente a função de pagar o valor combinado pela transação. Normalmente os marketplaces realizam a intermediação de todo o procedimento da transferência e do pagamento ao proprietário do NFT.

O aluguel é uma outra modalidade e possibilidade de transação com os NFTs em que é permitido alugar e realizar o pagamento do aluguel de NFTs da mesma forma da venda. Porém, neste caso o NFT recebe o status de alugado, onde o usuário que o alugou pode usá-lo, mas não pode transferi-lo. Somente quando o aluguel terminar e o NFT voltar ao seu proprietário, o NFT poderá ser transferido novamente.

Como o pagamento nestes mercados é realizado normalmente por meio de criptomoedas, existem também mercados de criptomoedas, também chamados de exchanges, que oferecem aos usuários o recurso de poderem comprar e vender as criptomoedas. Este serviço é normalmente incrementado pelas exchanges para oferecerem trocas por moedas correntes tais como, por exemplo, o dólar ou o euro. Há três tipos de exchanges de criptomoedas: exchanges centralizadas que são dirigidas por uma empresa ou organização, exchanges descentralizadas que fornecem processos automatizados para negociações entre os próprios usuários e exchanges híbridas que combinam as duas opções (Xia et al., 2020). Os usuários dispõem de saldos em sua aplicação de carteira ao adquirir as criptomoedas, que poderão ser utilizadas para negociação dos NFTs em seus marketplaces.

2.4 Arquitetura típica para Aplicação Descentralizada em Blockchain

O projeto típico de uma arquitetura contém a descrição e organização dos componentes presentes em uma solução específica. No caso de aplicações descentralizadas, os componentes necessários para dar suporte a essa aplicação dispõem principalmente de meios tecnológicos para acessar e interagir com funções de contratos inteligentes, implementados e implantados em uma plataforma Blockchain. Esta arquitetura deve ser projetada com a finalidade de permitir que uma aplicação externa possa comunicar com a Blockchain e permitir a chamada das funções implementadas no contrato inteligente. A Figura 2.7 apresenta um exemplo e modelo de arquitetura para aplicações descentralizadas. Esse modelo foi aplicado no desenvolvimento da solução proposta pela aplicação cliente do CST e estabeleceu suporte ao funcionamento da estrutura projetada. A arquitetura necessária para o projeto do CST é composta pelos componentes: Plataforma Blockchain, Contratos Inteligentes, Sistema de retaguarda ou *Back-end*, Interface do utilizador ou *Front-end*, Carteira digital ou (*Wallet*) e dispositivos com acesso à Internet. Esses componentes são estruturados, de acordo com o modelo,com o propósito de oferecer a possibilidade de avaliação dos custos de recursos computacionais necessários para prover a aplicabilidade e viabilidade do token de cessão.

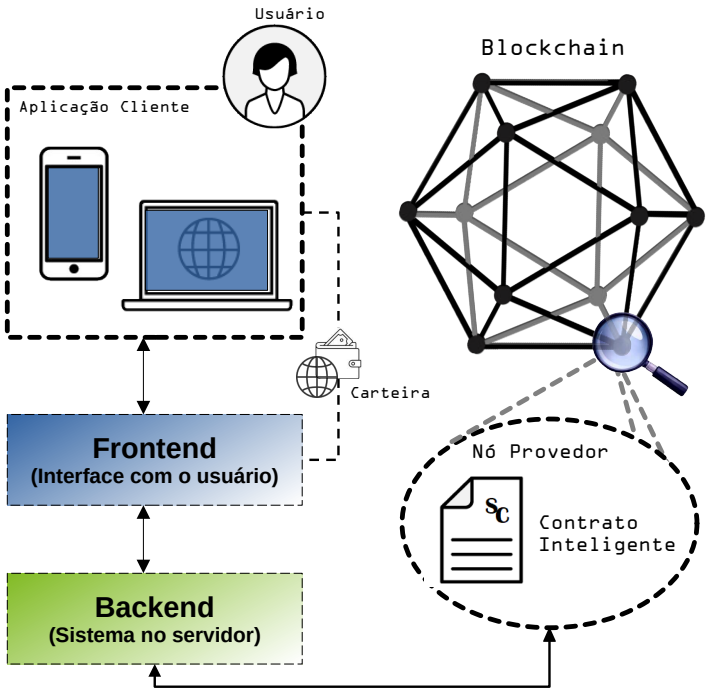


Figura 2.7: Arquitetura utilizada pela solução CST em Blockchain. Fonte: elaborado pelo autor

O componente **Blockchain** caracteriza a utilização de uma plataforma pública

blockchain como a base fundamental de toda a estrutura desenvolvida. Conforme já mencionado, a tecnologia blockchain é que oferece às dApps a segurança quando não há confiança entre as partes envolvidas em uma transação, a rastreabilidade das transações e a descentralização das responsabilidades da rede. Um nó da rede é capaz de servir como meio de entrada e saída de dados. O processo de implantação e invocação dos contratos gera transações que são agrupadas em blocos e que serão propagadas para os outros nós pertencentes à rede. O acesso externo a estes **Nós Provedores** da rede é realizado por meio de encaminhamento de mensagens, estruturadas em padrão de dados JSON RPC Request, a endereços que dispõem de portas e serviços de Web Service e/ou HTTP disponíveis.

Os **Contratos Inteligentes** são implantados nas plataformas Blockchain através de um dos nós da rede e é por onde também que as suas funções são invocadas. Estabeleceu-se a operação de forma modular para a conexão ao nó, invocação das funções dos contratos e interface de interação dos usuários por meio dos conceitos de Back-end e Front-end.

O módulo **Back-end** é a parte que estabelece a comunicação da aplicação com a Blockchain e gerencia as conexões dos usuários. Inclui ainda toda a parte lógica da abertura de conexões com o nó da plataforma blockchain e uma API. Essa API contém um conjunto padronizado de chamadas de funções para interação com os contratos inteligentes através de um front-end.

O módulo **Front-end** contempla todo o meio de interação do usuário com a aplicação através de elementos visuais. A parte visual inclui telas com componentes gráficos e componentes para a integração com carteiras digitais (Wallet). Estas carteiras armazenam chaves públicas e privadas das contas dos usuários de forma segura. O acesso ao Front-end é por meio de dispositivos com navegadores e acesso à Internet.

2.5 Interoperabilidade

As plataformas Blockchains executam diferentes conjuntos de transações por meio de implementações distintas e normalmente isoladas. Devido a esta heterogeneidade, há uma grande dificuldade em compartilhar informações através de várias redes blockchain. Estas características trouxeram em evidência o estudo de tecnologia para prover a interoperabilidade da Blockchain. Interoperabilidade é definida como a capacidade de haver cooperação entre partes envolvidas em uma solução, ainda que utilizem diferentes tecnologias (Wegner, 1996).

A tecnologia cross-chain é o envolvimento de um par de blockchains em que um tipo de aplicação descentralizada facilita a transferência de ativos de uma blockchain para outra, promovendo a interoperabilidade. Isso significa que é possível mover ativos, como criptomoedas e tokens, de uma blockchain para outra sem a necessidade de intermediários centralizados. Uma das maneiras de implementar cross-chain envolve a

criação de pontos de conexão entre as blockchains, conhecidos como "pontes". Essas pontes permitem que as transações sejam validadas em ambas as blockchains, garantindo a segurança e a integridade dos ativos transferidos. De maneira geral, as transferências de ativos entre cadeias seguem um procedimento atômico, baseado no protocolo *Cross-chain communication protocol (CCCP)* em que há o bloqueio de um ativo na origem, a responsabilidade de transferência e a criação do ativo no destino (Belchior et al., 2021).

(Qasse et al., 2019) cita inúmeros casos de uso em que a interoperabilidade em Blockchain pode ser alcançada. Dentre eles estão as transferências de ativos de uma aplicação ou cadeia para outra. Troca de ativos entre dois usuários em diferentes redes blockchain de maneira segura e atômica. Bloqueio de ativos e liberação em outra cadeia mediante condições. Uso de oráculos entre cadeias quando é necessário o uso de dados para executar uma ação em cadeia diferente. Contratos em que há dependência de dados de várias cadeias para que uma ação seja acionada. Existem ainda alguns desafios para atingir a interoperabilidade. Dentre eles está a necessidade de garantia de atomicidade, melhoria na eficiência de manutenção da segurança, tolerância à diversificação de plataformas, bem como facilidade para desenvolvedores de aplicações (Jin et al., 2018).

Os tipos de interoperabilidade em Blockchain mais comuns encontrados são: interoperabilidade entre Blockchains (homogêneas), interoperabilidade entre dApps usando diferentes Blockchains e interoperabilidade de Blockchains com outras tecnologias (heterogêneas) Besançon et al. (2019). Assim, a Figura 2.8 demonstra um diagrama de fragmentação desses tipos e apresenta como são definidas as transações entre os tipos. As transações de Blockchains (homogêneas) são nomeadas como uma transação cross-chain (CC-Tx), onde “CC” significa cross-chain e “Tx” significa transação. Uma transação cross-Blockchain (CB-Tx) é uma transação entre diferentes blockchains (heterogêneos) e, por fim, uma aplicação descentralizada cross-chain (CC-dApp) que é um dApp que utiliza as transações cross-Blockchain para implementar seus requisitos (Belchior et al., 2021).

No contexto dos tokens, a interoperabilidade entre plataformas pode contribuir para a usabilidade ao implicar a capacidade de transferir um ativo entre cadeias distintas, mantendo o estado e o histórico consistentes. A interoperabilidade de cadeias deve atingir a eficiência de dois tipos, cada um dos quais traz considerações distintas, porém contribuindo para a usabilidade. A troca de ativos digitais entre cadeias é um dos tipos de interoperabilidade. Ele deveria conter a capacidade de transferir e trocar ativos originários de diferentes cadeias sem intermediários confiáveis, como trocas centralizadas. Um exemplo disso seria tornar um token originário de uma cadeia válido em qualquer outra cadeia disponível. Outro tipo de interoperabilidade desejada diz respeito à troca de informações que mantém a capacidade de fazer algo em uma cadeia que reflete em outra cadeia. Esta troca deve permitir o rastreamento não só de ativos ou

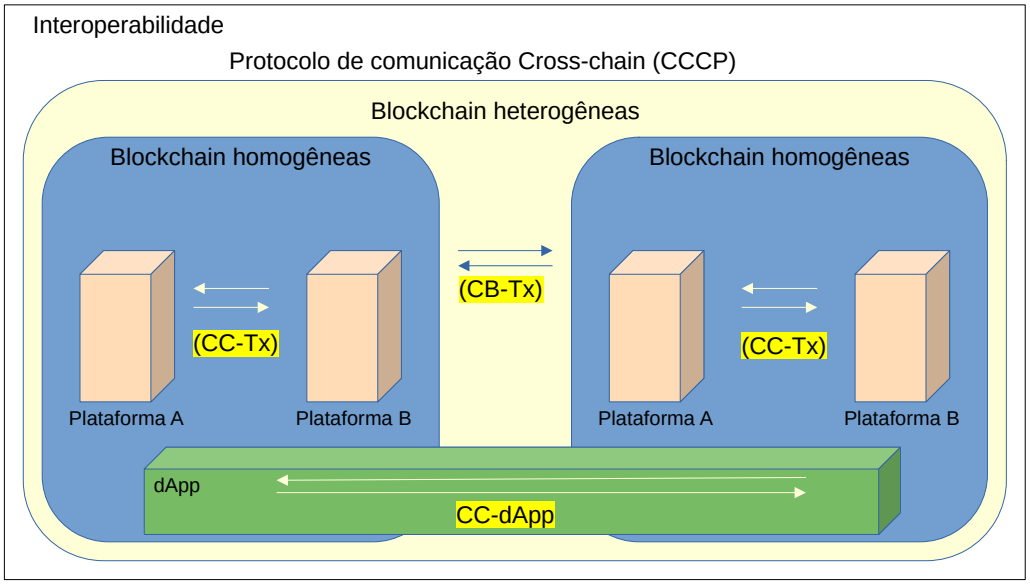


Figura 2.8: Tipos de interoperabilidade e transações em Blockchain. Fonte: elaborado pelo autor

itens negociáveis, mas também das operações executadas. Como exemplo, o compartilhamento do histórico de transações de um determinado item contendo negociações e proprietários.

2.5.1 Mecanismos Cross-chain

Com tantas oportunidades de aplicativos de negócios com requisitos de interoperar Blockchains, prover soluções com mecanismos genéricos de *cross-chain* para conectar redes Blockchain homogêneas e heterogêneas amplia o espaço de desenvolvimento da tecnologia Blockchain (Buterin, 2016). Algumas soluções foram propostas para interoperar acesso a dados e transações entre Blockchains, como o mecanismo notarial (*Notary mechanism*) e bloqueio de hash (*Hash-locking* ou *Hash time lock*) (Belchior et al., 2021). Estes mecanismos propõem soluções que podem abranger um número maior de variedades de aplicações e distintas soluções de blockchain.

2.5.1.1 SideChain

As sidechains são blockchains independentes conectadas a uma blockchain principal, geralmente chamada de mainchain. Essas redes paralelas têm como principal objetivo aumentar a eficiência do ecossistema blockchain, permitindo a transferência segura de ativos e informações entre diferentes cadeias. Ao operar de forma autônoma, uma sidechain pode executar tarefas específicas, testar novas funcionalidades e processar transações com maior velocidade, sem sobrecarregar ou comprometer a segurança da rede principal (Ethereum, 2024).

A principal característica técnica que viabiliza essa comunicação entre cadeias é o chamado pegging bidirecional (Back et al., 2014). Esse mecanismo permite que ativos sejam bloqueados na mainchain e liberados de forma correspondente na sidechain, garantindo que o valor econômico seja mantido de forma segura e verificável. Esse processo geralmente é realizado por meio de contratos inteligentes ou soluções criptográficas automatizadas, que asseguram a integridade e a rastreabilidade das transferências. Dessa forma, os usuários podem migrar seus ativos entre redes distintas, sem que ocorra duplicação ou perda de valor.

Uma das vantagens mais expressivas das sidechains é a possibilidade de utilizar algoritmos de consenso próprios, distintos daqueles empregados na blockchain principal. Por exemplo, enquanto o Bitcoin adota o mecanismo de Proof of Work (PoW), uma sidechain pode operar com Proof of Stake (PoS) ou outros modelos mais eficientes, conforme a finalidade da aplicação. Essa independência operacional permite que as sidechains sejam moldadas para usos específicos, como jogos, sistemas financeiros, redes privadas ou experimentação de novas tecnologias (Sguanci et al., 2021).

Apesar de conectadas à mainchain, as sidechains não dependem dela para segurança. Cada sidechain é responsável por seus próprios validadores ou mineradores, o que significa que precisa adotar suas próprias políticas e mecanismos de proteção contra ataques, fraudes ou falhas. Ao mesmo tempo, essa independência também viabiliza ganhos significativos de desempenho, já que as sidechains podem ser otimizadas para processar transações com maior rapidez e menor custo, aliviando a carga computacional da rede principal (Antonopoulos, 2014).

Para assegurar a comunicação entre essas diferentes redes, são utilizadas estruturas chamadas pontes. Essas pontes são componentes de software ou infraestrutura criptográfica que gerenciam a validação e sincronização de transações entre blockchains distintas. Em muitos casos, contam com operadores especializados, contratos inteligentes ou oráculos que garantem a segurança e a consistência dos dados transferidos. Dessa forma, mesmo operando com regras diferentes, as blockchains conseguem manter um fluxo confiável de ativos e informações entre si.

O potencial de aplicação das sidechains é vasto e já pode ser observado em diversos projetos concretos. A Polygon, por exemplo, é uma sidechain amplamente utilizada para melhorar a escalabilidade da rede Ethereum, oferecendo transações mais rápidas e com taxas significativamente menores (Bjelic et al., 2017). Já a Liquid Network, sidechain do Bitcoin, foi desenvolvida para atender a instituições financeiras que necessitam de maior privacidade em suas transações, utilizando tecnologias como as Confidential Transactions. Além disso, as sidechains têm sido fundamentais para aplicações em finanças descentralizadas (DeFi). A plataforma RSK, por exemplo, funciona como uma sidechain do Bitcoin e é compatível com contratos inteligentes no padrão Ethereum, expandindo as possibilidades de uso financeiro dessa criptomoeda (Lerner, 2015) .

Outro uso relevante ocorre em ambientes de experimentação tecnológica, nos quais desenvolvedores testam novos algoritmos, modelos de governança ou mecanismos de consenso antes de aplicá-los em redes maiores. Também há aplicações corporativas em tokenização de ativos, como pontos de programas de fidelidade ou créditos internos, e em identidade digital, onde informações sensíveis podem ser armazenadas com mais privacidade e controle.

Dessa forma, as sidechains vêm se consolidando como uma solução versátil e robusta no universo blockchain, permitindo maior inovação, escalabilidade e interoperabilidade. Ao funcionarem de forma modular e personalizada, elas contribuem diretamente para a superação de limitações das blockchains tradicionais, ao mesmo tempo que ampliam as possibilidades de uso descentralizado em diversos setores, como finanças, jogos, identidade digital e privacidade de dados.

2.5.1.2 Relays

O protocolo relay é um método de interoperabilidade entre blockchains que permite que uma blockchain observe o estado de outra blockchain. Nele, uma blockchain pode enviar eventos, informações ou transações para outra blockchain, onde um contrato inteligente ou verificador pode processar e validar esses dados. Através do conhecimento do estado da blockchain de destino pela blockchain de origem, é eliminada a necessidade de uma comunicação direta entre as duas. Os relays são comumente utilizados para permitir transferências *cross-chain*, como a movimentação de ativos de uma blockchain para outra, mantendo a segurança e a confiabilidade, uma vez que a blockchain receptora pode verificar diretamente o estado da blockchain de origem (Zamyatin et al., 2019).

O funcionamento do protocolo Relay baseia-se em uma arquitetura na qual uma blockchain, chamada de blockchain verificadora, mantém um registro de estado ou uma prova da blockchain monitorada. Esse processo é viabilizado por provas criptográficas, como as *Merkle proofs*, que permitem à blockchain verificadora validar transações da blockchain monitorada sem depender de um intermediário de confiança (Kotey et al., 2023). Esse mecanismo é importante para assegurar a autenticidade das informações compartilhadas entre blockchains, garantindo que elas não possam ser manipuladas por terceiros, o que aumenta a segurança e a confiabilidade das operações *cross-chain*.

Para o protocolo Relay operar de maneira eficaz, é necessário que a blockchain verificadora esteja constantemente atualizada com os blocos e transações da blockchain monitorada. Essa atualização pode ocorrer de forma periódica ou em resposta a solicitações específicas, dependendo do design do sistema e das necessidades de integração. Dessa forma, sempre que ocorre uma transferência *cross-chain*, a blockchain verificadora consulta a prova de estado mais recente para assegurar que o ativo realmente existe e que as informações fornecidas pela blockchain monitorada são válidas

e atuais, reforçando a segurança do processo (Zamyatin et al., 2019).

Como mostrado na Figura 2.9, o funcionamento do protocolo Relay envolve um contrato inteligente que valida as transações de uma blockchain monitorada.

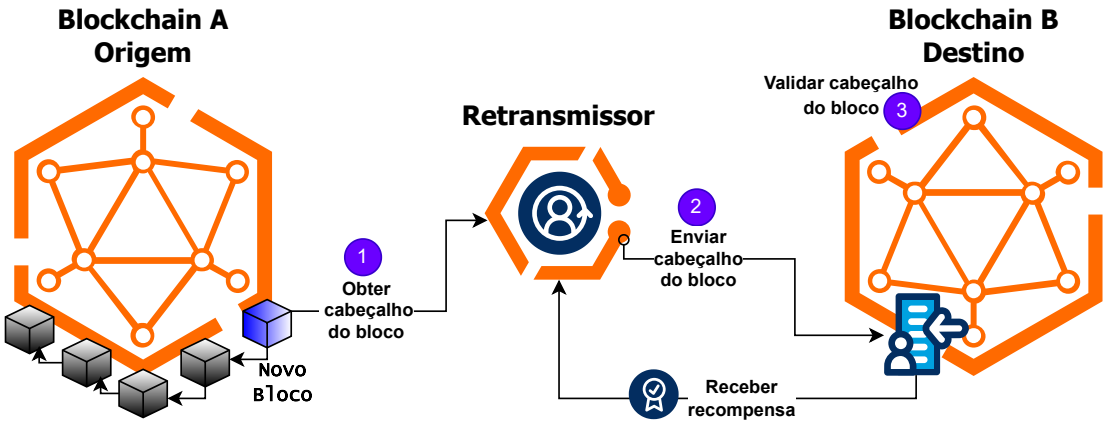


Figura 2.9: Esquema do Funcionamento do Relay. Fonte: elaborado pelo autor

2.5.1.3 Protocolos Agnósticos

Protocolos agnósticos de interoperabilidade são mecanismos que permitem a comunicação e transferência de dados ou ativos entre diferentes blockchains, sem depender das características específicas de uma blockchain em particular. Isso significa que esses protocolos funcionam independentemente do algoritmo de consenso, da estrutura de dados ou de outras particularidades técnicas de cada blockchain, proporcionando um meio universal de interação. Dessa forma, eles facilitam a conectividade entre ecossistemas variados, possibilitando a criação de redes interconectadas onde usuários e desenvolvedores podem transferir ativos ou informações entre blockchains de maneira transparente e eficiente, sem que as blockchains precisem ser modificadas para suportar o protocolo (Pupyshev et al., 2020).

O funcionamento geral dos protocolos agnósticos envolve três etapas principais. Primeiro, ocorre a comunicação de dados, em que uma blockchain de origem transmite informações, como transações ou estados, para uma blockchain de destino por meio de pacotes de dados protegidos criptograficamente. Em seguida, ocorre a etapa de verificação de provas, em que a blockchain de destino valida a autenticidade dos dados recebidos utilizando provas criptográficas, como *Merkle proofs* ou *Zero-Knowledge proofs*, que asseguram que a informação não foi manipulada. Por fim, após a verificação, a blockchain de destino executa a transação de acordo com os dados recebidos. Esse processo final garante que a integridade das transações seja mantida e que ambas as blockchains envolvidas permaneçam seguras e independentes (Herlihy, 2018).

2.5.1.4 Mecanismo Notarial

O mecanismo notarial é uma forma de implementar a interoperabilidade entre cadeias de forma relativamente simples. Ele consiste em verificar e encaminhar mensagens entre cadeias por meio de uma entidade confiável intermediária chamada de notário. Quando há troca e transferência de ativos entre diferentes sistemas Blockchain, uma ou mais organizações são designadas como notários para monitorar eventos entre as cadeias, e alcançar um consenso sobre a ocorrência do evento por meio de um algoritmo de consenso específico, e, por fim, responder de forma tempestiva (Belchior et al., 2021). O mecanismo notarial se divide em mecanismo notarial de assinatura única e de múltiplas assinaturas (Ou et al., 2022).

O mecanismo notarial de assinatura única, também denominado mecanismo notarial centralizado, consiste em designar um único nó ou instituição independente para atuar como notário, e o notário assume as tarefas de coleta de dados, verificação e confirmação de transações no processo de interação entre cadeias. O notário é composto por, pelo menos, uma conta nas cadeias de origem e de destino. Este mecanismo consegue ter um processamento rápido de transações e é bastante adaptável, apesar do escopo restrito, limitando-se à troca de ativos.

No mecanismo notarial de múltiplas assinaturas, o notário é geralmente composto por vários nós, onde cada nó possui uma chave, e somente quando uma determinada porcentagem destes nós assina em conjunto é que há um consenso e as transações entre cadeias podem ser confirmadas. Durante a verificação da transação, uma parte dos notários é selecionada aleatoriamente do grupo notarial, diminuindo o grau de dependência da confiabilidade dos notários (Yang et al., 2019). A Figura 2.10 ilustra o mecanismo notarial onde, na arquitetura do mecanismo notarial, os usuários envolvidos na transferência de tokens devem interagir com o notário.

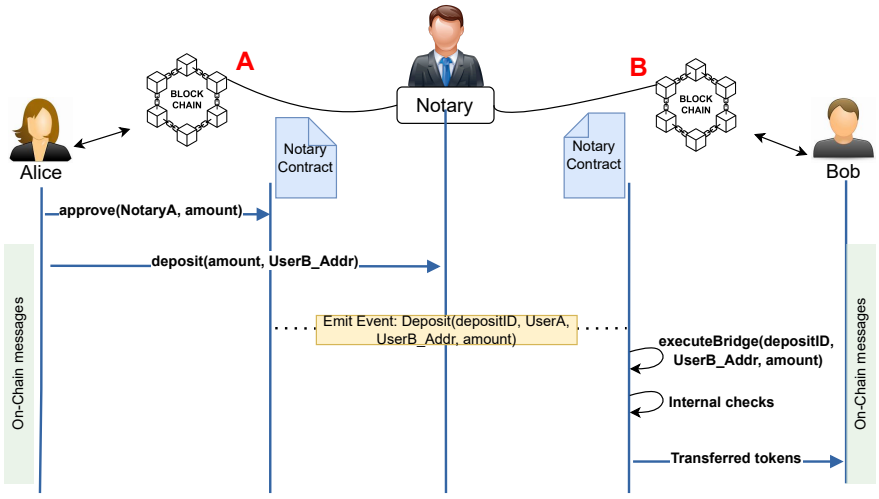


Figura 2.10: Arquitetura do mecanismo Notarial. Fonte: elaborado pelo autor

Essa interação pode ocorrer por meio de aplicativos descentralizados executados em blockchain ou contratos inteligentes, com o domínio de um terceiro confiável. O Notário desempenha o papel de receptor do token do usuário Alice (remetente) na blockchain A, transferindo-o para o usuário Bob (destinatário) na blockchain B e registrando informações sobre as transações realizadas. O Notário deve garantir a entrega segura dos recursos ao destinatário designado.

2.5.1.5 Bloqueio de Hash

O Mecanismo de Bloqueio de *Hash ou Hash Time Lock Contract* (HTLC) representa um marco significativo na evolução dos mecanismos de troca entre Blockchains, proporcionando uma solução inovadora para realizar transações entre redes sem depender de intermediários confiáveis (Ou et al., 2022). Ao implementar contratos HTLC nas blockchains envolvidas na negociação, o processo de troca de ativos é seguro e confiável. Esse contrato atua como uma garantia, bloqueando os ativos envolvidos até que as condições acordadas sejam atendidas. A utilização do conceito de *hash* adiciona uma camada adicional de segurança, criando uma trava com uma palavra secreta que deve ser correspondente em ambas as extremidades da transação. Além disso, a imposição de um limite de tempo para a conclusão da troca aumenta a eficiência e a segurança do processo. Em caso de não cumprimento dentro do prazo estipulado, o contrato automaticamente cancela a transação, revertendo os tokens para suas respectivas carteiras de origem. Esse mecanismo desempenha um papel fundamental na facilitação de trocas descentralizadas, promovendo a confiança e a segurança nas transações Blockchain (Belchior et al., 2021). A Figura 2.11 ilustra o processo do HTCL, demonstrando como os dois contratos funcionam juntos para garantirem a segurança e a atomicidade da transação.

No contexto do protocolo de bloqueio de *hash*, conforme a Figura 2.11, o usuário Alice deseja transferir em troca de seu token para uma conta do usuário Bob em outra rede. Para fazer essa transferência, ela escolhe uma "palavra secreta"e utiliza o *Hash* juntamente com o endereço de Bob para criar o contrato HTLC. Por meio do contrato criado na blockchain A, o contrato bloqueia o token A para a transferência ser realizada. De maneira similar, o Usuário Bob implanta o contrato de HTLC na blockchain B com o endereço de Alice e a palavra secreta em *Hash*. Sendo assim, o Usuário Alice faz a retirada do token B na blockchain B com sua palavra secreta, isto é, a etapa de (*withdraw*) do contrato HTLC. Ao fazer isso, a palavra secreta pode ser usada por Bob para retirar o token A da blockchain A.

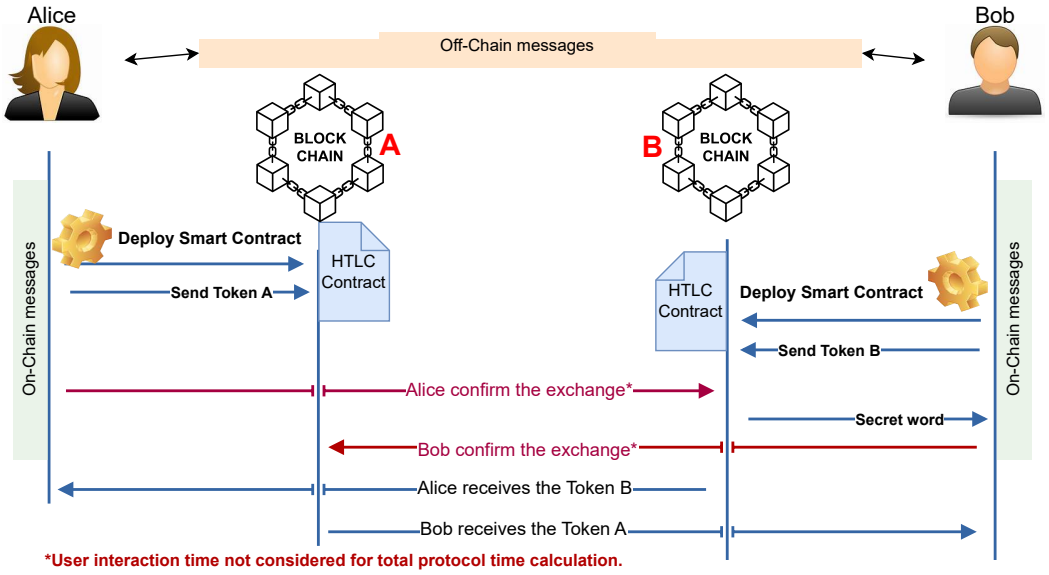


Figura 2.11: Arquitetura implementada para o protocolo *Hash-Time Lock*. Fonte: elaborado pelo autor

2.6 Considerações Finais

Esse capítulo apresentou um panorama abrangente e estruturado sobre os fundamentos da tecnologia blockchain, contextualizando sua aplicação em sistemas distribuídos e destacando conceitos essenciais como tolerância a falhas, protocolos de consenso, contratos inteligentes, carteiras digitais, tokens e interoperabilidade. A abordagem teórica evidenciou a relevância dos sistemas distribuídos como base das soluções blockchain, destacando as propriedades de escalabilidade, disponibilidade e resistência a falhas. Os algoritmos de consenso foram discutidos, evidenciando os desafios técnicos associados à segurança, à latência e à eficiência computacional. A tecnologia blockchain foi explorada em suas múltiplas formas — públicas e permissionadas, que abrangem as privadas e de consórcio — permitindo compreender a variedade de modelos possíveis para diferentes contextos de uso. As plataformas Ethereum e Hyperledger Fabric foram comparadas em detalhes, enfatizando suas arquiteturas, mecanismos de consenso, casos de aplicação e limitações quanto à escalabilidade e privacidade. Também foram abordadas questões práticas relacionadas aos *endpoints*, estruturas de nós e modelos de contratação de infraestrutura blockchain. A inclusão dos contratos inteligentes e das carteiras digitais revelou a importância desses componentes no desenvolvimento de aplicações descentralizadas seguras e eficientes. A discussão sobre tokens demonstrou sua função como representações digitais de ativos, detalhando padrões como ERC-20, ERC-721 e ERC-1155 e esclarecendo aspectos relacionados à estrutura, negociação, segurança e desafios normativos. Além disso, o capítulo apresentou de forma crítica os principais mecanismos de interoperabilidade, destacando tecnologias como sidechains,

relays, mecanismos notariais e contratos HTLC.

Assim, este capítulo consolida os fundamentos técnicos e conceituais necessários para sustentar as propostas da pesquisa, servindo como base para o desenvolvimento e avaliação das soluções apresentadas nos capítulos subsequentes.

Capítulo 3

Trabalhos Relacionados

Esta seção é dividida em três subseções, cada uma examinando um fluxo específico de estudos realizados e relacionados a este trabalho. A primeira subseção examina trabalhos relacionados com questões de utilização de tokens para diversos tipos de aplicação. A segunda subseção examina trabalhos com propostas e/ou implementações de interoperabilidade entre blockchains e a última subseção considera o impacto de custos e desempenho de plataformas na provisão de acesso e demanda de infraestrutura de nós em redes de blockchains. A análise dessas vertentes revela que existe um importante compromisso entre os custos e o desempenho da utilização de recursos computacionais.

3.1 Aplicações em Tokens

O artigo de Banaeian Far et al. (2022) explora o papel dos tokens não fungíveis como tecnologia emergente dentro da infraestrutura do blockchain, com foco especial em suas aplicações práticas tanto no mundo real quanto no metaverso. Os autores discutem como os tokens não fungíveis oferecem soluções inovadoras para autenticação de identidade, prova de propriedade digital e transferência segura de ativos únicos. Destacam o uso crescente de tokens não fungíveis para representar terrenos virtuais, itens de jogos e arte digital, além de abordar como esses tokens contribuem para a construção de economias virtuais e interoperáveis. O estudo conclui apontando áreas promissoras para desenvolvimento futuro, como governança descentralizada e integração com dispositivos IoT.

O trabalho Ma et al. (2024) apresenta um estudo focado exclusivamente em segurança no ecossistema de token não fungível. Com base em uma análise de 176 incidentes reais e 35 estudos acadêmicos, os autores criam uma taxonomia detalhada com 12 categorias de falhas, como ataques em contratos inteligentes, golpes de phishing e problemas de governança em marketplaces. O estudo também evidencia uma lacuna crítica entre as abordagens práticas e a pesquisa acadêmica, propondo um quadro de referência para mitigar vulnerabilidades. Esta contribuição é fundamental para informar desenvolvedores, investidores e formuladores de políticas sobre como proteger ativos em token não fungível

em um ecossistema de rápido crescimento.

Wang et al. (2021b) exploram a evolução dos tokens não fungíveis e fornecem uma análise dos componentes técnicos que sustentam os tokens não fungíveis, incluindo os padrões ERC-721 e ERC-1155. O trabalho cobre a estrutura de funcionamento de contratos inteligentes e como os tokens não fungíveis se integram com mercados e dApps. Além disso, discute os desafios em torno da interoperabilidade, padronização e acessibilidade de tokens não fungíveis. Também são abordadas questões regulatórias, como lavagem de dinheiro, e oportunidades para aplicações em jogos, música, arte e gerenciamento de IP. O estudo apresentado em Kräussl and Tugnetti (2022) foca na formação de preços e fatores econômicos que influenciam o valor de tokens não fungíveis. Os autores revisam modelos como regressão, machine learning e wavelet analysis para entender os principais fatores determinantes de preço, como escassez, utilidade, rede social e hype. Além disso, fazem uma crítica ao comportamento especulativo do mercado. Comparam tokens não fungíveis a ativos colecionáveis físicos e propõem critérios para avaliar sua viabilidade como classe de investimento.

Taherdoost (2023) realizou uma revisão sistemática da literatura entre 2012 e 2022, examinando 34 publicações relevantes. Apresenta uma visão do crescimento dos tokens não fungíveis, suas principais aplicações e desafios. As áreas analisadas incluem ciência da computação, economia, direito e regulação. O autor identifica lacunas, como a necessidade de melhores estruturas regulatórias, métricas de valorização e proteção contra fraudes. A pesquisa sistematiza o conhecimento atual e propõe perguntas de pesquisa para o futuro, como: “quais usos institucionais são possíveis para tokens não fungíveis?”.

A revisão apresentada por Hammi et al. (2023) cobre os aspectos técnicos, históricos e conceituais dos tokens não fungíveis. O artigo aborda desde os fundamentos de blockchain e contratos inteligentes até padrões como EIP-721 e EIP-1155. Também introduz o conceito de “tokens não fungíveis com royalties” e aplicações como bilhetes de eventos, arte digital e identidade digital. Um diferencial é a comparação entre o modelo monetário tradicional (moeda fiduciária) e o modelo de token não fungível. Além disso, os autores discutem questões legais, operacionais e éticas do uso de tokens não fungíveis.

Razi et al. (2024) oferece uma visão ampla em termos de escopo de aplicações. Os autores fazem um levantamento profundo das utilizações de tokens não fungíveis em setores como saúde, agricultura, identidade digital, cidades inteligentes, educação, caridade e propriedade intelectual. Explicam padrões técnicos, fluxos de criação de tokens e discutem problemas como consumo energético, direitos de governança e escalabilidade. Além de sistematizar aplicações existentes, o artigo propõe direções futuras para que tokens não fungíveis possam ser usados com segurança e eficiência em contextos institucionais.

A Tabela 3.1 mostra uma divisão em critérios, mediante as pesquisas apresentadas na seção dos trabalhos relacionados aos tokens não fungíveis. Os critérios para a

comparação são: Foco em Aplicações, Segurança, Modelos de Precificação, Revisão Sistemática, Padrões Técnicos e Infraestrutura e Desafios e Futuro.

Tabela 3.1: Comparativo de trabalhos relacionados aos Tokens não fungíveis

Critério	**Artigos**
Foco em Aplicações	Os artigos Banaeian Far et al. (2022), Hammi et al. (2023) e Razi et al. (2024) se destacam na apresentação de aplicações em múltiplos domínios como metaverso, saúde, etc.)
Segurança	O artigo Ma et al. (2024) é o único totalmente voltado à segurança de tokens não fungíveis
Modelos de Precificação	O artigo Kräussl and Tugnetti (2022) explora modelos financeiros e machine learning para análise de preços
Revisão Sistemática	Os artigos Wang et al. (2021b) e Taherdoost (2023) oferecem revisões detalhadas do estado da arte e lacunas
Padrões Técnicos e Infraestrutura	Os artigos Wang et al. (2021b) , Hammi et al. (2023) e Razi et al. (2024) detalham aspectos como os padrões ERC-721, ERC-1155 e blockchains alternativas
Desafios e Futuro	Os artigos Taherdoost (2023) e Razi et al. (2024) discutem os desafios de escalabilidade, regulamentação e sustentabilidade

3.2 Interoperabilidade de Tokens

A interoperabilidade entre redes blockchain é um campo de pesquisa recente, repleto de oportunidades para contribuições tanto na academia quanto na indústria. O avanço prático e significativo da usabilidade das blockchains depende do desenvolvimento de técnicas e soluções distintas. Seja por meio de abordagens baseadas em Cadeias, Pontes ou Aplicações Descentralizadas (DApp), ainda há diversas questões em aberto que precisam ser exploradas.

No trabalho de Wang (2021), uma revisão sistemática sobre os avanços e desafios da interoperabilidade entre blockchains foi apresentada. Os autores também abordam questões como diferenças estruturais entre transações e os desafios de manter propriedades de consistência e isolamento (ACID - Atomicidade, Consistência, Isolamento e Durabilidade) em operações entre redes distintas. Nesse trabalho, são discutidas ainda soluções práticas, como *atomic swaps* e protocolos de comunicação *cross-chain*. Assim, os autores não trazem uma metodologia concreta sobre o funcionamento dos protocolos.

Já o trabalho de Bellavista et al. (2021) propõe mais uma solução de interoperabilidade através de um esquema de transmissão baseado em *Trusted Execution Environment* (TEE) para fornecer melhores garantias de segurança. Os autores

apresentam também um protótipo que mede a latência e avalia o impacto de interações entre redes blockchains; contudo, os testes do trabalho só consideraram a latência da solução de interoperabilidade entre as plataformas, *Hyperledger Fabric* e *Sawtooth*.

O trabalho de Cao et al. (2024) propõe o protocolo denominado MAP, para interoperabilidade entre redes blockchain, destacando soluções baseadas em prova de conhecimento zero (*zk-proofs*) para melhorar a eficiência e reduzir custos. O trabalho justifica a criação do protocolo MAP, devido aos altos custos de transações *on-chain e off-chain* e problemas de escalabilidade dos protocolos existentes, mas não há avaliações de desempenho para a análise desses problemas.

Em Zhu et al. (2023), são discutidos os desafios da interoperabilidade em blockchain e é proposto um *framework* baseado em *side-chains* e pontes *cross-chain*. Os autores mostram também uma tabela comparativa de soluções de interoperabilidade (Cosmos, Polkadot, Aion, dentre outras), destacando características como: protocolo utilizado (p.ex., HTLC, *sidechains*, etc.); Mecanismo *cross-chain* de gerenciamento de segurança e tipo de blockchain usada (pública, privada, Consórcio). Apesar da contribuição, não foram discutidas metodologias para a avaliação de custos e desempenho dos protocolos de interoperabilidade mencionados neste trabalho.

Em Alhussayen et al. (2024) propõe uma técnica de interoperabilidade baseada em oráculos, projetada especificamente para plataformas blockchain permissionadas. Os autores apresentam a arquitetura da técnica e implementam um protótipo para demonstrar sua viabilidade, além de medir a latência das transações entre redes. O projeto conecta as plataformas *Hyperledger Fabric* e *Corda*. No entanto, embora a avaliação tenha focado em redes permissionadas e na latência das transações, várias questões permanecem em aberto, como o custo de processamento e a forma de cobrança durante as transações

O trabalho de Ghaemi et al. (2021) consiste em uma solução de interoperabilidade entre blockchains permissionadas baseada na arquitetura publicar/assinar. Essa abordagem visa facilitar a transferência de ativos e dados entre diferentes redes, que frequentemente operam de forma isolada, criando silos de informações e ativos. Os autores dessa solução implementaram um protótipo que integra diferentes redes blockchain, como o *Hyperledger Besu* (um cliente Ethereum) e duas versões distintas da plataforma *Hyperledger Fabric*. O desempenho da rede foi analisado usando uma ferramenta de *benchmark* para identificar os limites e gargalos da solução proposta; entretanto, na análise de desempenho dos autores, não há comparativos com as soluções de interoperabilidade já em uso.

3.3 Avaliação de Custos e Desempenho de Plataformas Blockchain

Existem disponíveis na literatura trabalhos que abordam a avaliação de custos e desempenho de plataformas blockchain, porém não realizando comparação com infraestruturas e plataformas diferentes. Grande parte desses estudos apresenta avaliações específicas para uma plataforma. Dessa maneira, os parâmetros e requisitos considerados impossibilitam a escolha da melhor plataforma e infraestrutura necessárias para as quais as aplicações serão projetadas. Portanto, esta seção sintetiza alguns trabalhos relacionados à avaliação de custos e desempenho de plataformas blockchain.

O trabalho desenvolvido por Rimba et al. (2020) investigou a questão do custo monetário de utilizar uma plataforma blockchain em comparação com uma infraestrutura de armazenamento em nuvem. Por meio de modelos de custo para processos de negócios, eles compararam os custos na plataforma Ethereum e no Amazon Simple Workflow Service (SWF). Os resultados apontaram uma grande variação de custo entre as duas soluções. O custo do blockchain Ethereum é, pelo menos, o dobro do dobro dos serviços tradicionais de nuvem fornecidos pelo Amazon SWF. Nosso trabalho se diferencia ao apresentar um modelo de custo para comparação da infraestrutura necessária para manter o provimento da plataforma blockchain, sendo ela pública ou permissionada.

Baliga et al. (2018); Thakkar et al. (2018); Wang and Chu (2020) analisaram o desempenho da plataforma Hyperledger Fabric. A abordagem de Baliga et al. (2018) utilizou a ferramenta Hyperledger Caliper sob diferentes configurações para avaliar a latência e a taxa de transferência do Hyperledger Fabric. Avaliaram também o desempenho variando o número de *chaincodes*, *channels* e *peers*. Concluíram que a taxa de transferência é sensível às configurações e que a latência é significativamente afetada pelo tamanho da carga experimentada. Thakkar et al. (2018) testou duas abordagens para avaliação de desempenho, otimização de cache e configuração de políticas de endosso. Como contribuição, os autores descreveram orientações sobre a configuração de parâmetros da rede e também os principais gargalos de desempenho. Nos estudos de Wang and Chu (2020), os autores caracterizaram o desempenho de cada fase do ciclo de vida de uma transação, sendo que a fase de execução mostrou boa escalabilidade de desempenho em políticas de endosso específicas. A fase de validação obteve desempenho pior porque a carga de trabalho de computação do nó de validação é pesada. Os resultados mostraram que o principal fator de desempenho foi a política de endosso, ou seja, quantos pares tiveram que aprovar uma transação.

Os artigos Leal et al. (2020); Rouhani and Deters (2017a); Zhang et al. (2020) fornecem avaliação de desempenho de redes blockchain privadas baseadas na plataforma

blockchain Ethereum de código aberto. Leal et al. (2020) avaliam o desempenho da rede utilizando um conjunto de dados para encontrar uma configuração ideal. Utilizaram diferentes custos, algoritmos de consenso e números de nós de rede para determinar a configuração. Como contribuição, é fornecida uma forma para encontrar uma configuração ideal para um determinado número de transações exigidas por um caso de uso. O trabalho de Rouhani and Deters (2017a) mostrou que o desempenho da rede Ethereum depende, além da configuração da rede, da implementação do cliente utilizado. O estudo mostra que o cliente Parity obteve desempenho significativamente melhor do que o cliente Geth. Em Choi and Hong (2021), os autores utilizaram o Hyperledger Caliper para avaliar a rede Ethereum. Os resultados mostram que o desempenho das transações pode diferir de acordo com seu conteúdo e configuração da rede.

Existem alguns estudos de análise de desempenho de Blockchain, que avaliam e comparam as plataformas Hyperledger Fabric e Ethereum. Em Monrat et al. (2020) é realizada uma análise de desempenho e escalabilidade, variando as cargas de trabalho, das plataformas Ethereum, Quorum, Corda e Hyperledger Fabric. A conclusão geral do trabalho é que o Hyperledger Fabric tem um desempenho superior às demais plataformas porque atinge o consenso de forma mais eficiente. Em Malik et al. (2019) é realizada uma comparação do desempenho das plataformas Ethereum e Hyperledger Fabric utilizando uma aplicação de comércio de energia e Hyperledger Caliper. A conclusão é que o Ethereum fornece a melhor solução para a aplicação em pequena escala, mas o Hyperledger Fabric pode ser mais adequado para aplicações de grande escala.

A Tabela 3.2 mostra a comparação da nossa abordagem com as pesquisas apresentadas nesta seção. Os critérios para a comparação são baseados em nossos objetivos para lidar com os aspectos da arquitetura de um nó em redes blockchain públicas e permissionadas que impactam o seu custo e desempenho da infraestrutura. São eles: a plataforma utilizada para a experimentação e avaliação, a avaliação de desempenho e a avaliação de custos de infraestrutura por transação.

3.4 Considerações Finais

A revisão dos trabalhos relacionados apresentada neste capítulo permitiu consolidar três vertentes fundamentais para a sustentação da pesquisa: aplicações de tokens não fungíveis, interoperabilidade entre Blockchain e avaliação de desempenho e custo de plataformas. Na primeira vertente, foi possível observar a evolução das aplicações em tokens, com destaque para os NFTs utilizados em ambientes virtuais, identidade digital e ativos físicos. Os estudos investigados revelam não apenas o potencial de uso, mas também os desafios em torno da segurança, regulação e viabilidade econômica, especialmente no contexto de

Tabela 3.2: Comparação deste trabalho aos trabalhos relacionados ao tema avaliação de custo e desempenho em Blockchains.

Trabalhos	Hyperledger Fabric	Ethereum	Desempenho	Custo
(Choi and Hong, 2021)	✗	✔	✔	✗
(Zhang et al., 2020)	✗	✔	✔	✗
(Leal et al., 2020)	✗	✔	✔	✗
(Rouhani and Deters, 2017b)	✗	✔	✔	✗
(Xu et al., 2021)	✔	✗	✔	✗
(Wang and Chu, 2020)	✔	✗	✔	✗
(Monrat et al., 2020)	✔	✗	✔	✗
(Thakkar et al., 2018)	✔	✗	✔	✗
(Baliga et al., 2018)	✔	✗	✔	✗
(Malik et al., 2019)	✔	✔	✔	✗
(Rimba et al., 2020)	✗	✔	✔	✔
Este trabalho	✔	✔	✔	✔

marketplaces e estruturas de governança descentralizada.

A segunda vertente abordou a interoperabilidade como um aspecto técnico e estratégico para ampliar o alcance das redes blockchain. Os trabalhos analisados demonstram diversidade de propostas teóricas e protótipos práticos, como o uso de sidechains, oráculos e protocolos baseados em provas de conhecimento zero. Embora muitas soluções ainda enfrentem limitações quanto à escalabilidade, à latência e ao custo, a busca por redes intercomunicáveis é um eixo central para futuras inovações.

A terceira vertente revelou a complexidade de avaliar o custo e desempenho das plataformas blockchain, evidenciando a falta de padrões comparativos entre redes públicas e permissionadas. Diversos estudos propuseram métricas e ferramentas de benchmarking que demonstram como configurações de rede, algoritmos de consenso e políticas de endosso impactam diretamente na eficiência e no custo por transação. Contudo, ainda persiste a necessidade de modelos que levem em conta o comportamento da infraestrutura em cenários variados e alinhados com aplicações reais.

Por fim, foram evidenciadas aplicações da blockchain em áreas como rastreabilidade e controle de acesso a dados, com base em estudos recentes e iniciativas concretas. A análise apontou benefícios como transparência, auditabilidade e segurança, mas também desafios a serem superados, incluindo privacidade, usabilidade, regulamentação e eficiência energética. Assim, este capítulo contribui para fundamentar as escolhas metodológicas e práticas da pesquisa, ao identificar lacunas, critérios de avaliação e direcionamentos relevantes para as propostas de solução apresentadas nos capítulos posteriores.

Capítulo 4

ARCHchain - Arcabouço de para avaliação de *endpoint* em infraestrutura Blockchain, Token de Cessão e Interoperabilidade

Este capítulo trata de elaborar e delinear uma metodologia de análise de custos em função do desempenho computacional de infraestrutura em redes Blockchain públicas. A elaboração do padrão desenvolvido estabelece uma referência para avaliação de nós provedores de acesso a plataformas blockchain baseadas em Ethereum por aplicações descentralizadas. Esses nós provedores são também conhecidos como **Gateways** ou **Endpoints** e são as portas de entrada e saída de dados em uma rede Blockchain. Uma vez que uma aplicação necessita propor uma transação em uma rede Blockchain, a mesma precisa realizar isso através de um nó provedor de acesso à rede.

Para delinear a metodologia de análise, foram produzidos cenários de experimentação da infraestrutura. Dois cenários distintos conduziram o escopo do planejamento da experimentação da metodologia. Primeiramente, foi proposto um cenário que utiliza o Token de Cessão (CST), apresentado na seção 4.2, como forma de submissão de transações para a avaliação da arquitetura. O CST é um token capaz de representar um ativo digital e ser **alugado** ou **cedido** de forma temporária a um usuário da rede. Portanto, o projeto do token de cessão foi implementado e utilizado como um dos contratos-teste para submissão de transações e avaliação do nó referido. Além do cenário dos tokens, também foi definido um segundo cenário em que são avaliados diferentes tamanhos de entrada em uma função de um contrato teste com complexidade de tempo $O(n)$. Para cada um desses dois cenários, são analisados, tratados e avaliados o desempenho e o custo do *Endpoint*. Portanto, o problema que trata este capítulo inclui a análise sobre as formas e custos de utilização da infraestrutura computacional pela tecnologia Blockchain, bem como as implicações provenientes dos requisitos do cenário de submissão de transações e complexidades computacionais.

4.1 ARCHchain

Esta seção foi dividida nas etapas de planejamento da metodologia de análise e planejamento dos cenários de experimentação. A etapa de planejamento da metodologia de análise de custo buscou criar uma arquitetura ampla e expansiva. Com a amplitude, a etapa pretende alcançar o aumento do nível de propensão e adoção da metodologia de avaliação e, além disso, alcançar outras plataformas de Blockchain por meio da expansibilidade. Em seguida, focou-se em identificar parâmetros e definir requisitos necessários ao desenvolvimento de cenários e experimentos. Os cenários experimentados contemplam a avaliação de testes exaustivos em infraestrutura computacional por meio de requisições em funções de contratos inteligentes. Os experimentos foram projetados de forma que possam ser reproduzidos e expandidos a outras plataformas para identificar quais os ganhos esperados na utilização da avaliação e definição da infraestrutura ideal para cada cenário.

Ainda neste capítulo, por meio dos conceitos elencados, desenvolveu-se um padrão de arquitetura genérica de rede Blockchain para resolução dos problemas encontrados na construção de redes de experimentação. É importante ressaltar que foram realizadas análises críticas e comparativas do estado empírico e atual da área a partir das pesquisas. A elaboração da etapa de construção de cenários de aplicações visa disponibilizar ainda referências de protótipos de prova de conceitos para que possam contribuir com o entendimento e o desenvolvimento de um ambiente que permita caracterizar a demanda por infraestrutura do nó *Endpoint*, considerando os diversos elementos envolvidos.

4.1.1 O Problema

As Blockchains possibilitam o registro seguro e descentralizado de dados ou transações entre entidades (pessoas e/ou organizações) que podem não se conhecer e assim não ter confiança mútua. Logo, os dados e transações entre essas entidades são registrados de forma imutável, com acesso público ou privado para fins de verificação de autenticidade e derivação de novas transações. Atualmente, os modelos de infraestrutura mais adotados para a tecnologia blockchain são compostos por nós com diferentes capacidades computacionais organizados em redes públicas ou permissionadas e cada uma com características de desempenho e custos específicos. Contudo, essa tecnologia encontra-se ainda em fase de amadurecimento e necessita de ferramentas para gerenciamento de custos e recursos computacionais (i.e., infraestrutura) que permitirão a sua adoção por organizações em setores como agronegócios, indústria, serviços e governos.

As redes blockchain públicas foram as primeiras a serem desenvolvidas e são ainda as mais utilizadas. Plataformas populares como Ethereum permitem o desenvolvimento e

execução de contratos inteligentes, sem restrição ao acesso ou uso desses recursos e constituem um intricado ecossistema de dApps. Contudo, transações nessas redes podem levar minutos para serem confirmadas dado o grande número de usuários que as submetem e o consenso distribuído realizado pelos nós mantenedores da rede para validar transações[1]. Esses nós têm permissão de gerar novos ativos (ou moeda) e adquiri-los (stake), assim como cobrar tarifa aos usuários por transação confirmada. Uma aplicação que submete a proposta de uma transação em uma rede pública requer fazê-lo através de um nó provedor de acesso. Estes nós provêm às aplicações uma forma de conexão com a rede, disponibilizada por serviços abertos em portas específicas de comunicação. Eles são nomeados de **Gateways** ou **Endpoints** da rede e irão encaminhar aos outros nós mantenedores pertencentes à rede as propostas de transações requisitadas pelos seus usuários. Portanto, a questão de pesquisa que norteia este capítulo trata de "Avaliar o custo de infraestrutura para provimento e utilização de aplicações descentralizadas em plataformas blockchain". Para responder a esta questão buscou-se levantar os custos e características da infraestrutura que devem ser avaliados em cada *Endpoint* por blockchains.

Esta seção apresenta uma arquitetura projetada para representar a estrutura de organização de nós em uma rede Blockchain, denominada Arquitetura Geral de Blockchain (ARCHchain). Essa arquitetura compõe uma rede com pares de nós organizados de forma genérica, isto é, pode abranger uma variedade de modelos e ser aplicada em diferentes implementações e plataformas blockchain.

As próximas subseções desta seção estão organizadas de forma a propor uma arquitetura de abrangência genérica e sua aplicação à plataforma pública Ethereum. A subseção 4.1.5 apresenta uma modelagem para avaliação de custos e desempenho do nó *Endpoint* configurado com base na arquitetura geral. Essa modelagem está sendo aplicada nas avaliações experimentais considerando a arquitetura para uma plataforma de rede pública no Capítulo 5. Na seção 4.1.4 são apresentadas referências que orientam a expansibilidade do modelo de arquitetura geral.

4.1.2 Definições ARCHchain

A ARCHchain é aplicada para avaliação em uma rede Blockchain pública no Capítulo 5 e demonstrada a possibilidade de expansão e aplicabilidade da sua utilização para outras plataformas específicas. Dessa forma, é possível equiparar os recursos utilizados de diferentes plataformas, a fim de comparar os custos essenciais para sua operação. Inicialmente é introduzida a descrição da arquitetura geral e, a seguir, é demonstrado como ela se aplica à plataforma padrão Ethereum e à extensão para a plataforma Hyperledger Fabric.

[1]Desempenho da rede Ethereum em tempo real: https://etherscan.io

O objetivo dessa proposta é unificar diferentes arquiteturas de rede blockchain (*e.g.*, redes públicas e permissionadas) para analisar requisitos de custo e desempenho de se participar dessas redes em termos de infraestrutura básica (i.e., um nó da rede). A partir dessa arquitetura unificada, é possível focar nos aspectos essenciais da tecnologia Blockchain, buscando reduzir a complexidade das análises de custo e desempenho e, ao mesmo tempo, mantê-las realistas em termos de comparação entre plataformas.

A arquitetura geral para um nó participante de uma rede blockchain é mostrada na Figura 4.1. Nessa figura, os componentes verticais em linhas contínuas representam nós da rede mantidos por uma entidade (pessoa ou organização) que participa da rede blockchain. O nó é um computador físico ou uma máquina virtual em serviços de computação em nuvem administrados pela entidade. Por sua vez, os componentes horizontais em linhas tracejadas representam os protocolos de *consenso* e par a par (*P2P*), elementos básicos para o funcionamento de uma rede blockchain que devem estar contidos em cada nó da rede. Nas principais implementações de blockchains atuais, esses dois elementos podem ser modularizados como dois processos diferentes, aproveitando as tecnologias de isolamento de recursos de computação leve como contêineres.

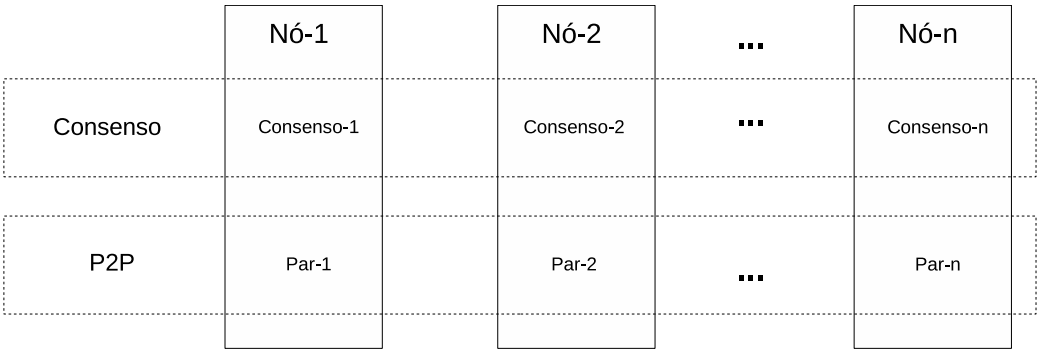


Figura 4.1: A arquitetura geral para um nó de rede blockchain. Fonte: elaborado pelo autor

O protocolo de consenso define as regras para a escolha do nó líder da vez, i.e., periodicamente escolhido, responsável pela construção do próximo bloco de transações a ser replicado para os demais nós da rede. Existem diferentes protocolos de consenso como, por exemplo, Prova de Trabalho (POW) e Prova de Participação (POS). Esses protocolos de consenso são adotados em redes blockchain públicas como *Bitcoin e Ethereum*. *Raft* e *Kafka* são serviços de ordenação de transações que funcionam como protocolos de consenso em redes blockchain permissionadas como Hyperledger Fabric (Greve et al., 2018).

O protocolo *P2P* é responsável pela comunicação entre os nós, estendendo-se também às etapas de processamento de transações pelos nós da rede. A comunicação geralmente segue protocolos *Gossip*, onde os nós obtêm uma lista limitada de parceiros e estabelecem conexões entre eles formando uma rede sobreposta para a difusão de blocos de transações.

O processamento desses blocos varia de acordo com a plataforma, sendo que Bitcoin e Ethereum adotam a estratégia ordenar-executar blocos, ao passo que Hyperledger Fabric adota a estratégia executar-ordenar-validar blocos (Androulaki and et al., 2018).

4.1.3 Aplicando ARCHchain para Rede Pública

Ethereum é atualmente a segunda maior rede pública de blockchain do mundo em arrecadação de fundos, portanto, uma representante importante desse modelo de rede[2]. A arquitetura básica do Ethereum é composta por dispositivos que executam software para verificar e manter as transações organizadas em blocos. Esses nós são computadores que executam os clientes Ethereum e que permitem que eles se conectem uns aos outros. Os clientes Ethereum são responsáveis por verificar se os dados inseridos ou solicitados por meio de transações cumprem as regras impostas pelo protocolo da rede. Existem dois tipos de clientes que são estabelecidos nas camadas de execução e consenso da rede. Esses clientes são interdependentes e devem ser executados de maneira conjunta, podendo ser em hosts separados, para fornecer acesso à rede.

Primeiramente, conforme ilustrado pela figura 4.2, um nó da rede recebe e executa as transações enviadas para a rede por um cliente (1), por meio da camada de execução, mantendo o banco de dados que representa o estado atual da rede. A camada de consenso implementa o algoritmo de consenso para validação de dados de acordo com o estado da rede mantido pelos clientes em execução. No algoritmo de consenso de prova de participação, ou em inglês, Proof of Stake – POS, se um nó da rede quiser se tornar um validador, ele primeiro deve enviar uma taxa de validador (2) e quando a transação for confirmada, ele poderá apostar algumas moedas para competir com outros validadores (3). Por sua vez, cada nó é responsável por transmitir as transações que recebe dos clientes aos outros nós (4). Quando uma quantidade suficiente de transações é recebida, os validadores elegem um líder, dentre os participantes com os maiores valores de moedas apostadas, sem garantia de escolha do que tem maiores apostas. O líder eleito então cria um bloco e o transmite para a rede (5), onde cada nó valida o bloco, executa todas as transações do bloco e adiciona o bloco à cadeia (6). O bloco também possui uma transação de recompensa especial, sendo que o líder da rodada recebe como recompensa as taxas de transação inferidas nas transações presentes no bloco.

Manter a propriedade de nós da rede para execução e consenso, em uma rede pública, oferece os benefícios da independência de terceiros, ao fornecer acesso à rede por meio de *Endpoint* próprio. Além desses benefícios, ajuda a promover a descentralização esperada da rede. O fato da mudança do uso do algoritmo de consenso para o POS resultou em uma redução significativa do consumo de recursos em comparação ao algoritmo anterior, POW,

[2]Mais informações podem ser encontradas em *https://ethereum.org/en/developers/docs/*.

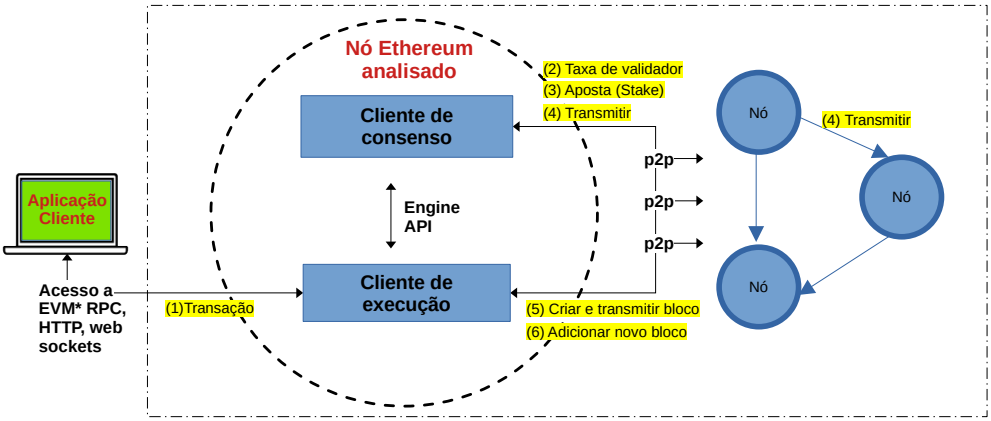


Figura 4.2: Modelo de rede pública Ethereum. Fonte: elaborado pelo autor

e, portanto, um menor custo para manter um nó da rede. Desta forma, a viabilidade para que uma aplicação tenha acesso aos dados disponíveis na Blockchain depende estritamente destes nós, que recebem as solicitações de transações, por meio da camada de execução e as submetem ao consenso da rede.

4.1.4 Expandindo a ARCHchain para Rede permissionada

A possibilidade de expansão é uma característica fundamental para o desenvolvimento de soluções tecnológicas de qualidade. O projeto de desenvolvimento da ARCHchain foi elaborado com o propósito de ampliação para outras plataformas. Portanto, a ARCHchain é passível de ser estendida para outras plataformas além do modelo de rede pública Ethereum demonstrado na seção 4.1.3 e experimentado no Capítulo 5. O Hyperledger Fabric é um grande projeto de código-fonte aberto envolvendo mais de 35 organizações e 200 desenvolvedores[3]. Como exemplo, a plataforma de rede permissionada Hyperledger Fabric, que é uma das mais populares atualmente desse tipo de rede, pode ser enquadrada no modelo genérico de nós de rede blockchain e ser avaliada mediante os testes de desempenho e obtidos os custos da infraestrutura. No artigo de Mendonça et al. (2023) foram apresentados experimentos envolvendo a análise e comparação dos custos das plataformas Hyperledger Fabric e Ethereum. Nele também é possível verificar a aplicabilidade da ARCHchain para a plataforma Hyperledger Fabric, onde se pode averiguar os custos dos nós mediante o desempenho e a demanda pelos recursos. Assim como no Ethereum, podemos organizar a estratégia em dois elementos essenciais para a arquitetura, que são os componentes P2P e consenso. Contudo, o componente P2P neste contexto tem atribuições extras. Em linhas gerais, os pares primeiramente executam uma transação, i.e., simulam seu processamento e proveem o endosso da transação para a aplicação cliente, e

[3]Mais informações podem ser encontradas em *https://hyperledger-fabric.readthedocs.io*.

posteriormente os pares validam as transações, mantendo-as na estrutura de dados encadeada da blockchain. Por sua vez, os ordenadores participam da estratégia após a etapa de execução para realizar o consenso, i.e., determinar o líder que ordena as transações em um novo bloco e posteriormente enviar o bloco aos pares para a etapa de validação.

A Figura 4.3 ilustra o fluxo de uma transação no Hyperledger Fabric e os componentes P2P e consenso representados por pares e serviço de ordenação, respectivamente. Inicialmente, um par recebe a proposta de transação da aplicação cliente (1). O par então simula a execução da transação invocando o contrato inteligente que a corresponde e envia uma mensagem de endosso ou sua negativa para a aplicação cliente (2). A aplicação aguarda endossos de outros pares, conforme a quantidade configurada na rede, e então envia a transação para o serviço de ordenação (3). A seguir, esse serviço recolhe transações da rede até alcançar o tempo (*timeout*) ou a quantidade de transações limite para gerar um novo bloco. Os blocos são então encaminhados para os pares da rede realizarem a validação (4), que consiste no encadeamento do bloco à blockchain e na atualização do seu estado global para consultas rápidas, p.ex., variáveis dos contratos inteligentes ou saldos de contas.

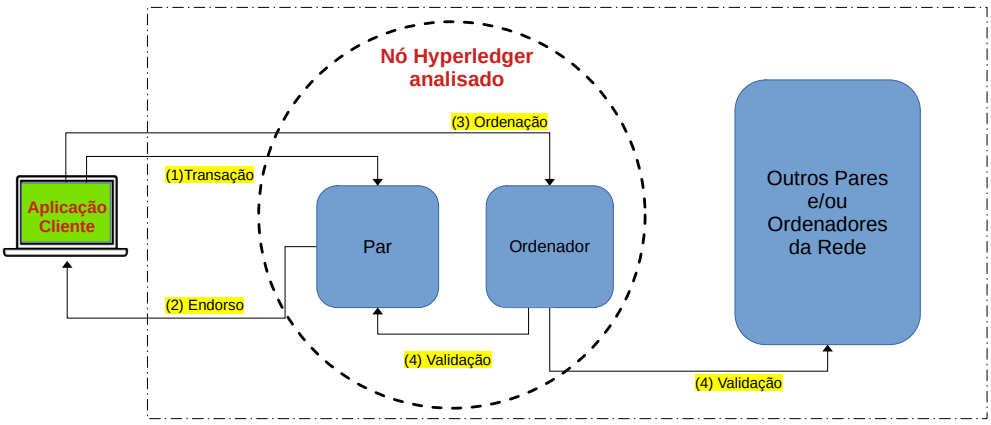


Figura 4.3: Componentes e fluxo de transações na plataforma Hyperledger Fabric. Fonte: elaborado pelo autor

Há alguns pontos importantes para observar na arquitetura particular do Hyperledger Fabric que permite também representá-lo pela arquitetura geral da Figura 4.1. Primeiro, execução e validação (passos 2 e 4) são realizadas pelos pares da rede, i.e., o componente P2P, ainda que estejam em etapas diferentes no fluxo da transação. Por sua vez, o serviço de ordenação é constituído por uma coleção de ordenadores – os serviços Raft e Kafka adotados no Hyperledger Fabric requerem ao menos três ordenadores – que determinam o líder da vez para a geração do novo bloco, i.e., o componente consenso. Logo, um nó participante de uma rede blockchain permissionada Hyperledger Fabric pode conter os componentes, par e consenso, como mostrado na arquitetura geral (Figura 4.1) em concordância com a arquitetura particular e fases de transações dessa rede (Figura 4.3).

4.1.5 Modelagem dos custos

Nesta seção é apresentado o modelo de obtenção de custos e desempenho por transações, baseado na representação de redes públicas e/ou permissionadas pela ARCHchain. A modelagem prioriza alcançar os custos fixos para obter parâmetros que sugerem uma tomada de decisão pela infraestrutura mais adequada para cada situação. Os custos variáveis, como por exemplo, as tarifas em redes públicas ou gastos com pagamento de ativos, são suprimidos do modelo, pois não se referem à infraestrutura e normalmente são os usuários das aplicações que se responsabilizam por essas tarifas. Sendo assim, o modelo baseou-se na seguinte questão de pesquisa:

> *Qual o custo da infraestrutura mínima necessária para implantar uma aplicação descentralizada, utilizando um nó Endpoint próprio para uma rede blockchain pública e/ou permissionada?*

Desta forma, os custos pretendidos são valores monetários gastos perante a utilização de recursos computacionais, com o objetivo de processar dados de uma blockchain, por meio de aquisição de bens ou serviços. Esses custos estão diretamente relacionados à capacidade dos recursos de processar as transações. Portanto, os custos são calculados com a soma dos valores monetários despendidos com o pagamento de bens ou serviços e de acordo com o desempenho dos recursos em processar uma certa quantidade de transações, a $\boldsymbol{Vazão\ (Vz)}$, e tempo de resposta da rede, a $\boldsymbol{Latência(L)}$. O cálculo da $\boldsymbol{Vz}$ está representado na Equação 4.1 e o da $\boldsymbol{L}$ na Equação 4.2.

A vazão é calculada pela razão entre o valor total do número de transações confirmadas $\boldsymbol{(TTC)}$ e o tempo total medido $\boldsymbol{(TT)}$. As transações confirmadas são as transações que a blockchain conseguiu processar sem erro e retornou o *hash* da transação. Já o tempo total é o tempo medido no início do envio das transações até o final do tempo de retorno de todas as transações.

$$\boldsymbol{Vazão\ (Vz) = \frac{Total\ de\ transações\ confirmadas\ (TTC)}{Tempo\ total\ (TT)}} \tag{4.1}$$

A latência é calculada pela diferença entre o início e o final do tempo de resposta medido. O tempo medido no instante em que inicia o envio é contado e ao final do envio é contado o tempo para que possa subtrair o tempo no envio $\boldsymbol{(Te)}$ pelo tempo na confirmação $\boldsymbol{(Tc)}$.

$$\boldsymbol{Latência(L) = Tempo\ na\ confirmação\ (TC) - Tempo\ no\ envio\ (Te)} \tag{4.2}$$

A partir da análise dessas variáveis de desempenho apuradas, foi proposto um modelo que estima o custo por transação considerando uma aplicação típica para a tecnologia

blockchain ao realizar inserção e/ou consulta de registros em redes blockchain. Existem algumas particularidades de cada tipo de rede que afetam os custos fixos e variáveis e devem ser analisadas de maneira bastante criteriosa ao comparar cada tipo. Esse modelo tem o objetivo de mensurar uma infraestrutura de *endpoint* de custo mínimo $(\boldsymbol{custo_{ideal}})$ que alcança a vazão máxima $(\boldsymbol{Vz_{ideal}})$ para a carga de trabalho avaliada. Nesse sentido, o modelo foi formalizado mediante as seguintes equações:

Primeiramente são informados a carga de trabalho distribuída na rede, representada por $(\boldsymbol{w})$, e um conjunto de tipos de recursos computacionais disponíveis $(\boldsymbol{R})$, no qual tem-se o tipo de recurso caracterizado por $\boldsymbol{r_i = (cpu_i,\ memória_i\ ,\ custo_i) \in R}$. Logo, uma rede blockchain $(\boldsymbol{B})$ é considerada composta de nós com configuração uniforme onde $\boldsymbol{B = r_i \in R}$.

A função apresentada na Equação 4.3 retorna o conjunto de todas as vazões máximas $(\boldsymbol{t_i})$ para cada tipo de recurso $(\boldsymbol{r_i})$ como nó $(\boldsymbol{b \in B})$. Desta forma, obtém-se pela Equação 4.4 o conjunto alvo.

$$\boldsymbol{Maximiza\_Vazão(R, w, B) = T} \tag{4.3}$$

$$\boldsymbol{A = \{t_i > w | t_i \in T\}} \tag{4.4}$$

Por outro modo, a função da Equação 4.5 retorna o recurso com menor custo para a carga $(\boldsymbol{w})$ em $(\boldsymbol{A})$, onde $(\boldsymbol{r_{ideal}})$ representa o recurso com $\boldsymbol{Custo_{ideal} \in (A)}$ e também a vazão máxima $(\boldsymbol{Vz_{ideal}}) \in (\boldsymbol{A})$.

$$\boldsymbol{Minimiza\_Custo(A) = r_{ideal}} \tag{4.5}$$

Portanto, o custo de uma transação $(\boldsymbol{Custo_{transacão}})$ na blockchain $(\boldsymbol{B})$ para a carga $(\boldsymbol{w})$, considerando o conjunto de tipos de recursos computacionais $(\boldsymbol{R})$, é dado pela Equação 4.6, onde o menor valor encontrado representa o equilíbrio entre a demanda pelo *Endpoint* e a vazão.

$$\boldsymbol{Custo_{trans} = \frac{Custo_{ideal}}{Vz_{ideal}}} \tag{4.6}$$

Baseados na ArchChain e na modelagem da observação de variáveis para o levantamento de custo por transação, foram conduzidos experimentos, descritos no Capítulo 5, que demonstram a aplicabilidade da ArchChain e a capacidade do modelo de custos de extrair os parâmetros e demonstrar o custo-benefício da escolha de infraestrutura em cada demanda de aplicação de transações.

4.2 Token de Cessão

Nesta seção serão apresentados os conceitos desenvolvidos e tecnologias relacionadas ao modelo proposto do Token de Cessão (CST). O projeto e implementação do token envolvem a documentação, codificação, testes e implantação de contratos inteligentes em uma plataforma Blockchain pública. O CST tem como finalidade principal representar um ativo digital que possa ser **alugado** ou **cedido** de forma temporária a um usuário. Essa cessão ou aluguel pode também ser **revogado** de acordo com o entendimento das partes. Ainda como finalidade, a posse do CST pode ser **validada** durante uma cessão temporária com o intuito de prova de direitos temporários sobre o token. Será apresentada ainda nesta seção a extensão dos requisitos do CST para prover a interoperabilidade do token com diferentes plataformas blockchain, permitindo a validação de posse em multiplataformas. Essa extensão amplia o projeto do arcabouço de interoperabilidade com implementação, implantação e testes do token, juntamente com o mecanismo de enlace para interoperar com o CST.

4.2.1 O Problema

O termo tokenização advém de um esforço em digitalizar uma grande quantidade de demandas e refere-se a uma forma de distribuição de tokens que permite a representação e o acesso a ativos como uma ferramenta para autonomia organizacional e econômica. Lotti (2019) cita diversos ativos como imóveis, pedras e metais preciosos, poder computacional, dados, obras de arte, ingresso em eventos, entre outros, de maneira que se apresentam como uma proposta de valor atraente para o ecossistema descentralizado. Em blockchain, a tokenização é realizada por meio da criação de tokens provenientes de contratos inteligentes. Estes contratos permitem a automatização de negociações de tokens trazendo maior eficiência, transparência e constatação dos registros de dados nas transações realizadas. Portanto, o processo de tokenização depara com a questão de como ceder e revogar um token em blockchain de forma temporária, uma vez que há a premissa de imutabilidade dos dados registrados.

Encontrar a solução desta questão gera uma maneira de oferecer um instrumento capaz de controlar o consentimento de direitos de propriedade de tokens por terceiros e por um determinado tempo. Desta forma, este instrumento deve ainda validar as permissões concedidas, normatizado por um processo padronizado, visando identificar de forma simples as concessões e revogações. Como resposta ao problema, desenvolveu-se uma solução que evoluiu com a possibilidade de atribuição de propriedade de tokens a uma cessão, por meio de transferência de propriedade, e determinando um tempo em que a cessão deverá ser revogada. Ainda como prova de conceito, por meio de verificação de desempenho e aplicabilidade, responder se é possível escalar o controle dos tokens em

um ambiente de redes Blockchain.

A interoperabilidade também consta como um problema da distribuição de posse temporária de tokens. Desta forma, a solução deve ser capaz de utilizar mecanismos para interoperar a concessão e revogação de tokens entre blockchains distintas. A partir da obtenção dessa solução, como continuidade, dedica-se a proceder com a avaliação da viabilidade e dos impactos para prover a interoperabilidade do instrumento desenvolvido para conceder e revogar o direito de propriedade de tokens. Nesse propósito, um arcabouço foi definido para fornecer a integração de aplicações no contexto da solução e avaliação a partir de experimentos em redes blockchain.

Os tokens são considerados uma representação de um bem, como o direito de propriedade, e, quando baseados em blockchain, são registros individuais que estão vinculados a um endereço ou uma conta de um determinado usuário. Apenas o usuário proprietário do token pode exercer o domínio sobre o endereço por meio da sua chave privada (Konashevych, 2020). Os registros de tokens resultam em uma transação na blockchain, assim como as suas transferências de propriedades. A criação e o gerenciamento da propriedade dos tokens também são realizados por meio dos contratos inteligentes.

Os tokens contêm inúmeras finalidades em suas diversas variações conforme apresentado na seção 2.3. Utilizar os tokens para construir conhecimento por meio de processamento dos dados ou até mesmo verificação pontual de propriedade é mais um importante propósito de aplicação. No propósito de averiguação de posse temporária, é apresentado aqui o **Token de Cessão**. Este token caracteriza-se como uma variação de utilização dos tokens para conceder e suspender a sua posse por meio de uma cessão controlada por tempo de permissão de utilização. O padrão de desenvolvimento do Token de Cessão (CST) é por meio de uma extensão do padrão ERC-721. O CST apresenta a possibilidade de atribuir a posse de um determinado token a uma cessão por meio de seu identificador e do endereço do usuário. Esta cessão deve determinar um tempo em que ela será automaticamente expirada ou ainda receber a solicitação, por meio apenas do detentor da posse, de retornar ao seu proprietário de origem.

Como exemplos de utilizações do CST, tem-se o caso de empregá-los em sistemas que utilizam a posse de credenciais para liberar algum tipo de acesso, como o login em um software ou a entrada em uma catraca física de controle de acesso. A inserção de credenciais no formato de sistemas centralizados, para um determinado indivíduo, seria uma solução mais óbvia para este problema. Desta forma, o detentor de posse do software ou da catraca iria garantir que somente com as credenciais o acesso seria liberado. Porém, os usuários dos sistemas em questão não teriam nenhuma garantia de que o seu acesso se daria da forma em que lhes foi concedida a credencial. Podendo o detentor do sistema revogar a permissão a qualquer momento. Para a solução deste impasse, as duas partes necessitam de alguma forma de administrar e garantir o que lhes é devido. Desta forma, a

utilização da posse do CST, como meio para garantir as credenciais de acesso temporárias, resolve o problema da administração das credenciais e a confiabilidade entre as partes. Neste contexto, quando o proprietário do sistema quiser ceder as credenciais de acesso de forma a um registro público e descentralizado ao usuário, basta ele emitir um token e o ceder temporariamente. O usuário a partir desse ponto, tendo posse do token, é capaz de exercer as ações consentidas e associadas ao token em específico por meio da verificação de sua posse.

Algumas das ações que um usuário pode realizar com um CST são diferentes de outros tokens que utilizam o mesmo padrão ERC-721. Em tokens baseados neste padrão original, geralmente é permitido ao detentor da posse do token realizar a transferência de posse a qualquer outro usuário e em qualquer circunstância. Porém, no caso do CST, o direito de transferência de propriedade é controlado e permitido apenas por meio das funções internas do contrato inteligente. Visto que a aplicabilidade do CST pode beneficiar os lados envolvidos no problema em questão, e os requisitos e os atores que atuam diretamente com o sistema podem ser identificados, os próximos passos são descrever a arquitetura de software para rede Blockchain, envolvendo o projeto, a implementação e os testes. Esta arquitetura foi projetada com a finalidade de permitir que uma aplicação possa validar a posse de tokens temporários e permitir ou impedir o uso de recursos especificados na validação da cessão.

A partir da definição e descrição do projeto de arquitetura, faz-se necessário distinguir os perfis de utilizadores e de comportamentos e permissões para com cada componente. Desta forma, os usuários e comportamentos que exercem ações sobre os componentes da arquitetura projetada para o CST são: o **(Cedente)**, que é proprietário do contrato do CST, o **(Cessionário)**, que é o utilizador da cessão do CST, e o **(Enlace)**, responsável por gerenciar as cessões de posse concedidas, revogadas e expiradas. Dessa forma, as aplicações com requisitos de verificação de posse temporária terão atores com perfis distintos conforme a atuação que cada papel deve desempenhar. A Figura 4.4 apresenta os atores com o papel que cada um pode exercer nas transações realizadas pelas funções dos contratos e os CSTs. Os atores estão representados como Cedente, Enlace e Cessionário.

O Diagrama da Figura 4.5 foi elaborado com o intuito de distinguir as responsabilidades de cada ator das funcionalidades presentes em todo o ecossistema para a manutenção do CST. Uma distinção do que ocorre dentro ou fora da rede blockchain também está representada neste mesmo diagrama. A essa separação são nomeadas de **On-chain** as transações que são registradas diretamente na Blockchain e **Off-chain** as operações que são processadas por uma aplicação ou até mesmo em uma outra rede blockchain externa. Portanto, uma transação On-chain normalmente é registrada a partir de processamento de dados por uma solução Off-chain que gera uma transação.

Os componentes *On-chain* nesse diagrama são formados pelos contratos inteligentes

Atores	Comportamentos
 Cedente	• Implantar contrato do token de cessão (CST) • Cunhar token • Transferir propriedade do token cunhado para enlace e definir tempo de cessão
 Enlace	• Implantar contrato Enlace • Disponibilizar token para cessão • Gerenciar cessões
 Cessionário	• Receber token • Disponibilizar token para verificação de posse • Devolver token com prazo expirado

Figura 4.4: Ações exercidas por cada ator. Fonte: elaborado pelo autor

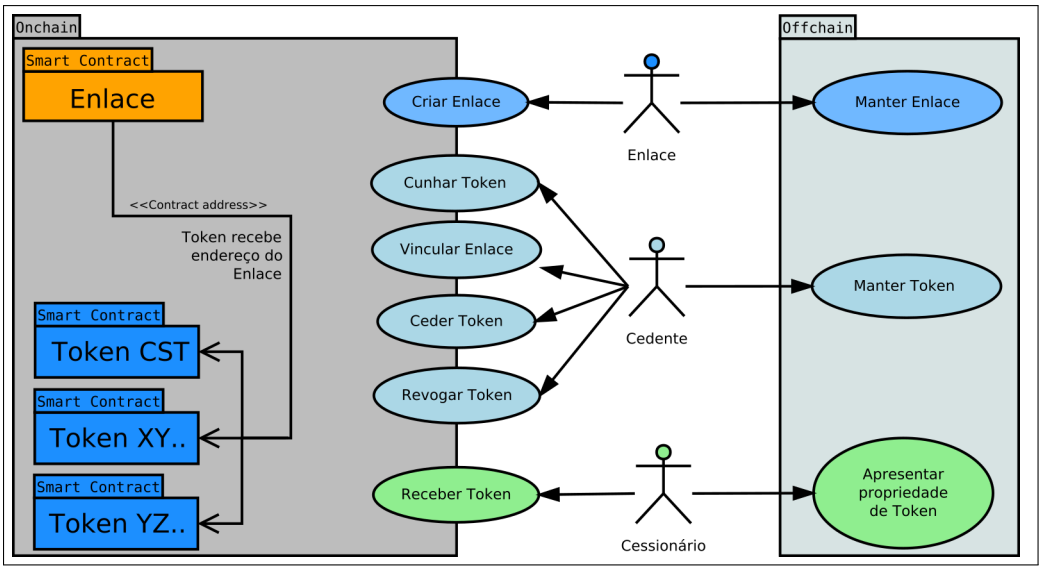


Figura 4.5: Diagrama representando a divisão dos componentes On-chain e Off-chain em conjunto com as interações entre atores e funções. Fonte: elaborado pelo autor

integrados com o propósito de controlar a emissão, concessão e revogação da posse dos tokens temporários. Já os componentes *Off-chain* estão formados por sistemas externos que são alimentados pelos dados dos componentes *On-chain*. O ator Cedente é o proprietário do contrato CST e quem tem a permissão de gerar os tokens. Ele exerce o papel de cunhar cada um dos tokens e oportunamente transferi-los para o Enlace. Ao transferir o token ao Enlace, deve ser definido o tempo de duração da cessão e meios para incorporá-lo a um determinado sistema e confirmação de posse. Esse sistema é uma solução externa à Blockchain, que tem a função de consultar e validar a propriedade do token para alguma finalidade específica e pré-determinada. Cabe ao Cedente gerenciar a emissão e as funções de ceder, revogar e validar a data de expiração de cada token criado. Sendo que a função de verificar a data de expiração deve proceder também de maneira que o token volte para a posse original quando a data de cessão já estiver

expirada. Ao ator Cessionário, cabe apenas manter a propriedade do CST em sua carteira ao receber o token e utilizá-lo para apresentação da posse quando for solicitado por funções vinculadas a este token. O Cessionário mantém a propriedade de um determinado token apenas durante a cessão concedida a ele, não dispondo da capacidade de transferi-lo a outro usuário.

4.2.2 Formulação dos Contratos Inteligentes

Os contratos inteligentes são sobretudo os recursos fundamentais para a composição e o desenvolvimento dos tokens e não é diferente para o CST. Os contratos mencionados assemelham-se ao conceito de classes em linguagens de programação orientadas a objetos, isto é, cada contrato pode possuir um construtor, variáveis com armazenamento persistente e métodos que manipulam estas variáveis. Embora haja similaridade com as classes, os contratos possuem particularidades intrínsecas, como por exemplo os eventos, manipuladores de erros e modificadores, que proporcionam vantagens de segurança ao tratarmos de boas práticas de modelagem de contratos. Os contratos são primeiramente modelados, têm seus recursos internos e seus manipuladores definidos e os eventos descritos. No que diz respeito à segurança de um contrato inteligente, apesar de existirem outras técnicas de modelagem, destaca-se a utilização de modificadores e tratamentos de erros de uma maneira eficiente e segura.

Os eventos são recursos muito utilizados em contratos e têm o objetivo principal de transmitir informações para as aplicações off-chain. Além disso, os eventos são úteis para retornar informações em funções que geram transações na Blockchain, pelo fato de os dados de um evento poderem ser acessados por meio dos *logs* da transação. Outros recursos difundidos na construção de um contrato são os manipuladores de erros, que são funcionalidades previamente definidas pela linguagem para tratamento de exceções ou para avaliar condições prévias que devem ser atendidas. Na formulação dos contratos do CST foi utilizado o manipulador *require*, que avalia se uma condição é falsa, para então disparar uma exceção e finalizar o fluxo principal do escopo em que o manipulador foi chamado. Adicionalmente, uma mensagem customizada pode ser retornada, caso a mesma esteja definida. No que tange aos modificadores criados internamente em contratos, os mesmos são semelhantes aos modificadores padrão da linguagem que definem quais regras a função deve seguir em seu fluxo de execução. Novos modificadores podem ser criados dentro dos contratos para serem utilizados em determinadas funções e delimitarem o uso das mesmas, como por exemplo, especificar quais funções apenas o dono do contrato poderá invocar.

Padrões de Tokens e Bibliotecas OpenZeppelin

Os padrões de tokens da plataforma Ethereum tornaram-se referência para a modelagem de contratos inteligentes de tokens, e juntamente com eles surgiram as demandas por prover segurança. Neste contexto, os padrões da OpenZeppelin[4] introduziram interfaces e contratos devidamente testados para que o processo de desenvolvimento de contratos seguros e eficientes se tornasse mais simples.

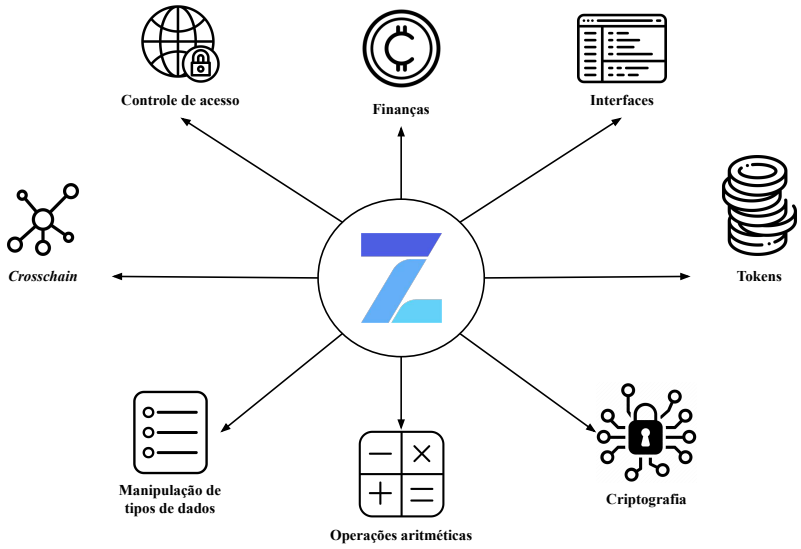


Figura 4.6: Subáreas de segurança abrangidas pela biblioteca de padrões ERC no OpenZeppelin. Fonte: elaborado pelo autor

A biblioteca OpenZeppelin contempla contratos e interfaces para diversos tipos de aplicações, além de possuir implementados os próprios padrões ERC. Na Figura 4.6 é possível observar como essa biblioteca perpassa várias aplicações importantes que podem ser desenvolvidas para Blockchain e os tokens. O primeiro passo para modelar o contrato inteligente de forma segura e eficiente baseia-se em importar os contratos da biblioteca OpenZeppelin.

Processo de desenvolvimento e codificação do CST

Os contratos inteligentes, que integram a solução para o token temporário, foram desenvolvidos de acordo com a lógica necessária aos requisitos do padrão de token ERC-721 e ao projeto do CST. Na sequência desta seção são dispostos os códigos e descrições que contêm todos os detalhes sobre a implementação e lógica de negócio dos contratos.

O processo de desenvolvimento do CST utiliza as boas práticas de modelagem apresentadas, em que foi planejado obter um contrato capaz de criar tokens com

[4] https://docs.openzeppelin.com/

funcionalidades disponíveis para serem utilizadas por meio de uma aplicação descentralizada e independente de plataforma. Os contratos foram desenvolvidos por meio da linguagem Solidity e, além da implementação e execução destes contratos, são apresentados códigos de testes e invocações a partir de uma API construída para uma aplicação descentralizada que foi desenvolvida como "Caso de Uso"no ambiente de desenvolvimento configurável Hardhat [5].

Inicialmente foi definida a interface do contrato, apresentada na Listagem 4.1, para servir de base para toda implementação do CST e que herda as funcionalidades do padrão ERC-721. Esta interface contém as assinaturas dos métodos e eventos obrigatórios, passando pela emissão de mensagens da criação de tokens e um construtor que deve receber o endereço do contrato de enlace como parâmetro. Também constam os métodos para consulta do endereço do enlace vinculado ao token, bem como o método para realizar a transferência do token para o enlace ou proprietário originário no caso de tempo expirado de uma cessão.

```
1  // SPDX-License-Identier: MIT
2  pragma solidity ^0.8.27
3  interface IERC0x0x {
4      event TokenId(uint256);
5      function criarNovoToken() public{}
6      function transferirTokenExpirado(address, address, uint256) external {}
7      function obter_enlace() public view returns (address){}
8      function burn(uint256) external{}
9      constructor(address) ERC721("Cessao", "CST"){}
10 }
```

Listagem 4.1: Padrão para Interface do CST

O código apresenta primeiramente a definição, na linha 1, do tipo de licença de distribuição de software utilizada. A versão utilizada do interpretador da linguagem *Solidity* está na linha 2. O nome da interface é definido na linha 3 e na linha 4 contém o evento que será disparado quando um novo token é gerado, retornando o seu número de identificação. Na linha 5 está escrita a assinatura da função que gera um novo token, na linha 6 a assinatura da função que transfere um token que está com o prazo de concessão expirado para o enlace e por último na linha 7 a assinatura da função que obtém o endereço do enlace a qual o contrato do token está vinculado.

Após a definição da interface com os métodos necessários para o contrato CST, seguem os códigos com a implementação da interface apresentada na Listagem 4.1. Conforme já mencionado, inicialmente deve-se importar as bibliotecas necessárias dos repositórios da OpenZeppelin. Na Listagem 4.2 estão escritas as importações do contrato do *"Padrão ERC-721"*, para criação dos tokens, e do contrato *"Ownable"* para controle de acesso às funções que tratam da manutenção dos tokens, bem como utilizamos ainda a biblioteca de contadores, *"Counters"*, para termos controle dos índices dos tokens criados e a biblioteca *ERC721Burnable*, que tem a função de destruir um token que já não mais será utilizável.

```
1  import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
```

[5] https://hardhat.org/hardhat-runner/docs/getting-started

```
2 import "@openzeppelin/contracts/access/Ownable.sol";
3 import "@openzeppelin/contracts/utils/Counters.sol";
4 import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol";
```

Listagem 4.2: Contratos e bibliotecas importadas.

Posteriormente, o contrato inteligente pode ser criado herdando as funcionalidades do contrato da Linha 2 na Listagem 4.2. Visto que esta importação herda as funcionalidades do padrão ERC-721. Internos ao mesmo, há atributos de identificação dos tokens, com o objetivo de armazenar o índice correto a cada criação de um novo token.

```
1 contract CessionToken is ERC721, IERC, Ownable{
2     uint256 private _nextTokenId;
3     address private _enlace;
4     event TokenId(uint256 tokenId);
5     constructor(address enlace) ERC721("Cessao", "CST"){_enlace = enlace;}
6 }
```

Listagem 4.3: Atributos do contrato CST.

Vale ressaltar que, conforme a Listagem 4.3, é possível observar a inicialização do construtor do padrão ERC-721. Esse construtor recebe como parâmetro o endereço do contrato Enlace, que será responsável por gerenciar as cessões de cada token. O construtor também demanda a inicialização do nome, bem como o símbolo destes tokens. Seguem na Listagem 4.4 as funções do contrato do token que devem ser implementadas conforme as assinaturas apresentadas. Por fim, a função $TokenId(id)$ emite o evento para notificar a criação de um novo token.

```
1     function get_enlace() public view returns (address){ ...}
2     function criarNovoToken() public{ ...}
3     function transferirTokenExpirado(address from, address to, uint256 tokenId) external onlyEnlace { ...}
4     function burn(uint256 Id) external onlyEnlace{ ...}
5     modifier onlyEnlace{ ...}
```

Listagem 4.4: Funções para criar e retornar um Token.

A Listagem de código 4.4 apresenta na linha 1 a assinatura da função $get\_enlace()$ para retornar o endereço do Enlace ao qual o CST está vinculado. A função apresentada na linha 2 tem a finalidade de criação de um novo token. Ela é responsável por realizar as chamadas de funções do padrão ERC-721, como por exemplo, as funções de cunhar um novo token $criarNovoToken()$. Nessa função, primeiramente deverá ser gerado um novo ID de token que é o código de identificação do token, onde é incrementado o contador de ID de token e posteriormente chamar as respectivas funções para criar o token herdado do padrão ERC-721. A função $transferirTokenExpirado()$, descrita na linha 3, apresenta a assinatura da função que realiza a transferência de um token cedido ao cessionário de volta para o enlace. Essa transferência ocorre a partir do momento em que a data de expiração atinge o limite da data atual. A função $burn()$ na linha 4 define o formato da assinatura para queimar um token. Essa função pode ser acionada somente pelo usuário autorizado, uma vez que contém o modificador $onlyEnlace()$ que está definido na linha 5.

Após a apresentação das definições e implementação do contrato CST, será apresentado a seguir o contrato Enlace. O Enlace é o contrato responsável por receber os CST e gerenciar as cessões entre os cedentes e cessionários. Cabe a esse contrato garantir que os tokens cedidos temporariamente voltem nesse prazo aos respectivos proprietários.

Primeiramente, o Enlace é definido como um contrato na linha 1 da Listagem 4.5, que deve conter uma estrutura para estabelecer um item referente a cada CST disponibilizado para cessão. A linha 2 dessa listagem apresenta uma estrutura que mapeia um item cedível chamado de *struct Item*{}. Nela devem constar os campos de dados como o identificador do cedente e do cessionário do token por meio de seus endereços, status indicando se o item está ou não cedido, endereço do contrato ao qual o CST pertence, identificador do CST e o tempo de duração da cessão. Em seguida, a linha 3 define uma lista dos itens cedíveis do enlace, que contém um mapeamento que armazena esses itens criados de acordo com seus identificadores. Após as definições do item e da lista de mapeamento, a linha 4 contém o evento para notificar a inclusão de itens criados para cessão retornando seus respectivos dados. A linha 5 contém a função *criaItemCedivel*() que alimenta a *struct Item*{} com os dados do item. Na linha 6, a função *cederItem*() realiza uma cessão começando pela transferência da propriedade do token para o cessionário e emitindo o respectivo evento. A linha 7 apresenta a assinatura da função $finalizaCessao()$, que retorna a propriedade do token quando as condições forem satisfeitas, e na linha 8 o modificador *onlyEnlace*() que tem a função de definir quem estará autorizado para utilizar uma função que contém esse modificador estabelecido.

```
1  contract Enlace{
2      struct Item{uint256 itemId; ...}
3      mapping(uint256 => Item) private listaItens;
4      event ItemCriado(uint256 indexed itemId, ...);
5      function criaItemCedivel(address contratoCST, uint256 tokenId, uint256 expiraEm) public payable{}
6      function cederItem(address contratoCST, address cessionario, uint256 itemId) public{uint256 tokenId =
           listaItens[itemId].tokenId; ...}
7      function finalizaCessao(uint256 itemId) external {}
8      modifier onlyEnlace() {require(msg.sender == owner, "");_;}
9
10 }
```

Listagem 4.5: Funções do Enlace para gerenciar cessões.

A revogação da cessão, realizada a partir da função $finalizaCessao()$, obtém o endereço do atual proprietário do token e atualiza as informações da cessão no item referente ao token armazenado no contrato Enlace. Por consequência, essa função aciona a função $transferirTokenExpirado()$ do CST que verifica e retorna o token para o endereço do proprietário do token. Essa função também aborda o caso de um token em que seu estado seja “cedido", porém com o prazo de cessão expirado. Em um cenário como este, a função do contrato é chamada com a finalidade de retornar a titularidade do token para o dono de origem e, consequentemente, finalizar a cessão deste token.

4.2.3 Modelagem de testes

Os testes em contratos inteligentes auxiliam na verificação da conformidade dos resultados de uma requisição. As verificações por testes confirmam se o retorno da execução de uma determinada função está ou não de acordo com o que foi pré-definido, garantindo assim que não houve modificação na implantação do contrato e se os dados retornados ou armazenados estão íntegros. Os testes funcionais baseiam-se em uma chamada de funções que realiza requisições com resultados conhecidos. Essas requisições representam as ações de um usuário final da aplicação, juntamente com uma API que retornará os dados requisitados a partir de uma solicitação direta ao contrato inteligente.

Conforme já especificado na Seção 4.2.2, os contratos CST e Enlace exercem em conjunto a função de ceder um token a um usuário de forma temporária. Neste sentido, os testes funcionais desses contratos seguem o modelo da Figura 4.7, onde são observados os componentes necessários para a execução e verificação das funcionalidades presentes nos contratos.

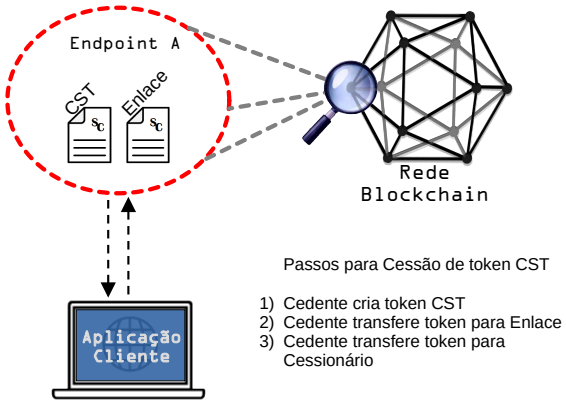


Figura 4.7: Modelagem dos testes no contrato do token CST. Fonte: elaborado pelo autor

O modelo apresenta o componente "Aplicação Cliente"que executa os testes com a função de requisitar, a uma API, os métodos dos contratos e assim alcançar os resultados de suas requisições. No caso do contrato CST, os métodos principais são criar e transferir os tokens, sendo a API quem irá atender às requisições, recebendo ou retornando os dados das solicitações. A definição desta API segue os padrões dos contratos fornecendo à Aplicação Cliente os dados processados pelos métodos, na devida ordem, para que os testes possam atuar verificando a exatidão das execuções. Os contratos CST e Enlace são os códigos gerados e implantados na rede Blockchain por meio de um nó *Endpoint*. Os códigos dos contratos são escritos na linguagem de programação específica para a plataforma, no caso do Ethereum, a linguagem Solidity, e formatados em um padrão JSON, contendo também metadados com informações necessárias à comunicação com as bibliotecas da API.

A fase dos testes contempla a criação de roteiros definidos com valores esperados dos

retornos e que serão confrontados com os valores retornados pela API ao serem requisitados dos contratos. As validações dos testes ocorrem após o confronto dos valores esperados e os retornados, resultando em "Verdadeiros"ou "Falsos". Este roteiro pode ser demonstrado em um exemplo onde uma função do contrato é chamada para realizar a verificação de saldo em uma conta, uma vez que, para transferência de valores, o saldo tem que ser maior ou igual ao valor desejado para transferência.

- Roteiro de teste 1 - CST

    1. Transferir $\boldsymbol{X}$ valor da $\boldsymbol{Conta_A}$para$\boldsymbol{Conta_B}$;
        (a) Obter $\boldsymbol{saldo_A}$
        (b) Se $\boldsymbol{saldo_A >= X}$; transferir o valor $\boldsymbol{X}$ para a $\boldsymbol{Conta_B}$;
        Senão => Abortar operação
        – Retorna $\boldsymbol{Verdadeiro}$ caso a transferência seja realizada;
        ou $\boldsymbol{Falso}$ caso não realize a transferência.

No caso em que, chamada a função para transferir $\boldsymbol{X}$ valores da $\boldsymbol{Conta_A}$ para $\boldsymbol{Conta_B}$, sendo que o valor de $\boldsymbol{X}$ a ser transferido é igual a 50, e que o $\boldsymbol{saldo_A}$ é igual a 100, o retorno do teste será $\boldsymbol{Verdadeiro}$.

Os roteiros e a codificação dos testes realizados para os contratos CST e Enlace, juntamente com os resultados das validações, estão descritos e apresentados no Apêndice C. Conforme os resultados apresentados, todas as funções passaram pelos testes e obtiveram aprovação integral em suas validações.

4.3 Token de Cessão Multicadeia

Esta seção apresenta a extensão dos requisitos do CST para prover a interoperabilidade do token entre plataformas blockchain homogêneas. A extensão está sendo nominada aqui como Token de Cessão Multicadeia $(\boldsymbol{CST_{mc}})$ e permite a validação de posse do CST em outras redes blockchain, preservando as características de observação da duração da cessão e revogação da posse do token. Além da especificação da extensão, essa seção abrange também o projeto do arcabouço de interoperabilidade, contando com a implementação, implantação e testes do CST, juntamente com o mecanismo de enlace para interoperar o CST.

4.3.1 O Problema

A motivação da utilização do $\boldsymbol{CST_{mc}}$ vem do paradigma atual de transferência de tokens. Uma vez que há perdas de lastro entre os tokens, mediante a forma em que os tokens são transferidos, passando pela criação do token em uma nova rede blockchain destinatária

e bloqueando-o ou excluindo-o na rede remetente. Esse padrão seguido faz com que o histórico das transações seja fragmentado ou até mesmo tornado inacessível, no caso de exclusão do token. Um outro padrão já estabelecido é quanto ao formato da transferência, entre diferentes blockchains, que não permitem a possibilidade de coexistência do token nas redes. Desta forma, permitir a coexistência e sincronizar o estado de tokens em mais redes blockchains, para que não haja perda de informação da propriedade atual do token e a fragmentação das informações de seu histórico, é um alcance considerável para este requisito.

Os contratos inteligentes desenvolvidos e que integram a solução para o $\boldsymbol{CST_{mc}}$ seguem o mesmo padrão e a lógica necessária aos requisitos dos contratos desenvolvidos para o CST. Porém, algumas mudanças se fizeram necessárias para que a comunicação entre diferentes plataformas Blockchain transcorresse conforme a necessidade da operação. Portanto, estão apresentados nesta seção as implementações e modificações dos contratos, os testes e invocações que também foram programados a partir do ambiente de desenvolvimento Hardhat . O processo de desenvolvimento do $\boldsymbol{CST_{mc}}$ utiliza a mesma modelagem do CST, demonstrada na Seção 4.2.2, que foi planejado e implementado de acordo com os requisitos apresentados.

A técnica conhecida como cross-chain, apresentada na seção 2.5, permite que as blockchains sejam interconectadas e que os ativos possam ser transferidos de forma segura entre elas, ampliando a interoperabilidade e a aplicabilidade de diferentes soluções em blockchains distintas. Porém, não estão consideradas nessas técnicas a possibilidade de conciliação do estado de ativos entre as blockchains, nem a atualização do estado mediante a validade de tempo. Portanto, o projeto $\boldsymbol{CST_{mc}}$ visa padronizar uma interface para verificação cross-chain de propriedade de token na camada de contrato, por meio da qual pode-se definir um proprietário do token global herdado do ERC-721 sobre múltiplas Blockchains.

4.3.2 Extensão do CST aplicada à interoperabilidade

O CST_{mc} utiliza contratos inteligentes, preexistentes nas redes que fazem parte da operação cross-chain, que estabelecem um protocolo para registrar os estados de propriedade dos tokens de forma síncrona. Esses contratos são nomeados como protocolo de Enlace, fazendo com que as concessões de tokens sejam realizadas com o padrão do CST, permitindo a revogação e finalização da cessão por tempo.

O diagrama apresentado pela Figura 4.8 demonstra os componentes pertencentes ao projeto CST_{mc} para realizar a sincronização dos tokens entre duas blockchains. A sincronização do estado global dos tokens é realizada por meio do Protocolo de Enlace. O protocolo é definido externamente aos contratos, porém com as garantias de segurança e integridade do que foi estabelecido pelos contratos do token e do enlace.

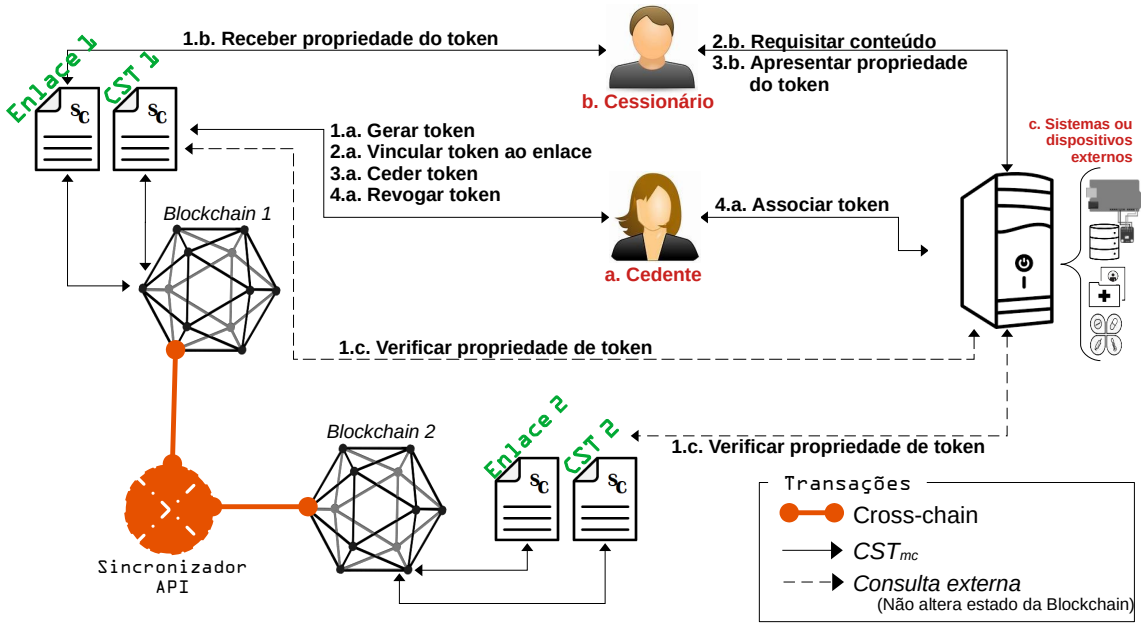


Figura 4.8: Diagrama representando as transações para o CST_{mc}. Fonte: elaborado pelo autor

Na sequência estão dispostas as descrições e listagem dos códigos contendo os detalhes sobre as modificações e implementação do contrato $Enlace_{mc}$. Inicialmente, é importante ressaltar que o contrato escrito para o CST 4.1 não sofreu modificações e segue os mesmos moldes de um contrato padrão ERC-721. Portanto, cabe apenas implantar o contrato CST em cada uma das redes pertencentes ao esquema cross-chain, bem como os contratos de $Enlace_{mc}$. Esse contrato, por sua vez, sofreu alterações consideráveis em relação ao contrato $Enlace$ por ter que incrementar a comunicação entre redes blockchain.

```
1  contract Enlacemc is ERC721 {
2      struct CSTTransfer {}
3      struct MintRecord {}
4      event CSTTransferInterChain();
5      event CSTMinted();
6      constructor() ERC721("ENLACE", "ELC") {}
7      function transferCSTInterChain() external {    }
8      function mintCST() public payable onlyEnlace {}
9  }
```

Listagem 4.6: Funções do contrato $Enlace_{mc}$ para interoperar um Token.

O código apresentado na Listagem 4.6 contém na linha 2 uma estrutura definida para armazenar informações da transferência e na linha 4 um evento para registrar a transferência interchain. A linha 3 contém uma estrutura de dados com identificadores da operação, endereços do cedente e do cessionário e identificadores do token. E na linha 4, o evento responsável por registrar os dados da função $MintRecord()$. Na linha 6, o construtor define o nome e o símbolo do token e deve definir o proprietário do contrato. A linha 6 contém a função $transferCSTInterChain()$, a qual é atribuída a tarefa de receber e transferir um CST, mantendo as informações e enviando para outra rede. Já a função mintCST(), na linha 8, pode ser invocada apenas pelo usuário proprietário do

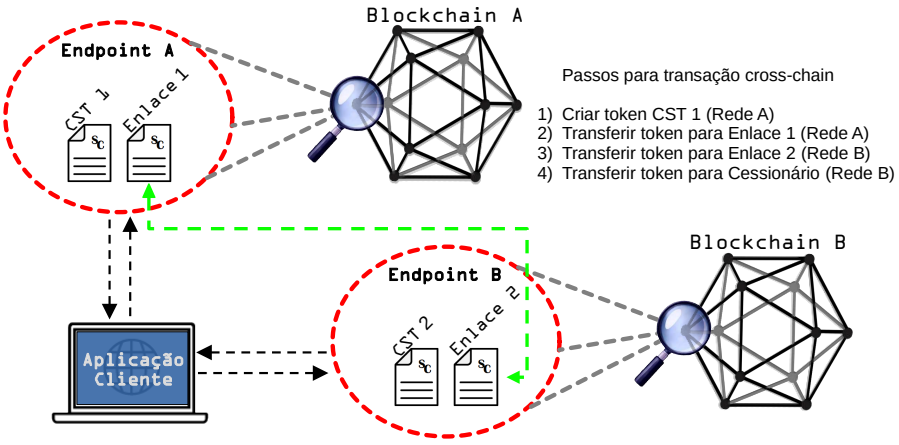


Figura 4.9: Modelagem dos testes de interoperabilidade do CST_{mc}. Fonte: elaborado pelo autor

contrato $Enlace_{mc}$ para criar uma réplica do token a ser enviada para a outra rede.

4.3.3 Modelagem de testes

Os testes para os contratos inteligentes utilizados na solução do CST_{mc} foram modelados com o intuito de realizar verificações que confirmem se o retorno da execução das funções está ou não de acordo com o que foi programado. Esta verificação aumenta a garantia de que não houve modificações na implantação do contrato e a integridade dos valores retornados ou armazenados. Os testes realizaram chamadas de funções dos contratos e seus retornos foram comparados com resultados previamente conhecidos. As chamadas de funções representam as possíveis ações de um usuário final da aplicação.

Os contratos CST_{mc} e $Enlace_{mc}$, conforme especificados nas Seções 4.2 e 4.3, exercem em conjunto as funções de ceder um token a um usuário de forma temporária e abrangendo outras cadeias além da originária do token. Desta forma, os testes modelados para estes contratos e apresentados pela Figura 4.9 demonstram os componentes necessários para a execução e verificação das funcionalidades presentes nos contratos.

Da mesma forma que a modelagem de teste para o CST, o modelo para o CST_{mc} também apresenta o componente "Aplicação Cliente"que requisita as funções dos contratos com o objetivo de obter os retornos e compará-los. No caso dos contratos para o CST_{mc}, os métodos principais também são criar e transferir os tokens. Porém, neste caso, a aplicação de teste deve interagir com as duas redes, seguindo os padrões dos contratos, na devida ordem, para que os testes possam atuar verificando a exatidão das execuções. Os dois contratos,CST_{mc} e $Enlace_{mc}$, devem ser implantados nas duas redes Blockchain por meio de um nó *Endpoint* de cada uma das redes. No caso da figura que representa a modelagem, as blockchains se diferem por A e B e os *endpoints* de cada rede também por A e B. Os contratos pertencentes à rede A receberam os nomes de CST 1 e Enlace 1 e os

contratos da rede B receberam os nomes de CST 2 e Enlace 2.

Os roteiros e a codificação dos testes realizados para os contratos CST_{mc} e $Enlace_{mc}$, juntamente com os resultados das validações, estão descritos e apresentados no Apêndice D. Conforme os resultados apresentados, todas as funções passaram pelos testes e tiveram resultados positivos em suas validações.

4.4 Considerações Finais

O uso da tecnologia blockchain requer escolhas assertivas em relação à plataforma e à infraestrutura a ser utilizada. O modelo de rede impacta nas qualidades não funcionais das aplicações, em especial desempenho e custo. Neste capítulo, investigou-se o impacto, comparando o custo de diferentes infraestruturas para a solução de aplicação CST. Primeiramente, foi modelado o custo monetário da infraestrutura para a aplicação obter a vazão máxima em função da carga esperada em transações por segundo. A seguir, ilustra-se como o modelo de custo pode ser usado para analisar o custo de infraestrutura para redes blockchains públicas, considerando diferentes cargas de trabalho. Por fim, foram discutidos os limites de escalabilidade dessa rede e o seu compromisso entre custo e desempenho. Foi apresentada uma solução para o problema de disponibilização de tokens temporários, o CST. Em primeiro lugar, uma arquitetura típica para aplicações descentralizadas foi definida e seus principais componentes foram apresentados. A formulação dos contratos foi introduzida juntamente com os padrões e bibliotecas utilizadas. O processo de desenvolvimento e codificação do CST foi apresentado e uma solução para interoperar esse mesmo token, o CST_{mc}, foi proposta. Os testes das funcionalidades desses contratos e a extensão aplicada para interoperabilidade foram modelados de acordo com os componentes necessários para a execução e verificação. Desta forma, todas as funções testadas obtiveram resultados integralmente positivos em suas validações.

Capítulo 5

Experimentos e Resultados Alcançados

Este capítulo apresenta uma descrição dos procedimentos adotados nos experimentos realizados e dos resultados obtidos. O objetivo principal é experimentar, demonstrar, comparar e avaliar a demanda de recursos computacionais de um nó *endpoint*. Esse nó *endpoint* é caracterizado neste trabalho como uma infraestrutura particular e adequada para participar ativamente de uma blockchain. A forma de variação da demanda por recursos ao nó nesses experimentos se dá por meio da variação de cargas de trabalho e da complexidade de processamento nos algoritmos programados nos contratos inteligentes utilizados. Dessa forma, os experimentos foram configurados com a finalidade de avaliar os custos provenientes da infraestrutura computacional necessária de um nó *Endpoint* para prover acesso à blockchain, apresentado no Capítulo 4, e apresentar a aplicabilidade das funcionalidades do Token de Cessão, apresentado na Seção 4.2, avaliando e comparando os seus custos. Ainda neste capítulo estão descritos os experimentos para analisar o desempenho e custo de protocolos utilizados para interoperar o CST_{mc}. Na seção 5.2 estão descritas as configurações dos experimentos, onde são introduzidos o ambiente experimental e as redes de testes de blockchains públicas utilizadas. Por fim, são detalhados o ambiente experimental elaborado nessas redes e as métricas utilizadas para obter os resultados e produzir dados práticos para a análise.

5.1 Experimentos utilizando o Token de Cessão

5.1.1 Descrição e Configuração

Os experimentos foram conduzidos de maneira a obter resultados que favoreçam a análise sobre os custos de transações com os contratos do CST e os custos de operação do *Endpoint* ao suprir uma aplicação com infraestrutura computacional utilizada por uma blockchain pública. As análises são provenientes dos requisitos dos cenários de submissão de transações e de variação da complexidade computacional em diferentes

recursos computacionais. A EVM da Ethereum foi a plataforma Blockchain pública escolhida como base para produzir dados de todas as métricas de custo e desempenho avaliadas. A utilização de uma plataforma pública nesse trabalho se deve ao fato de que a transparência, a imutabilidade e a distribuição são requisitos essenciais para a empregabilidade da verificação de propriedade de tokens, uma vez que o usuário requer confiança e verificabilidade nas transações. Sendo assim, as redes públicas são mais acessíveis, confiáveis e propícias a serem verificadas.

Dois cenários conduziram a modelagem de experimentação da infraestrutura utilizada para os experimentos. Esses cenários são caracterizados por diferentes formas de demanda da infraestrutura e identificados na Tabela 5.1 como Cenário 1 e 2. O primeiro cenário utiliza o CST, apresentado na Seção4.2 do Capítulo 4, como proposta de submissão de transações para demandar a infraestrutura computacional da rede. Portanto, o projeto do CST foi implementado e utilizado como um dos contratos-teste para submissão de transações e avaliação do referido nó. Além do cenário dos tokens, também foi definido o segundo cenário em que são avaliados diferentes tamanhos de entrada em uma função implementada em um contrato teste com complexidade de tempo $O(n)$, sendo que o tempo de execução depende do valor de (n). Para cada um desses dois cenários, foram analisados, tratados e avaliados o desempenho e o custo computacional do nó *endpoint*. Portanto, o problema relatado e tratado neste capítulo inclui a observação sobre as formas e custos de utilização da infraestrutura computacional pela tecnologia Blockchain, bem como as implicações provenientes dos requisitos do cenário de submissão de transações e complexidades computacionais.

	Abordagens
Cenário 1	Experimentos utilizando os contratos da solução para o CST
Cenário 2	Experimentos utilizando contrato implementado para algoritmo de complexidade $O(n)$

Tabela 5.1: Abordagens de cenários para avaliação e experimentos

Três experimentos foram definidos a partir da descrição desses cenários. Portanto, os experimentos têm o intuito de prover dados para gerar informações sobre a vazão e latência da rede e a demanda e o custo da infraestrutura. Os experimentos estão nomeados como Experimento 1, 2 e 3 e delimitados por meio das seguintes explicações:

- **Experimento 1** - O primeiro experimento utiliza os contratos inteligentes implementados com os algoritmos do CST para submeter um certo número de transações por um determinado tempo e em diferentes tipos de recursos computacionais. Foram observadas as medidas de vazão, medida em transações por segundo (tps), e latência, medida em segundos (s), considerando as duas principais transações relacionadas aos tokens, criar e transferir. Por esse

experimento também foi observado quais recursos computacionais foram mais demandados e que assim refletiram no desempenho da rede. Portanto, os fatores utilizados nesse primeiro experimento foram a carga de transações e o tipo de recurso computacional. Nesse Experimento 1 também observaram-se as influências dos recursos computacionais sobre a vazão e latência.

- **Experimento 2** - O segundo experimento utiliza um contrato inteligente implementado com um algoritmo de complexidade $O(n)$. Também foram variados o tipo de recursos computacionais por meio de diferentes *endpoints* e o valor da entrada n. Contudo, a taxa de transações foi mantida em 300 tps. A manutenção da taxa em 300 tps se deve ao fato de que nessa taxa todos os recursos foram capazes de processar as transações sem perdas. Foram observadas as medidas de vazão (tps) e latência (s) considerando as diferentes entradas n e os diferentes recursos computacionais. Portanto, os fatores utilizados nesse segundo experimento foram a entrada do algoritmo n e o tipo de recurso computacional.

- **Experimento 3** - O terceiro experimento apresenta uma proposta de avaliação, também utilizando o contrato inteligente implementado com um algoritmo de complexidade $O(n)$, porém expandindo o valor de entrada do algoritmo (n), mantendo o recurso computacional e também a taxa de transações em 300tps. Portanto, o fator utilizado nesse Experimento 3 foi apenas a entrada do algoritmo n. Foram ainda observadas as medidas de vazão (tps) e latência (s) considerando as diferentes entradas n.

5.1.2 Ambiente Experimental e Métricas

A partir da definição dos cenários e experimentos, fez-se necessário produzir ambientes experimentais utilizando diferentes tipos de recurso computacional para medir o desempenho e o custo de um nó *endpoint*. Esse *endpoint* foi configurado seguindo a arquitetura proposta na Seção 4.1, a $ARCHchain$, e o CST apresentado na Seção 4.2 do Capítulo 4. A Figura 5.1 demonstra como foi construído esse ambiente e em seguida são descritos os detalhes.

Outro ponto importante da configuração dos nós é em relação à plataforma Blockchain em que foi criado o nó *endpoint*. Os nós foram construídos a partir da implementação oficial do protocolo Ethereum, o *Go-ethereum (Geth)*, na versão 1.14.3. As máquinas virtuais (VMs) foram configuradas como clientes de execução *Geth* para receber as propostas de transações, retransmitir solicitações de transações e manter o livro-razão, e como clientes de consenso, i.e., os componentes consenso e P2P definidos na arquitetura geral. Os dois clientes foram iniciados em uma mesma máquina virtual, ambos configurados com interfaces de rede para comunicarem entre si via as portas TCP

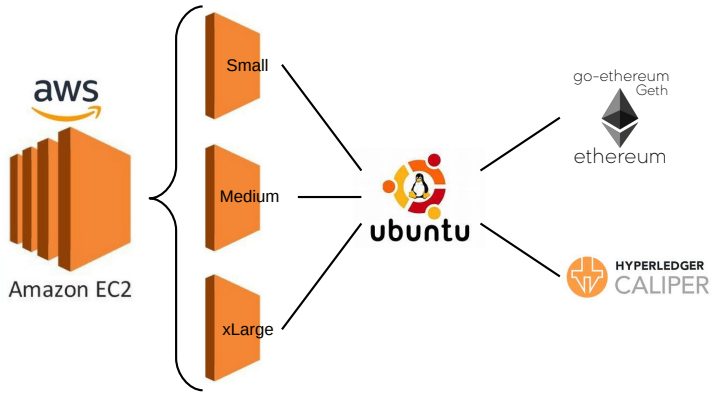


Figura 5.1: Configuração de VMs do Ambiente Experimental. Fonte: elaborado pelo autor

e UDP do protocolo Ethereum. O nó recebe requisições de transações da aplicação através da camada de execução e as processa e repassa ao cliente de consenso, que é responsável pela verificação das transações e geração dos blocos que encadeiam e armazenam essas transações. Contudo, o componente consenso não está conectado à rede principal pública Ethereum e os blocos gerados contêm apenas transações da aplicação de nosso ambiente experimental. Para simular uma vazão semelhante à rede principal Ethereum, o tempo de criação de bloco para o protocolo de consenso foi estipulado em 12 segundos.

Foi utilizada para coletar as métricas de vazão e latência a ferramenta de aferição ou *benchmark*, *Hyperledger Caliper*, na versão 0.6.0 (Caliper, 2019). O *Hyperledger Caliper* também foi configurado para atuar como aplicação cliente gerando cargas com emissão de transações para os dois cenários estipulados. As cargas sintéticas geradas emulam uma aplicação blockchain típica com foco na emissão de transações, i.e., a inserção de registros de criação e transferência de tokens no primeiro cenário, e no segundo cenário a inserção de registros após o processamento do algoritmo com complexidade $O(n)$ que é usualmente uma operação com maior uso de recursos computacionais e latências em blockchains como já observado em trabalhos anteriores (Spengler and Souza, 2021). Assim, a carga de trabalho submetida representa um conjunto de registros emitidos por segundo (transações por segundo - tps), de forma fixa em um dado período de tempo. O valor em tps das cargas de trabalho foi incrementado gradativamente até atingir o limite de cada VM, i.e., uma carga de alta intensidade. Para cada carga foram realizadas 10 rodadas de 60 segundos cada.

Três tipos de recursos computacionais foram utilizados para executar os experimentos nos ambientes que representam cada cenário na rede blockchain pública Ethereum. Esses tipos são máquinas virtuais (VMs) do serviço *Amazon Elastic Cloud Computing* (EC2) para compor os nós de cada rede. O poder computacional desses nós foi incrementado gradualmente nos experimentos, com o intuito de analisar o desempenho dos recursos em

função do aumento de carga e da variação de cenário. Nesse sentido, foram utilizadas as VMs T2 do tipo *small, medium* e *xlarge*, cujas respectivas especificações de processamento, memória e custo são apresentadas na Tabela 5.2.

	Small	**Medium**	**xLarge**
vCPUs	1	2	4
Memória (GB)	2	4	16
Custo/hora (USD)	0,0230	0,0464	0,1856

Tabela 5.2: Especificações dos nós que compõem cada tipo de infraestrutura: família AWS T2, processador Intel Xeon 3.0-3.3 GHz e disco SSD de 100 GB.

Foram coletadas as métricas de vazão e latência para cada carga de trabalho executada. Além dessas métricas, coletou-se também o uso dos recursos de processamento (CPU), memória, leitura e escrita em disco e entrada e saída de rede para os nós da rede. Por meio do *Hyperledger Caliper* foram registrados a vazão, a latência e a confirmação (sucesso ou falha) para cada transação. Foi implementado um algoritmo para a coleta dos dados sobre os recursos computacionais dos nós monitorados nos ambientes experimentais. Esse software foi desenvolvido utilizando a linguagem Python juntamente com a biblioteca *Psutil 5.9.0*. O código encontra-se integralmente no Apêndice B.

Intuitivamente, o crescimento da latência de transações pode estar associado ao aumento do consumo de recursos computacionais. Nesse sentido, foi examinado o consumo de recursos em nós de diferentes tipos e configurações com a finalidade de observar quais recursos são mais requisitados, preliminarmente ao início dos experimentos. Foi observado que CPU é o recurso que tem o uso mais impactado com os aumentos de carga (tps). O uso de memória permanece estável em torno de 1 GB, ao passo que o uso da rede (entrada e saída) e do disco cresce com o aumento de carga, mas fica distante das capacidades máximas que são iguais para diferentes tipos de nós, i.e., 1 Gbps de comunicação entre os componentes (containers) e 1 Gbps de leitura/escrita em discos SSD. Portanto, o foco das análises a seguir é no uso de CPU.

Inicialmente, para todos os experimentos, os contratos inteligentes foram implantados por meio do nó *Endpoint* da Blockchain e a aplicação configurada para realizar a geração e inserção de transações invocando as funções dos contratos. O código da aplicação desenvolvida para a execução dos testes ficou responsável por executar, em ordem, chamadas das funções previstas nos contratos de forma a observar sua correta execução conforme pré-estabelecidas pelos Cenários 1 e 2.

Para cada carga de trabalho executada, foram medidos a vazão da rede em tps e o atraso da transação, além da medição do uso dos recursos de processamento (CPU), memória, disco e rede para os nós da rede. O *Hyperledger Caliper* registra o instante de envio e de confirmação (sucesso ou falha) para cada transação. Assim, a vazão é calculada pela taxa total de transações com sucesso sobre o período total da carga aplicada, i.e., a diferença entre o instante da última confirmação e o instante da primeira submissão.

Por sua vez, o uso dos recursos computacionais foram coletados em granularidade de segundos via a biblioteca *Psutil) versão 5.9.0*. Esta biblioteca é multiplataforma e tem como finalidade o monitoramento de processos e sistemas em Python. Desenvolvemos um script utilizando a biblioteca *Psutil* e instalamos em cada nó da rede para coletar os dados referentes aos recursos monitorados.

5.1.3 Resultados

As duas seções anteriores apresentaram as configurações utilizadas nos experimentos para avaliar um nó participante de uma rede blockchain pública e as configurações do ambiente para obter valores das métricas necessárias para fomentar o modelo de custo apresentado no Capítulo 4. Esta seção apresenta os resultados e avalia experimentalmente o custo e desempenho desse nó de forma conjunta, buscando analisar o compromisso entre esses dois importantes fatores para desenvolvedores e organizações que necessitam dessa informação para planejarem a participação em redes blockchain. A avaliação busca responder as seguintes perguntas:

1. *A arquitetura proposta no Capítulo 4, que unifica os componentes em um único nó, consome rapidamente recursos computacionais do nó levando à redução de desempenho?*

2. *A arquitetura requer um nó de alto custo para executar transações com níveis razoáveis de desempenho.*

5.1.3.1 Avaliação de Desempenho do *Endpoint*

Esta subseção busca responder à primeira pergunta de pesquisa onde é questionado se: *"A arquitetura proposta consome rapidamente recursos computacionais do nó levando à redução de desempenho?"*. Para responder a essa pergunta, foi relacionada a utilização de processamento das VMs com a latência média de transações, considerando o crescimento da carga de trabalho, a complexidade das funções executadas e o aumento do poder computacional das VMs.

Primeiramente, estão apresentados no gráfico 5.2 os desempenhos alcançados individualmente pelas diferentes infraestruturas avaliadas. Nesse caso, as cargas de trabalho foram inteiramente baseadas em uma função de transferência da aplicação construída para o *CST*, rodando sobre a plataforma de rede Blockchain pública construída e especificada na Subseção 5.1.1. O gráfico da Figura 5.2 mostra a variação da vazão em função da carga de trabalho enviada e medida em transações por segundo (tps). As medições são observadas nos três tipos de VMs em que foram instanciados para representar os nós da rede. Ainda é possível observar que, para os três diferentes tipos de VMs, o nó *endpoint* comportou-se igualmente em relação à vazão alcançada

versus a taxa de envio. Portanto, para os cinco valores de taxas enviados, isto é, da faixa de 100 a 300 tps variando em 50 tps a cada rodada, obteve-se uma vazão igualmente ao valor de envio da carga de trabalho. Demonstrando que, para a carga de até 300 tps, todas as VMs realizaram o processamento de forma ótima e sem falhas.

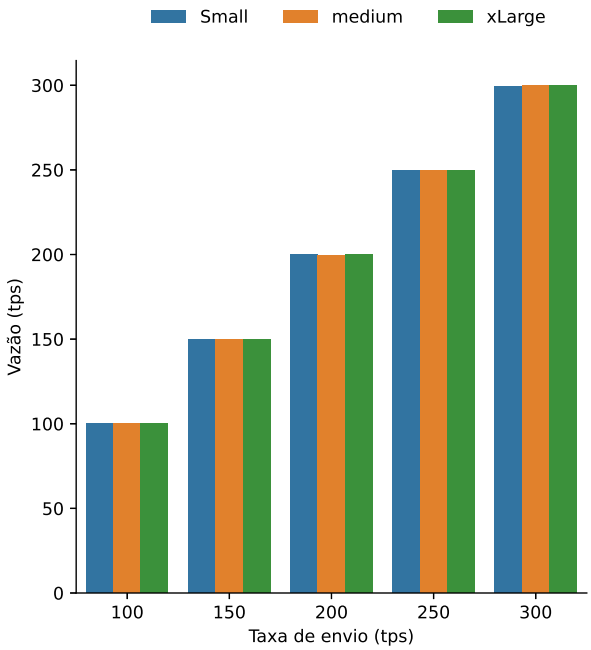


Figura 5.2: Carga de trabalho x Vazão - Execução dos métodos *CST* com variação de VMs

Em um segundo momento, tratou-se de verificar o desempenho, porém nesse caso, as cargas de trabalho foram inseridas com variação do valor de entrada de uma função com complexidade $O(n)$. Os valores dos resultados referem-se também aos testes alcançados individualmente pelas diferentes infraestruturas avaliadas. Sendo que a carga de trabalho enviada foi mantida em 300 tps, em decorrência da capacidade ótima máxima alcançada na vazão do gráfico da Figura 5.2. Os valores de entrada (n) variaram em uma faixa de 300 a 1500, para cada uma das VMs, e podem ser observados no gráfico da Figura 5.3.

Para a VM do tipo *Small*, foi possível processar toda a carga de trabalho enviada para o menor valor de (n), isto é, $n = 300$. Aos valores de (n) iguais a 400 e 500, a VM(Small) obteve um atraso na vazão, porém, ainda assim conseguiu processar todas as transações sem falhas. Para valores de (n), maiores e iguais a 750, a VM(Small) não conseguiu processar todas as transações, obtendo na sua quase totalidade falhas e travamento do nó da rede. A VM do tipo *Medium* processou a carga de trabalho enviada, sem falhas e atrasos até $(n = 500)$. Quanto ao valor de $(n = 750)$, essa VM processou as transações recebidas sem erro, porém com atrasos. Para valores de (n), maiores e iguais a 1000, a VM (Medium) não conseguiu processar todas as transações, obtendo na sua quase totalidade falhas e travamento do nó da rede. A Figura 5.3 apresenta ainda os resultados observados para a VM do tipo *xLarge*. Nessa VM, para todos os valores de (n), isto é,

na faixa de 300 a 1500, as transações recebidas foram processadas sem erro, porém para os valores maiores e iguais a 1000 houve atrasos. De modo geral, nota-se que a variação do aumento da complexidade de uma função em um Contrato Inteligente acarreta uma mudança significativa no comportamento das VMs.

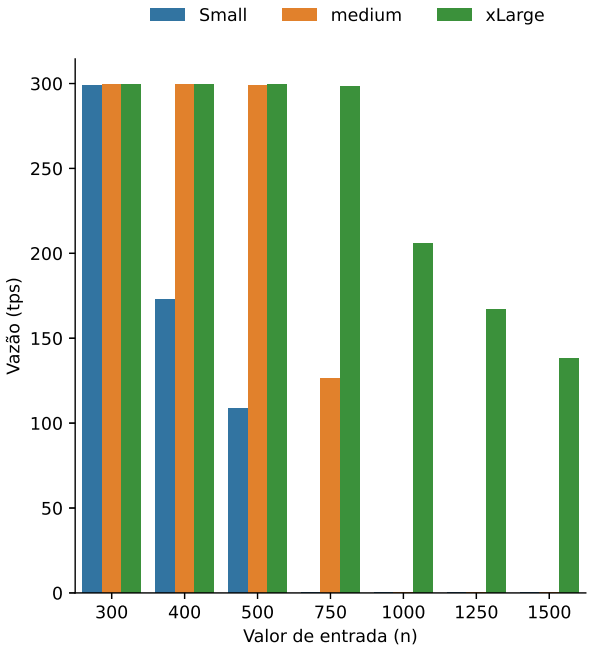


Figura 5.3: Gráfico n vazão computação com 300 TPS

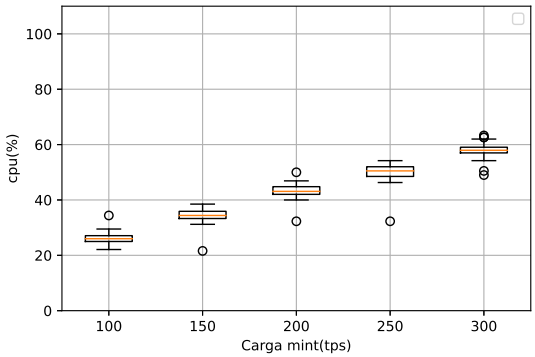
Experimento 1 - Transações com Token de cessão

Este primeiro experimento propõe-se a avaliar o desempenho necessário de uma infraestrutura alocada para provimento e utilização de aplicações em uma plataforma Blockchain pública compatível com EVM Ethereum. Buscou-se apresentar uma análise de utilização da aplicação, proposta como solução para a validação e rastreabilidade de propriedade de token, visando identificar a infraestrutura que leva ao melhor desempenho, considerando ao mesmo tempo o fator custo. Portanto, esta subseção apresenta uma avaliação que representa as características que interferem nos custos da infraestrutura por transação confirmada na Blockchain, considerando a relação entre o custo monetário do recurso computacional necessário para executar a aplicação Blockchain e o desempenho, medido pela vazão máxima obtida por esse recurso em transações.

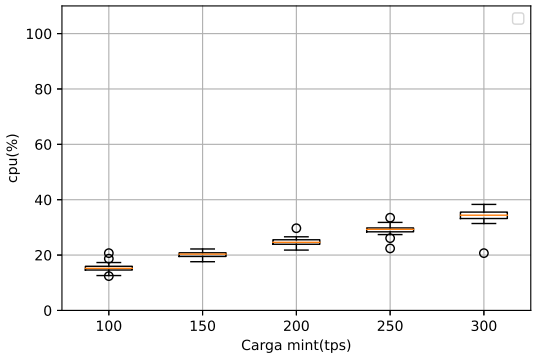
Portanto, como instrumento de pesquisa, criou-se um instrumento para conceder e revogar o consentimento de direitos de uso e posse de um token em específico, por terceiros e por um determinado tempo. Com o cumprimento desta meta, pode-se responder às seguintes perguntas: Como ceder e revogar permissões em blockchain de forma temporária? Redes Blockchain possibilitam um meio com escalabilidade e

disponibilidade para armazenar e controlar meios de verificação de propriedade e acesso por tokens?

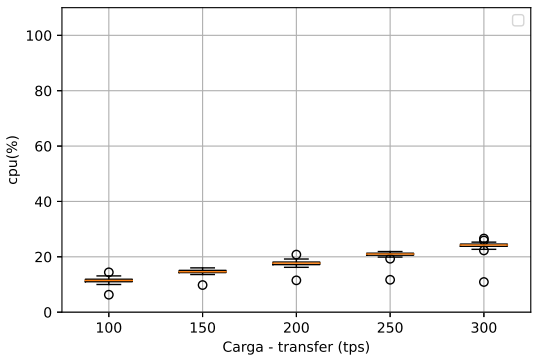
No gráfico da Figura 5.4 são apresentados os resultados do Experimento 1 em relação à demanda por processamento dos três tipos de VMs experimentados. Nesse caso, já é possível observar que a demanda pelo recurso computacional de processamento é aumentada à medida que o número de transações por segundo tem um maior número de solicitações.



(a) VM (Small)

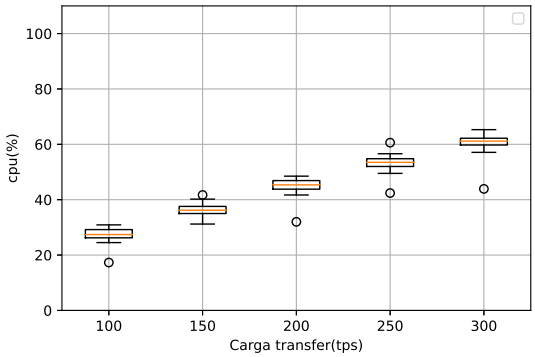


(b) VM (Medium)

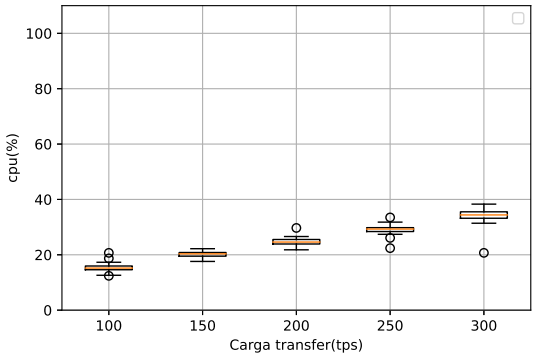


(c) VM (xLarge)

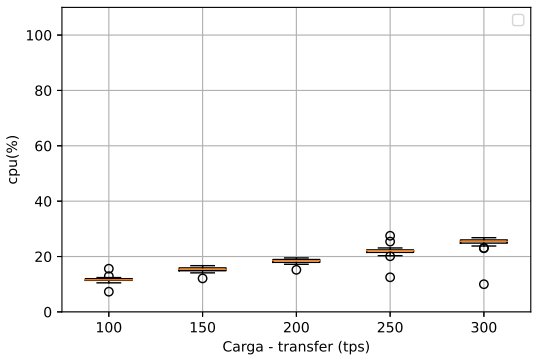
Figura 5.4: Uso de CPU (%) em chamada de função CST(mint).



(a) VM (Small)



(b) VM (Medium)



(c) VM (xLarge)

Figura 5.5: Uso de CPU (%) em chamada de função CST(transfer).

Foram executados experimentos baseados em redes blockchain públicas, padrão EVM - Ethereum, onde foram implantados contratos em nós e houve um crescimento gradativo da infraestrutura computacional, representada por CPU e memória principal, e também sob cargas de trabalho crescentes, conforme a metodologia descrita na seção anterior. Nesse caso, observaram-se desempenhos em níveis satisfatórios resultando em perdas zero ou inferiores a 1 da carga e vazões, apresentando um pouco do comportamento ligeiramente linear à carga de trabalho imposta pela aplicação. Contudo, observou-se por meio da

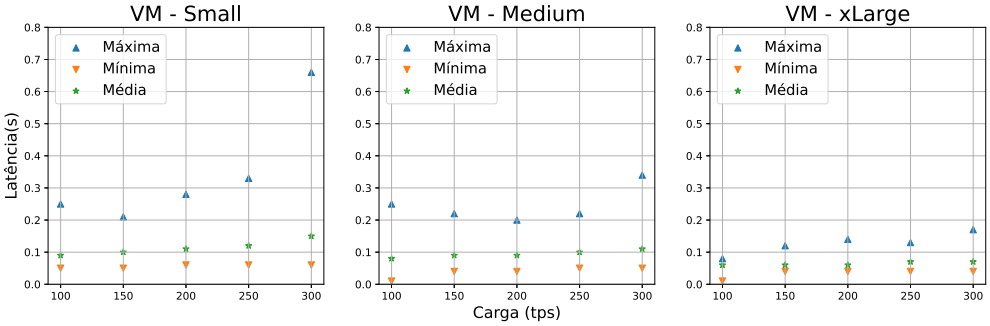


Figura 5.6: Latência em VMs para chamada de função CST(mint)

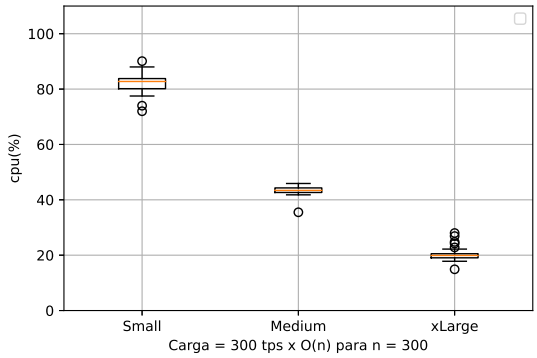
Figura 5.6 que o desempenho das redes alcança um estágio de perda de desempenho (i.e., aumento da latência) à medida que se aumenta a carga de trabalho para cada tipo de infraestrutura. Essas perdas coincidem com a alta utilização de CPU, Figura 5.4 que visivelmente demonstra o aumento de utilização dos recursos da infraestrutura em certos graus de aumento de carga. Para identificar quantitativamente a partir de quais cargas as perdas são desencadeadas, analisou-se a variação do uso de CPU em granularidade de segundos.

Experimento 2 - Transações com complexidade em processamento

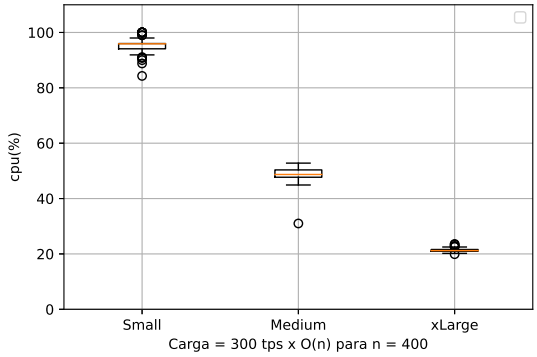
O segundo experimento propõe-se avaliar o desempenho da mesma infraestrutura utilizada pelo Experimento 1, porém utilizando um contrato que implementa um algoritmo de complexidade $O(n)$. Essa nova abordagem implementa uma solução que possibilita aumentar a complexidade da carga de trabalho de forma controlada. Esse experimento visa identificar a infraestrutura que, mesmo com tarefas mais complexas, em nível de processamento, consegue entregar um menor custo, levando ao melhor desempenho. Sendo que dessa forma, consideram-se ao mesmo tempo o fator de custo e a sobrecarga dos recursos computacionais. Portanto esta subseção apresenta uma avaliação que utiliza do aumento gradativo do número de entradas do algoritmo, fazendo com que seja possível identificar as características da infraestrutura que influenciam nos custo por transação confirmada na Blockchain, considerando a relação entre o custo monetário do recurso computacional necessário para executar a aplicação Blockchain e o desempenho, medido pela vazão máxima obtida por esse recurso em transações. Por fim, os resultados desse experimento 2, são capazes de demonstrar, por meio da expansão da utilização das blockchains para outros fins, que não os financeiros, a escalabilidade e disponibilidade de processamento de algoritmos mais complexos.

Nos gráficos da Figura 5.7 são apresentados os resultados do Experimento 2 em relação à demanda por processamento dos três tipos de VMs experimentados. Nesse caso, é possível observar que o recurso computacional de processamento é o de maior relevância. Dessa forma, o aumento de valor da entrada nesse algoritmo ocasiona perdas de transações

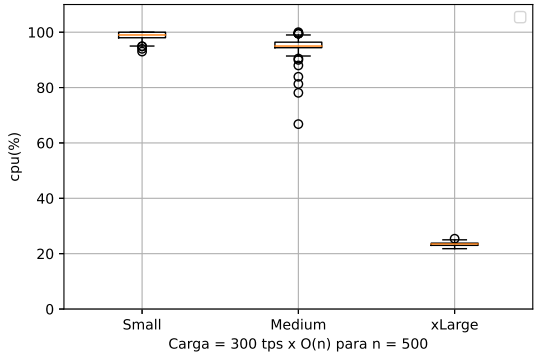
e até mesmo a implicação de travamentos devido à alta demanda e negação de resposta das solicitações de transações.



(a) VM (Small)



(b) VM (Medium)



(c) VM (xLarge)

Figura 5.7: Uso de CPU (%) em chamada de algoritmo em complexidade O(n).

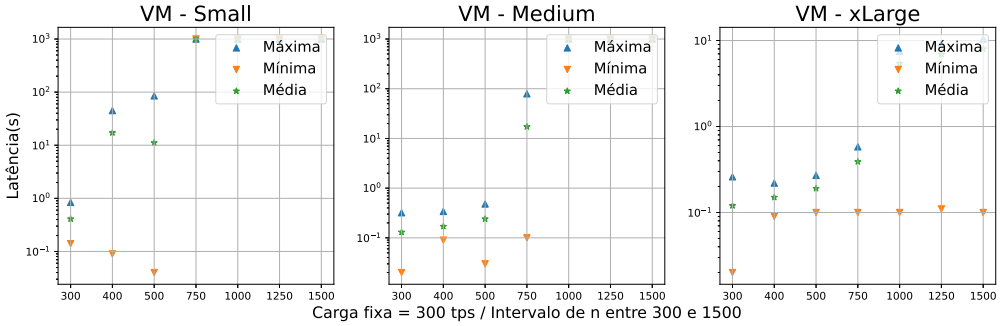


Figura 5.8: Latência em diferentes VMs com algoritimo em complexidade O(n)

Experimento 3 - Transações com complexidade em processamento em vm XLarge

O experimento três foi proposto devido à VM Xlarge, que é a VM de maior capacidade de processamento utilizada nesse trabalho, ter alcançado resultados de entradas maiores e sem perda, nos testes realizados pelo algoritmo de maior complexidade. Propôs-se então avaliar o desempenho da VM Xlarge mediante o crescimento gradativo da entrada no contrato que implementa um algoritmo de complexidade $O(n)$. Uma vez que, ao considerar o aumento do tempo de latência e o nível de utilização do recurso de processamento, pode-se verificar que a VM encontra-se sobrecarregada devido ao esgotamento dos recursos computacionais.

Por fim, o experimento 3 demonstra que, apesar da utilização da CPU não ultrapassar os 60 %, as transações com n superior a 1500 não são possíveis de processar devido à limitação imposta pela EVM.

Carga x Latência com complexidade em processamento em VMs

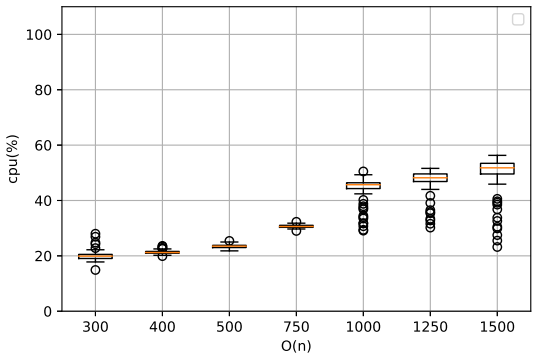


Figura 5.9: Uso de CPU(%) em variação de n em O(n)

A seguir, será incluído o fator de custo, analisando a melhor relação entre custo e desempenho observada para as infraestruturas avaliadas.

5.1.4 Compromisso entre Custo e Desempenho

Esta seção busca responder à segunda pergunta: *A arquitetura requer um nó de alto custo para executar transações com níveis razoáveis de desempenho?* Para isso foi aplicado às latências observadas na seção anterior um peso proporcional ao custo do tipo do nó por hora, como é usual na precificação de recursos computacionais, o que nos permite analisar o compromisso entre custo e desempenho.

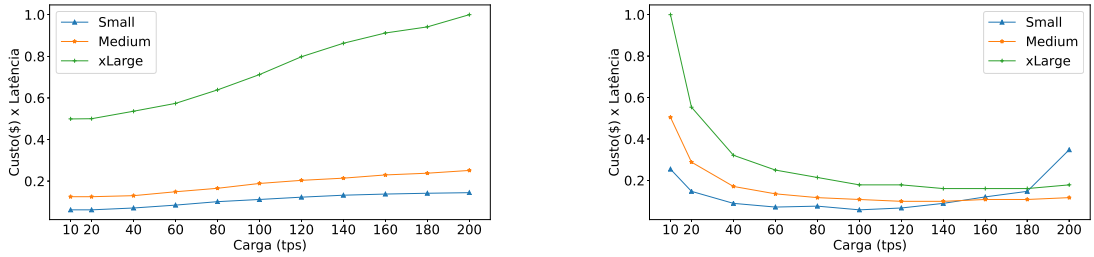
A Figura 5.10 mostra esse compromisso para os três tipos de nós em função da carga (tps). Cada tipo é representado por uma curva obtida via o produto $custo \times latência$ normalizado, onde o custo se refere aos preços praticados pela Amazon mostrados na Tabela 5.2. Assim, o tipo de nó com o menor valor desse produto é recomendado nessa análise em função da carga de trabalho. Nós da plataforma Ethereum mostrados na Figura 5.10 (a), notavelmente, temos o melhor compromisso entre custo e desempenho com o tipo *small*, pois não há diferenças relevantes para as latências observadas entre esse nó e os demais, mesmo com o aumento da carga, logo, o fator custo é dominante na escolha do tipo de nó. O tipo *small* é recomendado para cargas menores que 140 tps, e a partir dessa marca, o tipo *medium* é mais indicado, i.e., os ganhos com latência do nó *medium* compensam o seu custo, que é duas vezes superior ao do nó tipo *small*.

Alguns entendimentos interessantes sobre negócios envolvendo blockchain podem ser obtidos com essa análise. Primeiro, serviços de infraestrutura em blockchains públicas como Ethereum se tornam viáveis com o baixo custo de um nó nessa rede, em especial após a adoção do consenso Prova de Participação. Tomando como exemplo a Infura, que é o serviço mais popular atualmente, um único nó *small* pode atender com uma carga de 200 tps até 86 pagadores do plano básico[1], gerando uma receita expressiva (cerca de 250 vezes maior) em relação ao custo mensal desse nó.

A arquitetura avaliada neste trabalho unifica os componentes ordenador e par em um único nó completo, possibilitando estimar um teto de custo (i.e., uma estimativa conservadora) para uma organização participar numa rede blockchain permissionada. Em resposta à pergunta dessa seção, é elucidado que essa arquitetura requer um nó com capacidade intermediária (p.ex., *aws t2.medium*) para executar cargas intensivas com alto desempenho.

No modelo de custo apresentado (Seção 4.1.5), é necessário identificar a vazão máxima suportada em redes blockchain com recursos computacionais diferentes, especificamente, redes cujos nós tenham as capacidades apresentadas na Tabela 5.2. Intuitivamente, o crescimento da vazão está associado ao aumento do consumo de recursos computacionais, assim como a superutilização desses recursos pode levar à limitação da vazão. Nesse sentido, foram examinados quais recursos do nó são mais consumidos com o aumento da carga na rede blockchain, indicando a vazão máxima que pode ser alcançada nessa rede

[1] O plano *developer* atende até 200 mil requisições/dia a US$ 50/mês (https://www.infura.io/pricing).

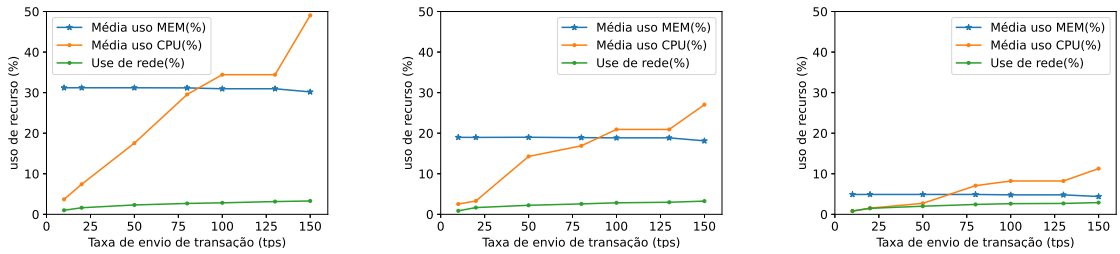


(a) Ethereum (b) Hyperledger Fabric

Figura 5.10: Compromisso entre custo e desempenho (normalizado) por nó em função da carga: melhores compromissos são os valores menores das curvas.

por contenção desses recursos.

As Figuras 5.11(a), (b) e (c) mostram o consumo médio de CPU, memória e rede[2] para cargas sintéticas submetidas à plataforma Ethereum, construída com nó do tipo *medium* com a finalidade de observar quais desses recursos são mais requisitados, preliminarmente ao início dos experimentos. Como pode ser observado, CPU é o recurso que tem o uso mais impactado com os aumentos de carga, i.e., a taxa do envio de transações, ao passo que o uso de memória permanece estável e o uso da rede (entrada e saída) cresce em relação à sua capacidade máxima (1 Gbps), mas não tão significativamente quanto CPU. Portanto, o foco é no uso de CPU para identificar a vazão máxima nos três tipos de infraestrutura para as redes blockchain, e estabelecemos o limite de 100 % de uso de CPU para o conjunto de infraestrutura alvo.



(a) VM (Small) (b) VM (Medium) (c) VM (xLarge)

Figura 5.11: Consumo médio de recursos para VMs do tipo Small, Medium e xLarge.

[2]Disco foi omitido dessa análise visto que a aplicação típica proposta para a avaliação foca no armazenamento *offchain*, como descrito na seção anterior.

5.2 Experimentos utilizando o Token de Cessão Multicadeia

5.2.1 Descrição e Configuração

Os experimentos com o Token de Cessão Multicadeia CST_{mc} foram conduzidos de maneira a obter resultados que favoreçam a análise sobre os custos de transações e desempenho dos protocolos de interoperabilidade. Para analisar o desempenho e custo do uso de protocolos de interoperabilidade, juntamente com a extensão ao CST_{mc}, foram projetados experimentos que utilizam duas redes de testes de blockchains, sendo essas redes oriundas de plataformas públicas e populares entre pesquisadores e desenvolvedores de aplicações descentralizadas. As análises apresentadas são provenientes dos cenários com requisitos de submissão de transações por meio da variação da utilização de dois protocolos para interoperabilidade do CST_{mc} entre as redes de teste.

O tempo despendido por um protocolo para prover a interoperabilidade entre redes blockchain depende de fatores que podem ser determinantes no tempo de execução total da operação. Primeiramente, para uma transação ser confirmada na blockchain, é necessário um processo de verificação e validação dos dados da transação que demanda tempo. Outro ponto importante em relação ao tempo é que o prazo necessário para a obtenção desses dados depende da plataforma que está sendo utilizada. Vários fatores como congestionamento, prioridade impostas por taxa ou até mesmo a complexidade dos algoritmos implementados nos contratos inteligentes podem influenciar o tempo total da operação, conforme apresentado na seção 5.1.3 desse capítulo.

Nos experimentos com o CST_{mc}, as transações foram disseminadas através de duas direções paralelamente: sendo uma abordando a rede de teste *Amoy* como origem e a rede *Fuji* como destino, e a outra invertendo a origem e destino dessas duas redes. Portanto, para o estudo e avaliação do tempo e das tarifas aplicadas nessas transações, optou-se por configurar o envio de transações a cada intervalo de tempo pré-definido entre as duas redes de teste e nas duas direções. As tarifas cobradas para transações de interoperabilidade entre blockchains são definidas com base em uma combinação de fatores e estão diretamente ligadas às redes que usarão o serviço, nesse caso, as redes Amoy e Fuji. A junção desses fatores resulta no valor tarifário total a ser cobrado pelo serviço de interoperabilidade, Equação 5.1.

$$Custo\ da\ Interoperabilidade = \sum_{i=1}^{n} Custo\ da\ transação(i) \tag{5.1}$$

De modo geral, quem paga a tarifa é a origem, ou seja, na transação da rede Amoy para a rede Fuji, a conta de usuário que inicia as transações na Amoy paga as taxas para registro das transações devidas ao protocolo específico utilizado. Como cada plataforma

tem sua própria criptomoeda ou token, e baseando-se no exemplo citado, as tarifas ficam da seguinte forma: Quando a Amoy for a origem, as taxas são pagas na moeda AVAX, e a Fuji, como destino, paga em POL.

5.2.2 Ambiente Experimental e Métricas

Algumas plataformas blockchain possuem redes de testes, as (*Testnets*), que são ambientes de testes que possibilitam desenvolver, experimentar e validar aplicativos, contratos inteligentes e funcionalidades antes de uma implementação real na rede blockchain principal, a *mainnet*. As redes de testes conseguem replicar o comportamento da rede principal, com algumas diferenças importantes: Os tokens não têm valor financeiro (não se aplica ou retira moeda financeira); Possibilita o uso de contratos inteligentes sem colocar em risco os softwares, fundos e usuários reais da rede; A acessibilidade aos ambientes, pois os desenvolvedores podem obter gratuitamente tokens das redes de testes gratuitamente através de "faucets"; As medidas de tempo das redes de testes dificilmente serão iguais às redes de produção, logo o número de transações nas redes principais são bem maiores do que nas redes de testes e isso pode alterar consideravelmente o valor das taxas de transações cobradas, como também o valor do GAS, mas as redes de testes e principais utilizam os mesmos métodos na hora de realizar e taxar as operações. Como o objetivo é avaliar o custo e o desempenho dos três protocolos citados, a diferença de valores avaliados entre os protocolos terá a mesma proporção se compararmos os resultados das redes de testes com as redes principais.

Para esses experimentos, optou-se pelas redes principais *Polygon* e *Avalanche* e suas redes de testes *Amoy* e *Fuji*, respectivamente. A rede *Amoy* é uma rede de testes desenvolvida para a blockchain *Polygon* que utiliza o (*Proof-of-Stake*) como protocolo de consenso, servindo como um ambiente de baixo risco para desenvolvedores construírem, testarem e aperfeiçoarem suas aplicações antes de implantá-las na rede principal. Criada em janeiro de 2024, utiliza a *Sepolia* (uma *testnet* do Ethereum) como sua cadeia raiz (L1), garantindo uma infraestrutura sustentável. Essa configuração permite que desenvolvedores continuem contando com validadores essenciais, ferramentas de infraestrutura, "*faucets*" e outros recursos necessários para o desenvolvimento eficiente (Polygon, 2024).

A Avalanche Fuji, rede de testes oficial da Avalanche, possibilita que desenvolvedores testem aplicações descentralizadas e contratos inteligentes sem custos financeiros associados às transações. A Avalanche é uma plataforma blockchain de alto desempenho que utiliza o protocolo de consenso Avalanche, que é baseado em *proof of stake*, conhecido por sua baixa latência e capacidade de processar milhares de transações por segundo (TPS).

Apesar das redes de testes não consumirem valores financeiros reais, e sim "faucets"

que são fornecidos gratuitamente, vale ressaltar que cada rede blockchain tem seu próprio token para transações. A Polygon utiliza o token chamado POL como sua moeda, enquanto a Avalanche utiliza o AVAX como token da sua moeda.

O mecanismo notarial necessita de um terceiro confiável para gerenciar as transações entre a blockchain de origem e destino. Dessa maneira, abstraiu-se o terceiro confiável por meio de um contrato inteligente. Esse contrato atua como o Notário do mecanismo recebendo o token do remetente da rede "A", e o transferindo para o destinatário na rede "B". Esse mecanismo possui limitações de segurança, atuando de maneira centralizada, sendo assim, a transação depende da integridade do Notário. Se o Notário for comprometido, pode ocasionar perdas financeiras aos usuários das redes.

O mecanismo HTLC necessita de implementação de contratos inteligentes responsáveis pela troca segura dos ativos. Esses contratos são implantados nas duas redes e possuem métodos de conectá-las. Como nesse mecanismo não há um terceiro confiável, o contrato atua sincronizando as redes com as funções de verificação das transações, da palavra secreta e da devolução dos valores, caso seja necessário abortar a transação. Sendo assim, o contrato para o HTLC possui as funcionalidades de bloquear os fundos que serão transferidos, registrar o horário da transação e exigir uma palavra secreta no momento da retirada dos fundos para o destinatário da segunda rede. Caso o tempo tenha ultrapassado o período de bloqueio, o contrato é capaz de devolver a quantia para o remetente.

A preparação do ambiente para a execução dos experimentos se deu por meio de scripts implementados na linguagem *Javascript* e contratos inteligentes na linguagem *Solidity*. Os experimentos foram executados a partir da configuração de máquinas virtuais da AWS-EC2[3]. Crontab é utilizado para agendamento e execução de comandos e scripts de maneira recorrente. Desse ponto em diante, configuramos as execuções por agendamentos via Crontab para que a cada 30 minutos fossem enviadas uma transação para a rede Amoy e outra para a rede Fuji. Também foi agendado, a cada 30 minutos, envio de uma transação da rede Fuji para a rede Amoy, mas com uma diferença de 5 minutos entre o envio Amoy/Fuji e Fuji/Amoy. As transações em questão são transferências de tokens CST_{mc} entre contas distintas em cada uma dessas redes testadas.

5.2.3 Resultados

São discutidos nesta seção os resultados da análise comparativa entre os protocolos de interoperabilidade, Notarial e HTLC, que foram implementados nesse trabalho de acordo com propostas recentes da literatura. Portanto, a análise visa comparar os dois protocolos com as métricas definidas e apresentadas. Onde o aspecto desempenho é medido em termos de tempo da operação de interoperabilidade e o aspecto custo é medido em termos

[3]Amazon Web Services - Elastic Compute Cloud - https://docs.aws.amazon.com/ec2

de valores da tarifa dessa operação.

5.2.3.1 Desempenho em tempo de operação

A Figura 5.12 apresenta a distribuição de tempos de operação para os dois mecanismos nas duas direções de redes de origem e destino, i.e., da rede Amoy para Fuji, assim como Fuji para Amoy. O gráfico apresenta *boxplots* que sumarizam a distribuição da seguinte forma: o retângulo central se expande entre o primeiro e o terceiro quartil, o segmento interior é a mediana, enquanto os indicadores abaixo e acima do retângulo representam o 10^{o} e o 90^{o} percentis. Adicionalmente, pontos acima e abaixo dos indicadores representam valores atípicos (*outliers*).

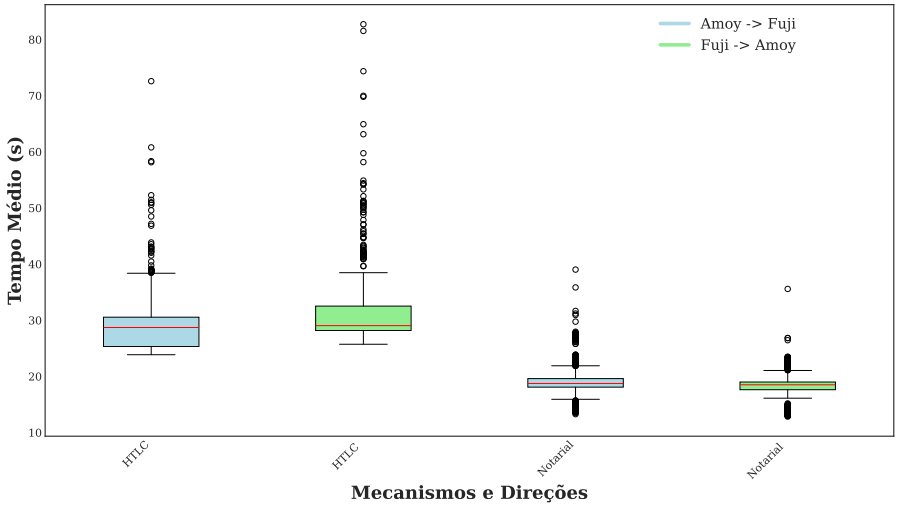


Figura 5.12: Distribuição do tempo de operação.

Observa-se na Figura 5.12 que o protocolo Notarial apresenta distribuição de tempos de operação com valores menores, ao passo que o protocolo HTLC tem valores maiores. Portanto, o protocolo Notarial tem melhor desempenho em termos de tempo de operação, o que é explicado devido à sua simplicidade, i.e., envolver apenas uma entidade central, o Notário, para interoperar tokens entre duas redes. Por sua vez, o protocolo HTLC tem desempenho menor, com tempos de operação de 29 segundos no sentido Fuji-Amoy e Amoy-Fuji [4]. Esse tempo é próximo ao Notarial, mas diferentemente deste, o HTLC é totalmente descentralizado, como descrito na Seção 2.5.1.5.

Foi analisado também o tempo de operações, distribuído ao longo das horas do dia (horário GMT), para maior detalhamento no desempenho dos protocolos. As Figuras 5.13 e 5.14 mostram esses tempos via médias e seus respectivos erros com intervalo de confiança de 95%. Os protocolos Notarial e HTLC permanecem com tempos de operação estáveis ao longo do dia com tempos médios inferiores a 40 segundos (i.e., menos de um minuto) por operação. Novamente, nota-se que esses dois protocolos têm desempenhos muito

[4]Os tempos de interação de usuários humanos na operação são desconsiderados

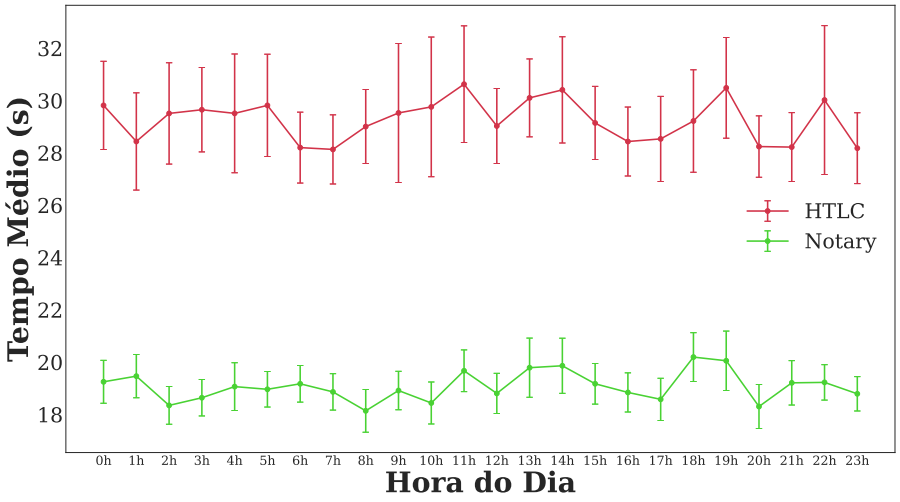


Figura 5.13: Tempo médio de operação ao longo do dia com erro da média em confiança de 95%. Amoy para Fuji

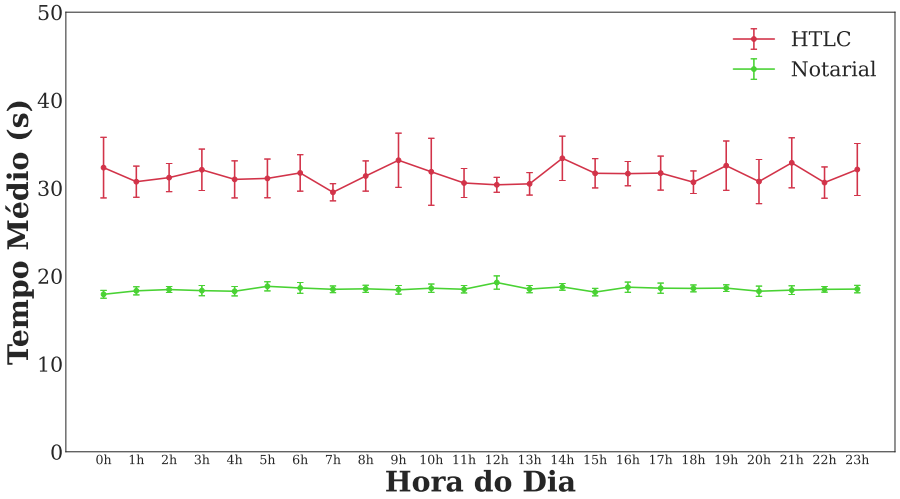


Figura 5.14: Tempo médio de operação ao longo do dia com erro da média em confiança de 95%. Fuji para Amoy

similares: os tempos médios de Notarial e HTLC se mantêm majoritariamente em 20 e 35 segundos ao longo do dia, respectivamente, sem diferenças significativas em algumas horas, i.e., há sobreposições entre erros das médias.

5.2.3.2 Custo em tarifas da operação

A Figura 5.15 apresenta a distribuição de custos de operação para os dois mecanismos nas duas direções de redes de origem e destino. *Boxplots* são utilizados assim como na Figura 5.12, mas aqui representando o custo total de tarifas por operação em cada mecanismo, convertido em dólar americano (USD). É importante observar que cada rede envolvida na operação de interoperabilidade tem tarifa distinta expressa no valor do token nativo da rede (i.e., POL na Amoy e Avax na Fuji). Logo, convertemos os valores de cada tarifa em USD de acordo com a cotação no momento da operação para fins de análise do custo de interoperabilidade.

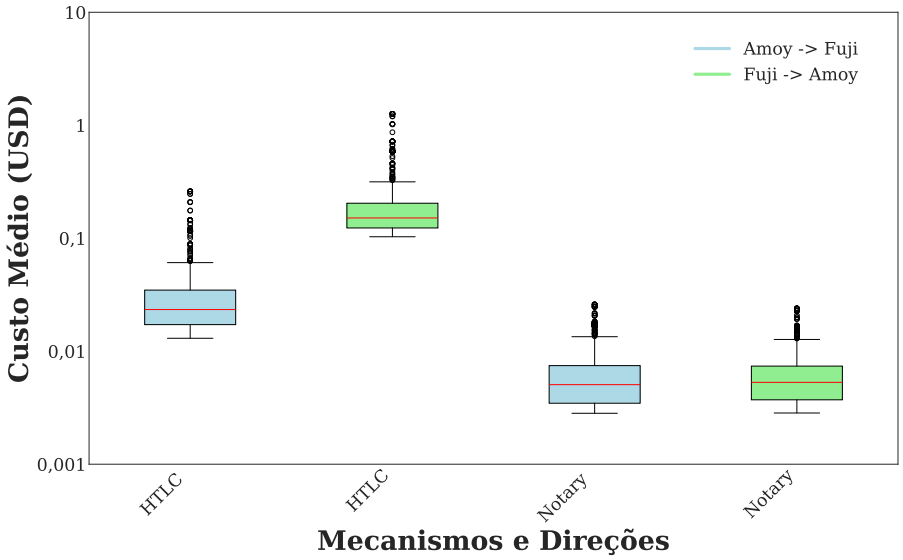


Figura 5.15: Distribuição do custo de operação.

Observa-se na Figura 5.15 que o protocolo Notarial tem o menor custo, i.e., cerca de 90% das operações (indicador do percentil 90^o.) têm valores abaixo de 1 centavo de dólar. Por sua vez, o protocolo HTLC tem valores entre 10 centavos e 1 dólar. Nesse caso, observa-se que a rede de destino tem maior impacto no custo, dado que a direção Amoy-Fuji tem distribuição de valores de custos superior e o token Avax tem custo em dólar superior ao POL atualmente.

A Figura 5.15 mostra também que o protocolo HTLC é, notavelmente, impactado pela direção da operação. Isso ocorre porque a rede de origem é responsável pela implementação do contrato na blockchain, ao passo que a rede de destino realiza apenas a transação de resgate dos tokens bloqueados pela origem previamente, como foi descrito na Seção 2.5.1.5. Usualmente, implementação de contratos tem custo computacional (gas) maior em redes

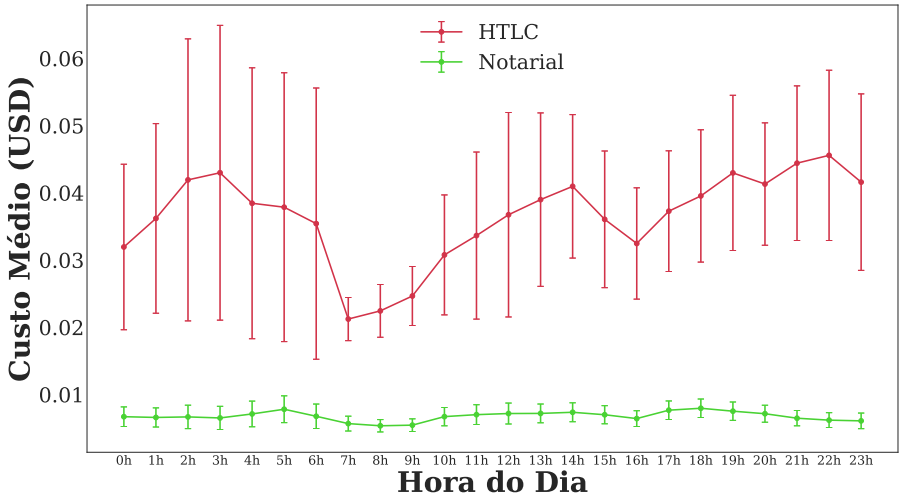


Figura 5.16: Custo médio de operação ao longo do dia com erro da média em confiança de 95%. Amoy para Fuji

blockchain, e por conseguinte tarifa maior. Logo, a direção Fuji-Amoy tem custo maior, pois o custo mais elevado do token Avax na rede Fuji para a implementação do contrato torna o HTLC mais oneroso nessa direção.

Detalhamos o custo de operação dos protocolos distribuídos ao longo das horas do dia (horário GMT), assim como na análise de desempenho. As Figuras 5.16 e 5.17 mostram esses custos em dólares (USD), via médias e seus respectivos erros, com intervalo de confiança de 95%. É possível observar, dessa vez, alta instabilidade ao longo do dia no protocolo HTLC. O custo do protocolo HTLC tem alta variação na direção Fuji-Amoy devido ao impacto da tarifa Fuji na origem da rede (i.e., o token Avax tem maior custo em dólares), fazendo com que os custos médios variem, enquanto o protocolo Notarial tem custo estável abaixo de 5 centavos de dólares.

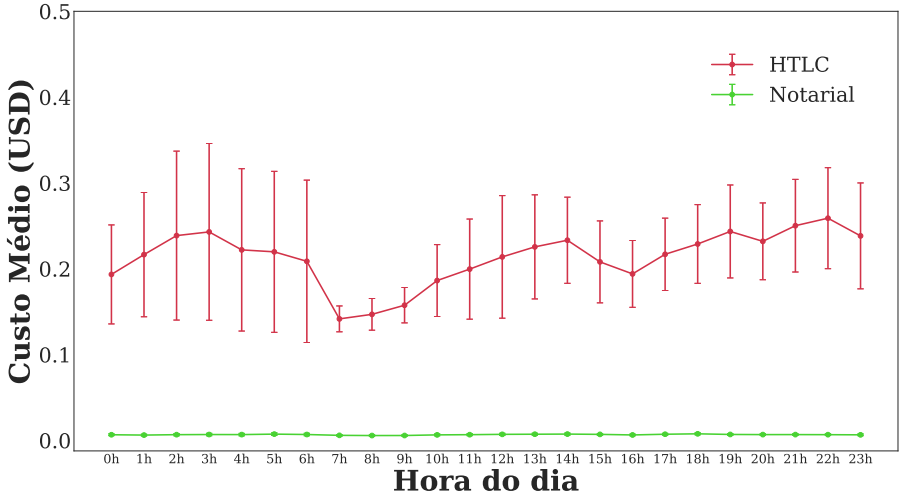


Figura 5.17: Custo médio de operação ao longo do dia com erro da média em confiança de 95%. Fuji para Amoy

5.3 Considerações Finais

Neste capítulo, foram realizados e analisados experimentos fundamentais para a validação prática das propostas apresentadas ao longo da pesquisa. Inicialmente, os testes com o Token de Cessão permitiram avaliar o impacto da complexidade de contratos inteligentes e da variação de carga de trabalho sobre o desempenho de nós *endpoints* em plataformas blockchain públicas. Os resultados demonstraram que, embora máquinas de menor porte possam oferecer desempenho aceitável sob cargas limitadas, há um limite claro de escalabilidade condicionado ao uso intensivo de CPU e à complexidade algorítmica dos contratos utilizados.

A análise da latência e vazão sob diferentes configurações de máquinas virtuais mostrou que a escolha da infraestrutura impacta diretamente no custo-benefício da operação, permitindo recomendações práticas baseadas no equilíbrio entre desempenho e custo monetário. A aplicação da arquitetura proposta revelou-se eficiente, especialmente ao considerar o nó medium como alternativa viável para operações em larga escala com desempenho satisfatório e custo controlado.

Na sequência, os experimentos de interoperabilidade multicadeia trouxeram contribuições importantes ao avaliar os protocolos Notarial e HTLC na operação entre redes de teste Polygon (Amoy) e Avalanche (Fuji). A análise revelou que, embora o protocolo Notarial apresente melhor desempenho e menor custo, o protocolo HTLC se destaca por operar de forma mais segura e descentralizada — ainda que com maior consumo de recursos e tarifas associadas às transações.

Por fim, o estudo comparativo entre desempenho e custo dos protocolos e das infraestruturas utilizadas possibilitou identificar o compromisso ideal para diferentes cenários de aplicação. As observações práticas obtidas neste capítulo fundamentam a aplicabilidade das soluções propostas na tese, reforçando seu potencial de adoção em sistemas reais de rastreabilidade, controle de acesso e interoperabilidade blockchain.

Capítulo 6

Conclusões

Apresentamos um estudo desenvolvido com o objetivo de verificar as possibilidades de uso da Blockchain para a validação e controle de propriedade de ativos em múltiplas cadeias por meio do Token de Cessão. A pesquisa utilizou inicialmente a tecnologia Blockchain para armazenar e validar dados em forma de *hashes* criptográficos. A arquitetura proposta e os componentes utilizados no desenvolvimento são apresentados de forma a favorecer um acesso facilitado à tecnologia Blockchain. Definimos os contratos inteligentes como as soluções baseadas em Blockchain para esse tipo de aplicação e devem receber uma atenção especial com o intuito de progredir e validar as possibilidades de uso da Blockchain em soluções de validação e controle de propriedade de tokens. Nossas implementações demonstram que o uso da Blockchain para a finalidade apresentada é viável e pode oferecer transparência nas transações.

A Blockchain é uma área de pesquisa e desenvolvimento recente, com grandes desafios a serem superados para que possa ser amplamente implementada e utilizada. As aplicações voltadas para o controle e compartilhamento de dados são capazes de ofertar inúmeros benefícios para os usuários finais. Dentre as diversas questões de pesquisa relacionadas ao compartilhamento de dados estão a segurança e a privacidade. Como contribuição para esta questão, apresentamos no Capítulo 4 uma solução capaz de tratar concessão e revogação de acesso a registros eletrônicos de saúde baseada em Blockchain. Os contratos inteligentes foram utilizados como solução para registrar as concessões e revogações de acesso em Blockchain. A concessão e revogação de acessos foram elaboradas de forma abrangente, resultando em uma estrutura capaz de oferecer suporte para qualquer aplicação desenvolvida para tal finalidade e que siga as regras de negócio estabelecidas pelo contrato. A arquitetura consiste na integração de soluções definidas de maneira que o paciente possa ter o controle de acesso aos seus dados e atenda aos requisitos de aplicações centradas no paciente.

Com o intuito de subsidiar a pesquisa com dados que corroboram a viabilidade de utilização das aplicações em quesitos de custo e desempenho, no Capítulo 4 propusemos uma avaliação da infraestrutura Blockchain, necessária para prover acesso de aplicações à rede, traçando uma comparação de desempenho entre as plataformas *Ethereum* e *Hyperledger Fabric*. Avaliamos por meio de um modelo de custo por transação para

aplicações em redes Blockchain públicas e permissionadas, considerando simultaneamente o desempenho máximo em função da infraestrutura e carga de trabalho imposta. Realizamos um experimento com a implementação de aplicações nas duas plataformas para aplicarmos o modelo em diferentes tipos de infraestrutura. Como resultado, fornecemos um modelo capaz de estimar o custo da infraestrutura por transação confirmada na Blockchain, considerando redes públicas e permissionadas. A partir do modelo proposto, nossos resultados mostraram os limites de escalabilidade dessas redes e os compromissos entre custo e desempenho para aplicações Blockchain.

Apresentamos ainda os conceitos, aplicações e desafios da tokenização e o padrão ERC-721 para tokens não fungíveis na Seção 2.3. Onde definimos os fundamentos essenciais para um bom entendimento dos tokens passando pelas blockchains, contratos inteligentes, plataformas, carteiras, oráculos, aplicações e finanças descentralizadas, Daos e os cenários de aplicações. Em seguida, foram apresentados as estruturas e os padrões de tokens utilizados em rede Blockchain. Padrões estes que deram origem aos *NFTs* e que tomaram grande proporção no ano de 2021, com milhões de transações realizadas em blockchains. Foram elencados os principais aspectos de segurança relacionados ao desenvolvimento de contratos inteligentes para tokens, bem como os desafios desta tecnologia emergente, que incluem a interoperabilidade entre blockchains.

No Capítulo 4 apresentamos como os tokens podem fornecer a certificação de propriedade de arte digital, documentos, direitos de propriedade intelectual, licenças e marcas por meio de registros em Blockchain permanentemente. Desta maneira vimos as funções de criação e cessão de um token, podendo armazená-lo em uma carteira de ativos digitais e cedê-lo virtualmente. Como resultado destes registros, podemos verificar a verdadeira propriedade e origem de ativos de forma confiável. Por fim, este capítulo objetivou proporcionar uma base de conhecimento e visão do estado da arte para a área de tokens em Blockchain, bem como possibilitar o avanço do trabalho para proceder com projetos e implementações das soluções para tokens em múltiplas cadeias. Os próximos passos para essa pesquisa envolvem a ampliação da avaliação da metodologia de forma sincronizada em múltiplas cadeias, fazendo com que os recursos sejam sincronizados e possam ser avaliados em grandes demandas.

A interoperabilidade de blockchains possibilita uma gama de soluções antes fragmentadas no universo blockchain. Avaliou-se o desempenho das transações em redes blockchains utilizando o protocolo Notarial e o HTLC, levando em consideração métricas de tempo de processamento e custo operacional de transação. Os resultados evidenciam que cada abordagem possui vantagens e desafios distintos, dependendo do cenário da aplicação. Para trabalhos futuros, a indicação é analisar a escalabilidade e mitigação de ataques específicos, além de avaliar outros modelos híbridos que combinem a eficiência de cada um dos protocolos de interoperabilidade analisados.

Referências Bibliográficas

Alhussayen, A. A., Jambi, K., Khemakhem, M., and Eassa, F. E. (2024). A blockchain oracle interoperability technique for permissioned blockchain. *IEEE Access*.

Androulaki, E. and et al. (2018). Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In *Proc. of the EuroSys Conference*.

Antonopoulos, A. M. (2014). *Mastering Bitcoin: unlocking digital cryptocurrencies*. "O'Reilly Media, Inc.".

Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Timón, J., and Wuille, P. (2014). Enabling blockchain innovations with pegged sidechains. *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, 72:201–224.

Baliga, A., Solanki, N., Verekar, S., Pednekar, A., Kamat, P., and Chatterjee, S. (2018). Performance characterization of hyperledger fabric. *Proceedings - 2018 Crypto Valley Conference on Blockchain Technology, CVCBT 2018*, pages 65–74.

Banaeian Far, S., Hosseini Bamakan, S. M., Qu, Q., and Jiang, Q. (2022). A review of non-fungible tokens applications in the real-world and metaverse. In *Procedia Computer Science*, volume 214, pages 755–762. Elsevier.

Belchior, R., Vasconcelos, A., Guerreiro, S., and Correia, M. (2021). A survey on blockchain interoperability: Past, present, and future trends. *ACM Computing Surveys (CSUR)*, 54(8):1–41.

Bellavista, P., Esposito, C., Foschini, L., Giannelli, C., Mazzocca, N., and Montanari, R. (2021). Interoperable blockchains for highly-integrated supply chains in collaborative manufacturing. *Sensors*, 21(15):4955.

Besançon, L., Silva, C. F. D., and Ghodous, P. (2019). Towards blockchain interoperability: Improving video games data exchange. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 81–85.

Bjelic, M., Nailwal, S., Chaudhary, A., and Deng, W. (2017). Pol: one token for all polygon chains.

Buterin, V. (2016). Chain interoperability. *R3 research paper*, 9:1–25.

Caliper, H. (2019). Caliper. https://hyperledger-caliper.github.io/caliper. (Accessed on 09/23/2021).

Cao, Y., Cao, J., Bai, D., Wen, L., Liu, Y., and Li, R. (2024). Map the blockchain world: A trustless and scalable blockchain interoperability protocol for cross-chain applications. *arXiv preprint arXiv:2411.00422*.

Chainlink (2023). Cross-chain interoperability protocol (ccip). https://chain.link/cross-chain.

Charles, P. (2013). Openzeppelin. https://github.com/OpenZeppelin.

Chevet, S. (2018). Blockchain technology and non-fungible tokens: Reshaping value chains in creative industries. *Available at SSRN 3212662*.

Chohan, U. W. (2021). Non-fungible tokens: Blockchains, scarcity, and value. *Critical Blockchain Research Initiative (CBRI) Working Papers*.

Choi, W. and Hong, J. W. K. (2021). Performance Evaluation of Ethereum Private and Testnet Networks Using Hyperledger Caliper. *22nd APNOMS 2021*, pages 325–329.

Entriken, W., Shirley, D., Evans, J., and Sachs, N. (2018). Eip-721: Non-fungible token standard,"ethereum improvement proposals, no. 721, january 2018. https://eips.ethereum.org/EIPS/eip-721.

Ethereum (2024). Sidechains. [Online; accessed 2024-06-23].

Fairfield, J. (2021). Tokenized: The law of non-fungible tokens and unique digital property. *Indiana Law Journal, Forthcoming*.

Farahani, B., Firouzi, F., Chang, V., Badaroglu, M., Constant, N., and Mankodiya, K. (2018). Towards fog-driven iot ehealth: Promises and challenges of iot in medicine and healthcare. *Future Generation Computer Systems*, 78:659 – 676.

Ghaemi, S., Rouhani, S., Belchior, R., Cruz, R. S., Khazaei, H., and Musilek, P. (2021). A pub-sub architecture to promote blockchain interoperability.(1 2021). *arXiv preprint arXiv:2101.12331*.

Gordon, W. J. and Catalini, C. (2018). Blockchain technology for healthcare: Facilitating the transition to patient-driven interoperability. *Computational and Structural Biotechnology Journal*, 16:224 – 230.

Greve, F., Sampaio, L., Abijaude, J., Coutinho, A. A., Brito, I., and Queiroz, S. (2018). Blockchain e a Revolução do Consenso sob Demanda. In *Proc. of SBRC Minicursos*.

Hammi, B., Zeadally, S., and Perez, A. J. (2023). Non-fungible tokens: A review. *IEEE Internet of Things Magazine*, 6(1):46–51.

Herlihy, M. (2018). Atomic cross-chain swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 245–254. ACM.

Hewa, T., Ylianttila, M., and Liyanage, M. (2021). Survey on blockchain based smart contracts: Applications, opportunities and challenges. *Journal of Network and Computer Applications*, 177:102857.

Huang, D., Ma, X., and Zhang, S. (2020). Performance analysis of the raft consensus algorithm for private blockchains. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(1):172–181.

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

Konashevych, O. (2020). General concept of real estate tokenization on blockchain. *European property law journal*, 9(1):21–66.

Kotey, S., Tchao, E. T., Ahmed, A.-R., Agbemenu, A., Nunoo-Mensah, H., Sikora, A., Welte, D., and Keelson, E. (2023). Blockchain interoperability: the state of heterogenous blockchain-to-blockchain communication. *IET Communications*, 17:1–24.

Kräussl, R. and Tugnetti, A. (2022). Non-fungible tokens (nfts): A review of pricing determinants, applications and opportunities. *CFS Working Paper Series*, 1(693).

Lamport, L. (2001). Paxos made simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pages 51–58.

Lamport, L., Shostak, R., and Pease, M. (2019). *The Byzantine generals problem*, page 203–226. Association for Computing Machinery, New York, NY, USA.

Leal, F., Chis, A. E., and González–Vélez, H. (2020). Performance Evaluation of Private Ethereum Networks. *SN Computer Science*, 1(5):1–17.

Lerner, S. D. (2015). Rsk. *RootStock Core Team, White Paper*.

Lotti, L. (2019). The Art of Tokenization: Blockchain Affordances and the Invention of Future Milieus. *Media Theory*, 3(1):287–320.

Ma, K., Huang, J., He, N., Wang, Z., and Wang, H. (2024). Sok: On the security of non-fungible tokens. *Blockchain: Research and Applications*, To appear:100268.

Malik, H., Manzoor, A., Ylianttila, M., and Liyanage, M. (2019). Performance Analysis of Blockchain based SG with Ethereum and Hyperledger Implementations. *IEEE International Conference on ANTS*.

Mendonça, R., Dantas, P., Gonçalves, G., Vieira, A., and Nacif, J. (2022a). Análise de custo de infraestrutura em redes blockchain públicas e permissionadas. In *Anais do V Workshop em Blockchain: Teoria, Tecnologias e Aplicações*, pages 26–39, Porto Alegre, RS, Brasil. SBC.

Mendonça, R., Moura, E., Gonçalves, G., Vieira, A., and Nacif, J. (2023). Comparação e análise de custo e desempenho entre nós de redes blockchain permissionadas e públicas. In *Anais do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 141–154, Porto Alegre, RS, Brasil. SBC.

Mendonça, R., Ítallo Cardoso, Coelho, R., Campos, J., Gonçalves, G., Vieira, A., and Nacif, J. (2024). Mecanismos de interoperabilidade em blockchains: Um comparativo de custo de transações cross-chain para tokens erc-20. In *Anais do VII Workshop em Blockchain: Teoria, Tecnologias e Aplicações*, pages 15–28, Porto Alegre, RS, Brasil. SBC.

Mendonça, R. D., Campos, J. N., Vieira, L. F., Vieira, M. A., Vieira, A. B., and Nacif, J. A. (2022b). Tokens não fungíveis (nfts): Conceitos, aplicações e desafios. *SBC*.

Monrat, A. A., Schelen, O., and Andersson, K. (2020). Performance Evaluation of Permissioned Blockchain Platforms. *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering, CSDE 2020*.

Muniz, F., Mendonça, R., de Miranda, E., Ítalo Cardoso, Coelho, R., Vieira, A., Nacif, J., and Gonçalves, G. (2025). Análise de custo e desempenho de protocolos para interoperabilidade de tokens em redes blockchain. In *Anais do VIII Workshop em Blockchain: Teoria, Tecnologias e Aplicações*, pages 1–14, Porto Alegre, RS, Brasil. SBC.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Nakamoto, S. et al. (2008). Bitcoin: A peer-to-peer electronic cash system. *Bitcoin*.

Noether, S. (2015). Ring signature confidential transactions for monero. *IACR Cryptol. ePrint Arch.*, 2015:1098.

Ou, W., Huang, S., Zheng, J., Zhang, Q., Zeng, G., and Han, W. (2022). An overview on cross-chain: Mechanism, platforms, challenges and advances. *Computer Networks*.

Pilkington, M. (2016). Chapter 11: Blockchain technology: principles and applications. In *Research Handbook on Digital Transformations*. Edward Elgar Publishing, Cheltenham, UK.

Polygon (2024). *Polygon Technology: Web3, Aggregated*. Acesso em: 05 nov. 2024.

Pupyshev, A., Gubanov, D., Dzhafarov, E., Sapranidi, I., Kardanov, I., Zhuravlev, V., Khalilov, S., Jansen, M., Laureyssens, S., Pavlov, I., et al. (2020). Gravity: a blockchain-agnostic cross-chain communication and data oracles protocol. *arXiv preprint arXiv:2007.00966*.

Qasse, I. A., Abu Talib, M., and Nasir, Q. (2019). Inter blockchain communication: A survey. In *Proceedings of the ArabWIC 6th Annual International Conference Research Track*, pages 1–6.

Radomski, W., Cooke, A., Castonguay, P., Therien, J., Binet, E., and Sandford, R. (2018). Eip-1155: Multi token standard,"ethereum improvement proposals, no. 1155, june 2018. https://eips.ethereum.org/EIPS/eip-1155.

Raman, R. K., Vaculin, R., Hind, M., Remy, S. L., Pissadaki, E. K., Bore, N. K., Daneshvar, R., Srivastava, B., and Varshney, K. R. (2018). Trusted multi-party computation and verifiable simulations: A scalable blockchain approach. *arXiv preprint arXiv:1809.08438*.

Razi, Q., Devrani, A., Abhyankar, H., Chalapathi, G., Hassija, V., and Guizani, M. (2024). Non-fungible tokens (nfts)—survey of current applications, evolution, and future directions. *IEEE Open Journal of the Communications Society*, 5:2765–2782.

Rimba, P., Tran, A. B., Weber, I., Staples, M., Ponomarev, A., and Xu, X. (2020). Quantifying the Cost of Distrust: Comparing Blockchain and Cloud Services for Business Process Execution. *Information Systems Frontiers*, 22(2):489–507.

Rogers, I., Carter, D., Morgan, B., and Edgington, A. (2022). Diminishing dreams: The scoping down of the music nft. *M/C Journal*, 25(2).

Rouhani, S. and Deters, R. (2017a). Performance analysis of ethereum transactions in private blockchain. In *2017 8th IEEE(ICSESS)*, pages 70–74. IEEE.

Rouhani, S. and Deters, R. (2017b). Performance analysis of ethereum transactions in private blockchain. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 70–74. IEEE.

Sguanci, C., Spatafora, R., and Vergani, A. M. (2021). Layer 2 blockchain scaling: A survey. *arXiv preprint arXiv:2107.10881*.

Spengler, A. C. and Souza, P. S. (2021). Avaliação de desempenho do hyperledger fabric com banco de dados para o armazenamento de grandes volumes de dados médicos. In *Proc. of WPerformance*.

Taherdoost, H. (2023). Non-fungible tokens (nft): A systematic review. *Information*, 14(1):26.

Tanenbaum, A. and van Steen, M. (2002). *Distributed Systems: Principles and Paradigms*. Prentice Hall.

Thakkar, P., Nathan, S., and Viswanathan, B. (2018). Performance benchmarking and optimizing hyperledger fabric blockchain platform. *Proceedings - IEEE, MASCOTS 2018*, pages 264–276.

Vinodhini, M. M. and Kavitha, M. S. (2021). Privacy-preserving storage of healthcare data executes block chain innovation nft and ipfs: A review with future directions. *Design Engineering*, pages 14310–14328.

Vogelsteller, F. and Buterin, V. (2015). Eip-20: Token standard, ethereum improvement proposals. https://eips.ethereum.org/EIPS/eip-20.

Wang, C. and Chu, X. (2020). Performance characterization and bottleneck analysis of hyperledger fabric. *Proceedings - International Conference on Distributed Computing Systems*, 2020-Novem:1281–1286.

Wang, G. (2021). Sok: Exploring blockchains interoperability. *Cryptology ePrint Archive*.

Wang, Q., Li, R., Wang, Q., and Chen, S. (2021a). Non-fungible token (nft): Overview, evaluation, opportunities and challenges. *arXiv preprint arXiv:2105.07447*.

Wang, Q., Li, R., Wang, Q., and Chen, S. (2021b). Non-fungible token (nft): Overview, evaluation, opportunities and challenges. Technical report, CSIRO Data61. arXiv preprint arXiv:2105.07447.

Wang, X. and Feng, J. (2018). The research of consortium blockchain dynamic consensus based on data transaction evaluation. In *2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, volume 2, pages 214–217. IEEE.

Wang, Y. and Kogan, A. (2018). Designing confidentiality-preserving blockchain-based transaction processing systems. *International Journal of Accounting Information Systems*, 30:1–18.

Wegner, P. (1996). Interoperability. *ACM Computing Surveys (CSUR)*, 28(1):285–287.

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

Xia, P., Wang, H., Zhang, B., Ji, R., Gao, B., Wu, L., Luo, X., and Xu, G. (2020). Characterizing cryptocurrency exchange scams. *Computers & Security*, 98:101993.

Xu, X., Sun, G., Luo, L., Cao, H., Yu, H., and Vasilakos, A. V. (2021). Latency performance modeling and analysis for hyperledger fabric blockchain network. *Information Processing and Management*, 58(1).

Xu, X., Weber, I., and Staples, M. (2019). *Architecture for blockchain applications*. Springer.

Yang, W., Aghasian, E., Garg, S., Herbert, D., Disiuta, L., and Kang, B. (2019). A survey on blockchain-based internet service architecture: requirements, challenges, trends, and future. *IEEE access*, 7:75845–75872.

Zamyatin, A., Al-Bassam, M., Zindros, D., Moreno-Sanchez, P., Knottenbelt, W., and Danezis, G. (2019). Xclaim: Trustless, interoperable, cryptocurrency-backed assets. *IEEE Symposium on Security and Privacy*.

Zhang, L., Lee, B., Ye, Y., and Qiao, Y. (2020). Ethereum transaction performance evaluation using test-nets. In *Euro-Par 2019: Parallel Processing Workshops*, Cham. Springer International Publishing.

Zhu, S., Chi, C., and Liu, Y. (2023). A study on the challenges and solutions of blockchain interoperability. *China Communications*, 20(6):148–165.

Apêndice A

Contribuições e Publicações

- Artigos como primeiro autor
  - **Interoperability of Cession Tokens as a Tool for Cost and Performance Assessment in Blockchain.** Ronan Dutra Mendonça, Alex B. Vieira, José A. Miranda Nacif. In: IEEE or ACM Journal ***(Submitted)***
  - **Performance and Cost Analysis of Protocols for Token Interoperability in Blockchain Networks.** Ronan Dutra Mendonça, Fredison M. de Sousa,Emanuel A. F. de Miranda, Ítallo W. F. Cardoso,Rafael Coelho,Josué N. Campos, Glauber D. Gonçalves, Alex B. Vieira, José A. Miranda Nacif. In: IEEE Communications Magazine ***(Submitted)***
  - **Mecanismos de Interoperabilidade em Blockchains: Um Comparativo de Custo de Transações Cross-chain para Tokens ERC-20**. Ronan D. Mendonça, Ítallo W. F. Cardoso, Rafael Coelho, Josué N. Campos, Glauber D. Gonçalves, Alex B. Vieira, José A. M. Nacif . XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2024) VII Workshop em Blockchain.***(Published)***
  - **Comparação e Análise de Custo e Desempenho entre Nós de Redes Blockchain Permissionadas e Públicas**. RD Mendonça, E Moura , GD Gonçalves, AB Vieira, JA Nacif In: Anais do 41º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. SBC, 2023 ***(Published)***
  - **Detecção de Vulnerabilidades em Contratos Inteligentes Utilizando Árvore Sintática Abstrata.** EVB Esquivel, JN Campos, RD Mendonça, AB Vieira, JA Nacif In: Anais do XXIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais. SBC, 2023. p. 335-348. ***(Published)***
  - **Análise de Custo de Infraestrutura em Redes Blockchain Públicas e Permissionadas.** RD Mendonça, PH Dantas, GD Gonçalves, AB Vieira, JA Nacif, Anais do V Workshop em Blockchain: Teoria, Tecnologias e Aplicações, 26-39 2022 ***(Published)***

  - **Tratamento de Concessão e Revogação de Acesso a Registros Eletrônicos de Saúde em Blockchain.** RD Mendonça, OS Gomes, AB Vieira, JA Nacif Anais do IV Workshop em Blockchain: Teoria, Tecnologias e Aplicações, 100-113 1 2021 ***(Published)***
  - **Utilização de Blockchain na Rastreabilidade da Cadeia Produtiva do Leite.** RD Mendonça, OS Gomes, PC Pereira, AB Vieira, JA Nacif Anais do III Workshop em Blockchain: Teoria, Tecnologia e Aplicações, 55-60 6 2020 ***(Published)***
  - **Performance and Cost Analysis of Protocols for Token Interoperability in Blockchain Networks.** Ronan Dutra Mendonça, Fredison Muniz, Emanuel F. de Miranda, Ítallo W. F. Cardoso, Rafael Coelho, Glauber D. Gonçalves, Alex B. Vieira, José A. Miranda Nacif ***(Submitted)***
  - **Blockcoldchain: Vaccine cold chain blockchain.** RD Mendonça, OS Gomes, LFM Vieira, MAM Vieira, AB Vieira, JAM Nacif arXiv preprint arXiv:2104.14357 9 2021 ***(Submitted)***
  - **Blockchain-based Model for Patient-Centered Control of Medical Records.** RD Mendonça, OS Gomes, AB Vieira, JAM Nacif ***(Submitted)***
  - **Non-Fungible Tokens (NFTs): Concepts, Applications and Challenges.** RD Mendonça, JN Campos, LFM Vieira, MAM Vieira, AB Vieira, JAM Nacif ***(Submitted)***

- Artigos como segundo autor

  - **Análise de Custo e Desempenho de Protocolos para Interoperabilidade de Tokens em Redes Blockchain.** MUNIZ, Fredison; MENDONÇA, Ronan D.; DE MIRANDA, Emanuel F.; CARDOSO, Ítalo W. F.; COELHO, Rafael; VIEIRA, Alex B.; NACIF, José A. M.; GONÇALVES, Glauber D. Anais do VIII Workshop em Blockchain: Teoria, Tecnologias e Aplicações, 2025 . p. 1-14. ***(Published)***
  - **Blockchain applied to the traceability of animal products: a systematic literature review.** Zanetoni, H. H. R., Queiroz, D. M. de ., Chizzotti, M. L., Mendonça, R. D., Baêta, F. da C., Coelho, A. L. de F., Nacif, J. A. M.. Revista Ciência Agronômica, v. 55, p. e20238702, 2024.***(Published)***
  - **Finanças Descentralizadas em Redes Blockchain: Perspectivas sobre Pesquisa e Inovação em Aplicações, Interoperabilidade e Segurança.** JN Campo, RD Mendonça, A Fontinele, LHS de Carvalho, IR Oliveira, Jornada de Atualização em Informática 2024 1, 199***(Published)***

- Capítulo de livro
  - **(Jornada de Atualização em Informática 2024). Finanças Descentralizadas em Redes Blockchain: Perspectivas sobre Pesquisa e Inovação em Aplicações, Interoperabilidade e Segurança.** Josué N. Campo, Ronan D. Mendonça, Alexandre Fontinele, Luís H. S. de Carvalho, Isdael R. Oliveira, Ítallo W. F. Cardoso, Rafael Coelho, Allan E. S. Freitas, Glauber D. Gonçalves, José A. M. Nacif, Alex B. Vieira. XLIV Congresso da Sociedade Brasileira de Computação - 20274 ***(Published)***
  - **(Minicurso SBRC 2022) Tokens Não Fungíveis (NFTs): Conceitos, Aplicações e Desafios.** Ronan Mendonça (UFV), Josué Campos (UFV), Luiz F. M. Vieira (UFMG), Marcos A. M. Vieira (UFMG), Alex Borges Vieira (UFJF), José A. M. Nacif (UFV) Minicurso SBRC 2022 ***(Published)***
- Aplicações (Registro de software)
  - **UFVBeefChain - Rastreio Carne UFV: Aplicação Blockchain para Rastreio de Carne Bovina no Brasil: Um estudo de caso.** 2022 - INPI (Processo No: BR512022002778-0) ***(Registered)***

Apêndice B

Resultados de Medições da Utilização de Recursos nas VMs Monitoradas em Experimentos

B.1 Monitoramento dos Rrecursos

Software desenvolvido utilizando a linguagem Python juntamente com a biblioteca *Psutil 5.9.0*. O código monitora dados sobre os recursos computacionais.

```
1  import psutil
2  import time
3  import logging
4  import threading
5  import os
6  import sys
7  timeOut = int(sys.argv[1])
8  cont = True
9  def aguardarEnter():
10     global cont
11     #input("\n*** Pressione ENTER para parar! ***\n")
12     time.sleep(timeOut)
13     cont = False
14
15 def monitorar():
16     global cont
17     num = 0
18     inicioentradaNET = psutil.net_io_counters()[0]
19     iniciosaidaNET = psutil.net_io_counters()[1]
20     inicioUsoDisco = psutil.disk_usage('/')[1]
21     logMonitor = logging.getLogger('logCPU_MEM')
22     logMonitor.info("horario,memoria(%),cpu(%),Block-use(bytes),Net-in(bytes),Net-out(bytes),TimeStamp")
23
24     while(cont == True):
25         #CPU_Pct=str(round(float(os.popen('''grep 'cpu ' /proc/stat | awk '{usage=($2+$4)*100/($2+$4+$5)} END {
               print usage }' ''').readline()),2))
26         cpu=psutil.cpu_percent()
27         mem = psutil.virtual_memory()[2]
28         entradaNET = psutil.net_io_counters()[0] - inicioentradaNET
29         saidaNET = psutil.net_io_counters()[1] - iniciosaidaNET
30         usoDisco = psutil.disk_usage('/')[1] - inicioUsoDisco
31         valor = str(mem) + "," + str(cpu) +"," + str(usoDisco)+ "," +str(entradaNET)+ "," + str(saidaNET)+ ","+str(
               int(time.time()))
32         logMonitor = logging.getLogger('logCPU_MEM')
33         logMonitor.info(valor)
34         num+=1
35         print(num,'-',valor )
36         time.sleep(1)
37
38 def setup_logger(logger_name, log_file):
```

```
39      log_setup = logging.getLogger(logger_name)
40      formatter = logging.Formatter('%(asctime)s,%(message)s', datefmt='%H:%M:%S')
41      filename = os.path.join(os.path.dirname(os.path.realpath(__file__)), log_file)
42      fileHandler = logging.FileHandler(filename, mode='a')
43      fileHandler.setFormatter(formatter)
44      log_setup.setLevel(logging.INFO)
45      log_setup.addHandler(fileHandler)
46
47
48  # -- MAIN -- #
49  setup_logger('logCPU_MEM', 'TxMonitor.csv')
50  print("Monitorando")
51  print("indice,memoria(%),cpu(%),Block-use(bytes),Net-in(bytes),Net-out(bytes),TimeStamp")
52
53  t1 = threading.Thread(target=aguardarEnter)
54  t2 = threading.Thread(target=monitorar)
55  t1.start()
56  t2.start()
57
58  t2.join()
```

Listagem B.1: Funções para criar e retornar um Token.

B.2 Recursos

Estes recursos são necessários para a execução e processamento dos experimentos, porém não têm sua quantidade limitadora.

Memória Principal

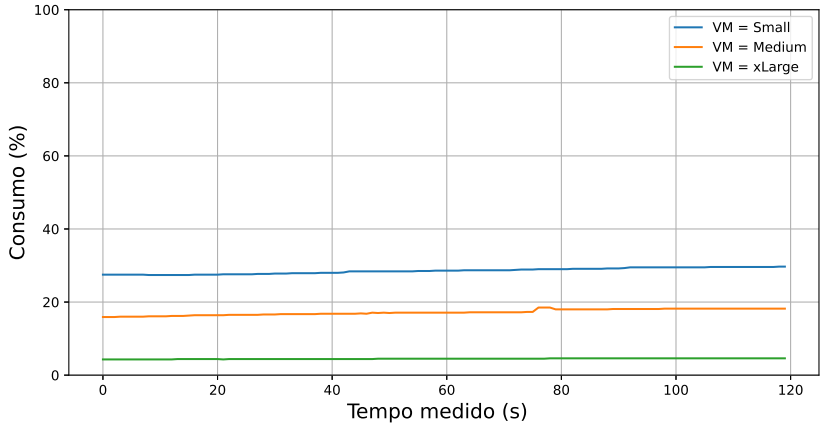


Figura B.1: Utilização do recurso "Memória"por VM

Armazenamento

Rede

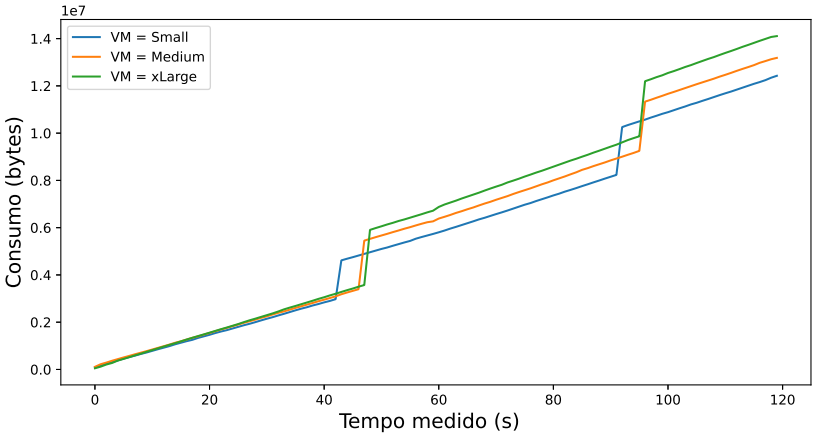


Figura B.2: Utilização do recurso "Armazenamento por"VM

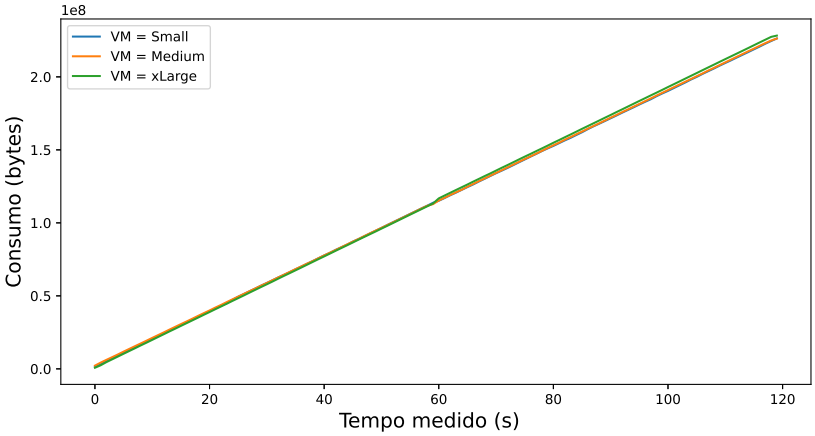


Figura B.3: Utilização do recurso "Rede in"por VM

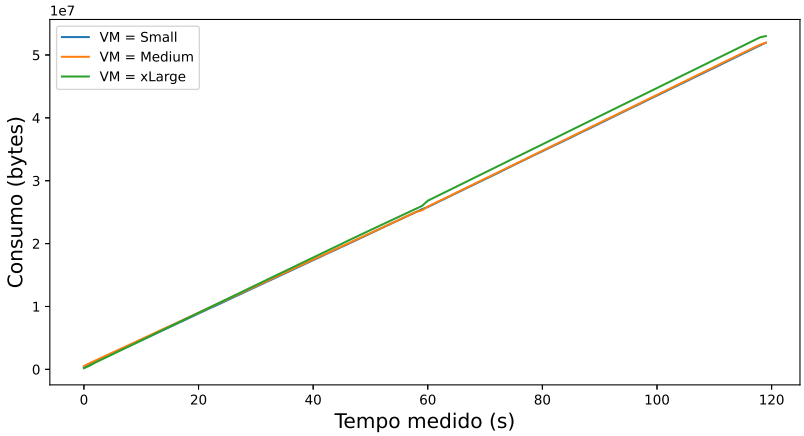


Figura B.4: Utilização do recurso "Rede out"por VM

Apêndice C

Codificação de Contratos CST, CST-Inter e Enlace

C.1 CST e CSTmc

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.27;
3
4  import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
5  import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol";
6
7  contract CST is ERC721{
8      uint256 private _nextTokenId;
9      address private _enlace;
10
11     event TokenId(uint256 tokenId);
12
13     constructor(address enlace) ERC721("Cessao", "CST"){
14         _enlace = enlace;
15     }
16
17     function get_enlace() public view returns (address){
18         return _enlace;
19     }
20
21     function criarNovoToken() public{
22         uint256 tokenId = _nextTokenId++;
23         _mint(msg.sender, tokenId);
24         approve(_enlace, tokenId);
25         emit TokenId(tokenId);
26     }
27
28     function transferirTokenExpirado(address from, address to, uint256 tokenId) external onlyEnlace {
29         _transfer(from, to, tokenId);
30     }
31
32     function burn(uint256 Id) external onlyEnlace{
33         _burn(Id);
34     }
35
36     modifier onlyEnlace{
37         require(msg.sender == _enlace, "Restrict - Only Enlace!");
38         _;
39     }
40 }
```

Listagem C.1: Funções para criar e retornar um Token.

C.1.1 Enlace

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.27;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

contract Enlace{
    // Contador para os IDs dos itens
    uint256 private _itemIds;
    // Contador para a quantidade de itens
    uint256 private _countItensCedidos;
    // Contador para a quantidade de itens devolvidos
    uint256  private _countItensDevolvidos;
    // Endereco do dono do contrato
    address payable _donoContrato;
    // Taxa cobrada pelo Enlance para um cedente
    // colocar o seu item disponivel para ceder
    uint256  private taxaEnlace;
    // Estrutura que mapeia um item cedivel
    struct Item{
        uint256 itemId; // Identificador do item
        bool statusCedido; // Status indicando se o item esta cedido (true) ou nao (false)
        address contratoCST; // Endereco do contrato que o CST pertence
        uint256 tokenId; // Identificador do CST
        address cedente; // Endereco do cedente responsavel pelo item
        address cessionario; // Endereco da pessoa que esta recebendo o item
        uint256 expiraEm; // Tempo de duracao da cessao
    }

    // Lista dos itens cediveis do enlace
    mapping(uint256 => Item) private listaItens;

    // Evento que sera disparado para retornar um item apos sua criacao
    event ItemCriado(
        uint256 indexed itemId,
        bool statusCedido,
        address indexed contratoCST,
        uint256 indexed tokenId,
        address cedente,
        address cessionario,
        uint256 expiraEm
    );

    // Construtor que inicializa o dono do contrato
    constructor(){
        _donoContrato = payable(msg.sender);
        taxaEnlace = 0.02 ether;
    }

    // Funcao que retorna a taxa cobrada pelo enlace
    function getTaxaEnlace() public view returns (uint256) {
        return taxaEnlace;
    }

    // Funcao que altera a taxa cobrada pelo enlace
    // (Apenas o dono do contrato pode utilizar esta funcao)
    function setTaxaEnlace(uint256 _novaTaxa) external onlyEnlace{
        taxaEnlace = _novaTaxa;
    }

    // Funcao que cria um novo item para ser cedido
    function criaItemCedivel(address contratoCST, uint256 tokenId, uint256 expiraEm) public payable{

        //require(preco > 0, "Preco deve ser ao menos de 1 wei!");
        require(msg.value == taxaEnlace, "Pague exatamente a taxa que o enlace exige!");
        uint256 itemId = _itemIds++;

        listaItens[itemId] = Item(
```

```
68              itemId,
69              false,
70              address(contratoCST),
71              tokenId,
72              payable(msg.sender),
73              address(0),
74              expiraEm
75          );
76
77          IERC721(contratoCST).transferFrom(msg.sender, address(this), tokenId);
78          _donoContrato.transfer(taxaEnlace);
79
80          emit ItemCriado(
81              itemId,
82              false,
83              contratoCST,
84              tokenId,
85              msg.sender,
86              address(0),
87              expiraEm
88          );
89      }
90
91      // Funcao que cede um item disponivel
92      function cederItem(address contratoCST, address cessionario, uint256 itemId) public{
93          uint256 tokenId = listaItens[itemId].tokenId;
94
95          require(!listaItens[itemId].statusCedido, "Este token ja foi cedidodo!");
96          require(IERC721(contratoCST).ownerOf(tokenId) == address(this), "Token nao disponivel!");
97          //require(msg.value == preco, "Pague exatamente o valor da cessao!");
98
99          //listaItens[itemId].cedente.transfer(msg.value);
100         IERC721(contratoCST).transferFrom(address(this), cessionario, tokenId);
101
102         listaItens[itemId].expiraEm = listaItens[itemId].expiraEm + block.timestamp;
103         listaItens[itemId].cessionario = cessionario;
104         listaItens[itemId].statusCedido = true;
105         _countItensCedidos++;
106     }
107
108     // Funcao que finaliza uma cessao apos o prazo de um item
109     // ou caso o cessionario desejar finalizar antes do prazo
110     function finalizaCessao(uint256 itemId) external {
111         Item storage itemCedido = listaItens[itemId];
112         require(itemCedido.statusCedido, "Este token nao esta cedido!");
113         //require(msg.sender == address(this) || block.timestamp >= itemCedido.expiraEm,
114                 // "Este token ainda esta no periodo de cessao!;
115         (bool sucessoTransferencia, ) = (itemCedido.contratoCST).call(
116             abi.encodeWithSignature(
117                 "transferirTokenExpirado(address,address,uint256)",
118                 itemCedido.cessionario,
119                 address(this),
120                 itemCedido.tokenId
121             )
122         );
123         require(sucessoTransferencia, "Nao foi possivel transferir o CST de volta para o Cedente!");
124         itemCedido.statusCedido = false;
125         _countItensDevolvidos++;
126         delete listaItens[itemId];
127     }
128
129     // Funcao que retorna a lista de CSTs disponiveis para cessao
130     function getCSTsCediveis() public view returns (Item[] memory) {
131         uint256 qtdeItensTotal = _itemIds;
132         uint256 qtdeItensCediveis = 0;
133         uint256 indiceAtual = 0;
134
135         // Contabiliza a quantidade de CSTs totais disponiveis
136         // para criar a lista de retorno
137         for (uint256 i = 1; i <= qtdeItensTotal; i++) {
```

```
138             if (listaItens[i].itemId != 0 && !listaItens[i].statusCedido) {
139                 qtdeItensCediveis += 1;
140             }
141         }
142 
143         Item[] memory listaItensDisponiveis = new Item[](qtdeItensCediveis);
144 
145         for (uint256 i = 1; i <= qtdeItensTotal; i++) {
146             if (listaItens[i].itemId != 0 && !listaItens[i].statusCedido) {
147                 listaItensDisponiveis[indiceAtual] = listaItens[i];
148                 indiceAtual += 1;
149             }
150         }
151         return listaItensDisponiveis;
152     }
153 
154     // Funcao que retorna os CSTs cediveis de um cedente
155     function getCSTsPorCedente() public view returns (Item[] memory) {
156         uint256 qtdeItensTotal = _itemIds;
157         uint256 countItensCedente = 0;
158         uint256 indiceAtual = 0;
159 
160         // Contabiliza a quantidade de CSTs totais do cedente
161         // para criar a lista de retorno
162         for (uint256 i = 1; i <= qtdeItensTotal; i++) {
163             if (listaItens[i].cedente == msg.sender) {
164                 countItensCedente += 1;
165             }
166         }
167 
168         Item[] memory listaItensCedente = new Item[](countItensCedente);
169 
170         for (uint256 i = 1; i <= qtdeItensTotal; i++) {
171             if (listaItens[i].cedente == msg.sender) {
172                 listaItensCedente[indiceAtual] = listaItens[i];
173                 indiceAtual += 1;
174             }
175         }
176         return listaItensCedente;
177     }
178 
179     // Funcao que retorna os CSTs cedidos por um locatario
180     function getCSTsPorCessionario() public view returns (Item[] memory) {
181         uint256 qtdeItensTotal = _itemIds;
182         uint256 countItensCessionario = 0;
183         uint256 indiceAtual = 0;
184 
185         // Contabiliza a quantidade de CSTs totais do locatario
186         // para criar a lista de retorno
187         for (uint256 i = 1; i <= qtdeItensTotal; i++) {
188             if (listaItens[i].cessionario == msg.sender) {
189                 countItensCessionario++;
190             }
191         }
192 
193         Item[] memory listaItensCessionario = new Item[](countItensCessionario);
194 
195         for (uint256 i = 1; i <= qtdeItensTotal; i++) {
196             if (listaItens[i].cessionario == msg.sender) {
197                 listaItensCessionario[indiceAtual] = listaItens[i];
198                 indiceAtual++;
199             }
200         }
201         return listaItensCessionario;
202     }
203 
204     // Funcao que retorna os itens expirados, mas ainda cedidos ate o momento atual da pesquisa
205     function getCSTsExpiradosECedidodos() public view returns (Item[] memory) {
206         uint256 qtdeItensTotal = _itemIds;
207         uint256 countItensExpiradosECedidos = 0;
```

```
208         uint256 indiceAtual = 0;
209
210         // Contabiliza a quantidade de CSTs totais expirados
211         // para criar a lista de retorno
212         for (uint256 i = 1; i <= qtdeItensTotal; i++) {
213             if (listaItens[i].statusCedido && listaItens[i].expiraEm <= block.timestamp) {
214                 countItensExpiradosECedidos++;
215             }
216         }
217
218         Item[] memory listaItensExpirados = new Item[](countItensExpiradosECedidos);
219
220         for (uint256 i = 1; i <= qtdeItensTotal; i++) {
221             if (listaItens[i].statusCedido && listaItens[i].expiraEm <= block.timestamp) {
222                 listaItensExpirados[indiceAtual] = listaItens[i];
223                 indiceAtual += 1;
224             }
225         }
226         return listaItensExpirados;
227         // Modificador utilizado para especificar quais funcoes apenas o dono do enlace pode chamar
228
229     }
230     modifier onlyEnlace() {
231         require(msg.sender == _donoContrato, "Apenas o enlace pode utilizar este metodo!");
232         _;
233     }
234 }
```

Listagem C.2: Funções para criar e retornar um Token.

C.1.2 Enlace mc

```
1
2  // SPDX-License-Identifier: MIT
3  pragma solidity ^0.8.27;
4
5  import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
6
7  contract EnlaceCrossChain is ERC721 {
8
9      // Proprietario do Notary
10     address public notaryOwner;
11
12     // Contador para o ID dos tokens do NFT e transferencias
13     uint256 private _tokenIds;
14     uint256 private _transferIds;
15     uint256 private _mintIds;
16
17     // Estrutura para armazenar informacoes da transferencia
18     struct NFTTransfer {
19         address nftContract;       // Endereco do contrato do NFT
20         address sender;            // Endereco do remetente
21         address receiverOnChain;   // Endereco que recebe na mesma rede (notaryOwner)
22         address receiverInterChain; // Endereco que recebera na outra rede
23         uint256 tokenId;           // ID do token NFT
24         uint256 transferId;        // ID unico da transferencia
25     }
26
27     struct MintRecord {
28         address sender;     // Quem solicitou o mint
29         address receiver;   // Destinatario do NFT mintado
30         uint256 tokenId;    // ID do token NFT
31         uint256 mintId;     // ID unico do mint
32     }
33
34     MintRecord[] public mintRecords;
35
```

```
36      // Lista de todas as transferencias de NFT
37      NFTTransfer[] public nftTransfers;
38
39      // Evento para registrar uma transferencia inter-chain
40      event NFTTransferInterChain(
41          address indexed nftContract,
42          address indexed sender,
43          address indexed receiverOnChain,
44          address receiverInterChain,
45          uint256 tokenId,
46          uint256 transferId
47      );
48
49      event NFTMinted(
50          address indexed sender,
51          address indexed receiver,
52          uint256 indexed tokenId,
53          uint256 mintId
54      );
55
56      // Evento para registro de um novo NFT mintado
57      // event NFTMinted(address indexed to, uint256 indexed tokenId);
58
59      // Construtor define o nome e simbolo do NFT e define o proprietario do contrato
60      constructor() ERC721("ENLACE", "ELC") {
61          //_donoContrato = payable(msg.sender);
62          notaryOwner = msg.sender;
63      }
64
65      // Funcao para receber e transferir um NFT, mantendo as informacoes e enviando para outra rede
66      function transferNFTInterChain(
67          address nftContract,
68          uint256 tokenId,
69          address receiverInterChain
70      ) external {
71          // Verifica se o contrato e compativel com o padrao ERC721
72
73
74          // Transfere o NFT da conta do remetente para o proprietario do contrato (notaryOwner)
75            IERC721(nftContract).transferFrom(msg.sender, address(this), tokenId);
76
77          // Incrementa o ID da transferencia
78          _transferIds++;
79
80          // Armazena as informacoes da transferencia
81          NFTTransfer memory newTransfer = NFTTransfer({
82              nftContract: nftContract,
83              sender: msg.sender,
84              receiverOnChain: notaryOwner,
85              receiverInterChain: receiverInterChain,
86              tokenId: tokenId,
87              transferId: _transferIds
88          });
89
90          nftTransfers.push(newTransfer);
91          (bool sucessoBurn, ) = (nftContract).call(
92              abi.encodeWithSignature(
93                  "burn(uint256)",
94                  tokenId
95              )
96          );
97          require(sucessoBurn, "Erro ao excluir o token!");
98
99          // Emite o evento de transferencia
100         emit NFTTransferInterChain(
101             nftContract,
102             msg.sender,
103             notaryOwner,
104             receiverInterChain,
105             tokenId,
```

```
106             _transferIds
107         );
108     }
109 
110     function mintNFT(address contratoCST, address to, address senderInterChain) public payable onlyEnlace {
111         _tokenIds++;
112         _mintIds++;
113         uint256 newItemId = _tokenIds;
114 
115         (bool sucessoMint, ) = (contratoCST).call(
116             abi.encodeWithSignature(
117                 "criarNovoToken()"
118             )
119         );
120         require(sucessoMint, "Erro ao criar o token!");
121 
122         IERC721(contratoCST).transferFrom(address(this), to, 0);
123 
124         //require(sucessoTransferencia, "Erro ao transferir o token!");
125         // Mintar o novo token para o endereco especificado
126         //_mint(to, newItemId);
127         // Armazenar as informacoes do mint
128         MintRecord memory newMintRecord = MintRecord({
129             sender: senderInterChain,
130             receiver: to,
131             tokenId: newItemId,
132             mintId: _mintIds
133         });
134 
135         mintRecords.push(newMintRecord);
136 
137         // Emite um evento de registro
138         emit NFTMinted(senderInterChain, to, newItemId, _mintIds);
139     }
140 
141     // Funcao para visualizar o proprietario do contrato (Notary Owner)
142     function getNotaryOwner() external view returns (address) {
143         return notaryOwner;
144     }
145 
146     // Funcao para obter as transferencias realizadas
147     function getTransferById(uint256 transferId) external view returns (NFTTransfer memory) {
148         require(transferId > 0 && transferId <= _transferIds, "Transfer ID invalido");
149         return nftTransfers[transferId - 1];
150     }
151 
152     function getNftTransfers() external view returns (uint256) {
153         return nftTransfers.length;
154     }
155         modifier onlyEnlace() {
156         require(msg.sender == notaryOwner, "Apenas o enlace pode utilizar este metodo!");
157         _;
158     }
159 }
```

Listagem C.3: Funções para criar e retornar um Token.

Apêndice D

Codificação dos Testes

Os testes apresentados na Listagem D.1 estão verificando a implantação bem-sucedida do contrato Token e sua funcionalidade para transferir tokens entre contas, verificar o saldo do token e atualizar a permissão. Primeiramente são importadas as dependências necessárias para o teste nas linhas de 1 a 4. Onde o pacote ***chai*** é usado para escrever asserções para testes de contrato. O pacote ***Ethers*** é usado para interagir com a blockchain Ethereum e implantar os contratos inteligentes. O script dos testes define um bloco ***describe*** para um grupo de testes relacionados ao contrato.

```
1  import { loadFixture } from "@nomicfoundation/hardhat-network-helpers";
2  import { expect } from "chai";
3  import { ethers } from "hardhat";
4  import '@nomicfoundation/hardhat-ethers';
5
6  describe("Deploy Contracts Enlace and CST", function () {
7
8    async function deployFixture() {
9      const [terceiroConfiavel, cedente, cessionario] = await ethers.getSigners();
10     const enderecoterceiroConfiavel = await terceiroConfiavel.getAddress();
11     const Enlace = await ethers.getContractFactory("Enlace");
12     const enlace = await Enlace.connect(terceiroConfiavel).deploy();
13     const CST = await ethers.getContractFactory("CST");
14     const cst = await CST.connect(cedente).deploy(enlace.getAddress());
15     const enderecoEnlace = await enlace.getAddress();
16     const enderecoCST = await cst.getAddress();
17     return { cst, enlace, terceiroConfiavel, cedente, cessionario, enderecoterceiroConfiavel, enderecoEnlace,
           enderecoCST };
18   }
19
20   it("Ciclo completo de cessao de token", async function () {
21     const { cst, enlace, enderecoCST, enderecoEnlace, cedente, cessionario} = await loadFixture(deployFixture);
22
23     await cst.connect(cedente).criarNovoToken();
24     expect(await cst.balanceOf(cedente)).to.equal(1);
25     expect(await cst.balanceOf(cessionario)).to.equal(0);
26     expect(await cst.balanceOf(enderecoEnlace)).to.equal(0);
27     console.log('Novo Token criado');
28     console.log('  Saldo de addr 1 :' ,await cst.balanceOf(cedente));
29     console.log('  Saldo de addr 2 :' ,await cst.balanceOf(cessionario));
30     console.log('  Saldo de enlace :' ,await cst.balanceOf(enlace));
31
32     await enlace.connect(cedente).criaItemCedivel(enderecoCST, 0, 120, {value: ethers.parseEther("0.02")});
33     console.log('Item 1 criado no Enlace');
34     expect(await cst.balanceOf(cedente)).to.equal(0);
35     expect(await cst.balanceOf(cessionario)).to.equal(0);
36     expect(await cst.balanceOf(enderecoEnlace)).to.equal(1);
37     console.log('  Saldo de addr 1 :' ,await cst.balanceOf(cedente));
38     console.log('  Saldo de addr 2 :' ,await cst.balanceOf(cessionario));
39     console.log('  Saldo de enlace :' ,await cst.balanceOf(enlace));
40
41     await enlace.connect(cedente).cederItem(enderecoCST,cessionario,0);
42     console.log('Item 1 cedido ao Cessionario');
```

```
    expect(await cst.balanceOf(cedente)).to.equal(0);
    expect(await cst.balanceOf(cessionario)).to.equal(1);
    expect(await cst.balanceOf(enderecoEnlace)).to.equal(0);
    console.log('  Saldo de addr 1 :' ,await cst.balanceOf(cedente));
    console.log('  Saldo de addr 2 :' ,await cst.balanceOf(cessionario));
    console.log('  Saldo de enlace :' ,await cst.balanceOf(enlace));

    await enlace.connect(cedente).finalizaCessao(0);
    console.log('Cessao finalizada');
    expect(await cst.balanceOf(cedente)).to.equal(0);
    expect(await cst.balanceOf(cessionario)).to.equal(0);
    expect(await cst.balanceOf(enderecoEnlace)).to.equal(1);
    console.log('  Saldo de addr 1 :',await cst.balanceOf(cedente));
    console.log('  Saldo de addr 2 :' ,await cst.balanceOf(cessionario));
    console.log('  Saldo de enlace :' ,await cst.balanceOf(enlace));
  });
});
```

Listagem D.1: Teste funcional do token de cessão

```
PS D:\GitHub\Period_Token> npx hardhat test
Compiled 2 Solidity files successfully (evm target: paris).


  Deploy Contracts Enlace and CST
Novo Token criado
  Saldo de addr 1 : 1n
  Saldo de addr 2 : 0n
  Saldo de enlace : 0n
Item 1 criado no Enlace
  Saldo de addr 1 : 0n
  Saldo de addr 2 : 0n
  Saldo de enlace : 1n
Item 1 cedido ao Cessionário
  Saldo de addr 1 : 0n
  Saldo de addr 2 : 1n
  Saldo de enlace : 0n
Cessão finalizada
  Saldo de addr 1 : 0n
  Saldo de addr 2 : 0n
  Saldo de enlace : 1n
    ✓ Ciclo completo de cessão de token (381ms)
```

Figura D.1: Retorno testes 2. Fonte: elaborado pelo autor

```
import { loadFixture } from "@nomicfoundation/hardhat-network-helpers"
import { expect } from "chai"
import { ethers } from "hardhat"
import '@nomicfoundation/hardhat-ethers'

describe("Deploy Contracts EnlaceCrossChain and CST", function () {

  async function deployFixture() {
    const [terceiroConfiavel, cedente, cessionario, cedente2, cessionario2] = await ethers.getSigners()
    //Deploy contratos REDE - 01
    const EnlaceCrossChain = await ethers.getContractFactory("EnlaceCrossChain")
    const enlaceCrossChain = await EnlaceCrossChain.connect(terceiroConfiavel).deploy()
    const CST = await ethers.getContractFactory("CST")
    const cst = await CST.connect(cedente).deploy(enlaceCrossChain.getAddress())
    //Deploy Contratos REDE - 02
    const EnlaceCrossChain2 = await ethers.getContractFactory("EnlaceCrossChain2")
    const enlaceCrossChain2 = await EnlaceCrossChain2.connect(terceiroConfiavel).deploy()
    const CST2 = await ethers.getContractFactory("CST2");
    const cst2 = await CST2.connect(cedente2).deploy(enlaceCrossChain2.getAddress())
    // Enderecos dos contratos
    const enderecoEnlace = await enlaceCrossChain.getAddress()
    const enderecoEnlace2 = await enlaceCrossChain2.getAddress()
    const enderecoCST = await cst.getAddress()
    const enderecoCST2 = await cst2.getAddress()

```

```
26      return { cst, enlaceCrossChain, cst2, enlaceCrossChain2, terceiroConfiavel, cedente, cessionario, cedente2,
            cessionario2, enderecoEnlace, enderecoCST, enderecoCST2 , enderecoEnlace2}
27    }
28    it("Interoperando CST da Rede-01 para Rede-02", async function () {
29      const { cst, enlaceCrossChain, cst2, enlaceCrossChain2, terceiroConfiavel, cedente, cessionario, cedente2,
            cessionario2, enderecoEnlace, enderecoCST, enderecoCST2 , enderecoEnlace2} = await loadFixture(
            deployFixture)
30
31      console.log('Criando novo CST');
32      await cst.connect(cedente).criarNovoToken()
33      expect(await cst.balanceOf(cedente)).to.equal(1)
34      expect(await cst.balanceOf(cessionario)).to.equal(0)
35      expect(await cst.balanceOf(enderecoEnlace)).to.equal(0)
36      expect(await cst.balanceOf(enlaceCrossChain)).equal(0)
37      console.log('  Saldo de cedente    :' ,await cst.balanceOf(cedente))
38      console.log('  Saldo de cessionario:' ,await cst.balanceOf(cessionario))
39      console.log('  Saldo de enlace     :' ,await cst.balanceOf(enlaceCrossChain))
40      console.log('  Saldo de Trust      :' ,await cst.balanceOf(terceiroConfiavel))
41
42      console.log('Transferindo CST para Enlace Rede-01');
43      await enlaceCrossChain.connect(cedente).transferNFTInterChain(enderecoCST, 0, cessionario)
44      expect(await cst.balanceOf(cedente)).to.equal(0)
45      expect(await cst.balanceOf(cessionario)).to.equal(0)
46      expect(await cst.balanceOf(enderecoEnlace)).to.equal(0)
47      expect(await cst.balanceOf(cedente2)).to.equal(0)
48      console.log('  Saldo de cedente    :' ,await cst.balanceOf(cedente))
49      console.log('  Saldo de cessionario:' ,await cst.balanceOf(cessionario))
50      console.log('  Saldo de enlace     :' ,await cst.balanceOf(enlaceCrossChain))
51      console.log('  Saldo de Trust      :' ,await cst.balanceOf(terceiroConfiavel))
52      console.log('CST com id=0 transferido para Cessionario2 Rede 2')
53      console.log('Burn CST com id=0 da Rede-01');
54      await enlaceCrossChain.connect(cedente).nftTransfers(0)
55      await enlaceCrossChain2.connect(terceiroConfiavel).mintNFT(cst2, cessionario2, cedente2)
56      expect(await cst2.balanceOf(cedente)).to.equal(0)
57      expect(await cst2.balanceOf(cessionario2)).to.equal(1)
58      expect(await cst2.balanceOf(enderecoEnlace2)).to.equal(0)
59      expect(await cst2.balanceOf(cedente2)).to.equal(0)
60      //  Imprimindo contas Rede 2
61      console.log('Token transferido para Cessionario Rede 2')
62      console.log('  Saldo de cedente 2    :' ,await cst2.balanceOf(cedente))
63      console.log('  Saldo de cessionario 2:' ,await cst2.balanceOf(cessionario2))
64      console.log('  Saldo de enlace 2     :' ,await cst2.balanceOf(enlaceCrossChain2))
65      console.log('  Saldo de Trust        :' ,await cst2.balanceOf(terceiroConfiavel))
66    })
67  })
```

Listagem D.2: Teste interoperabilidade do token de cessão - 02

```
PS D:\GitHub\Period_Token> npx hardhat test
Compiled 2 Solidity files successfully (evm target: paris).

  Testes Funcionais - Deploy contrato CST e Enlace
    ✓ Criar e aprovar token (83ms)

  Deploy Contracts EnlaceCrossChain and CST
Criando novo CST
  Saldo de cedente    : 1n
  Saldo de cessionario: 0n
  Saldo de enlace     : 0n
  Saldo de Trust      : 0n
Transferindo CST para Enlace Rede-01
  Saldo de cedente    : 0n
  Saldo de cessionario: 0n
  Saldo de enlace     : 0n
  Saldo de Trust      : 0n
CST com id=0 transferido para Cessionário2 Rede 2
Burn CST com id=0 da Rede-01
Token transferido para Cessionário Rede 2
  Saldo de cedente 2    : 0n
  Saldo de cessionario 2: 1n
  Saldo de enlace 2     : 0n
  Saldo de Trust        : 0n
    ✓ Interoperando CST da Rede-01 para Rede-02 (203ms)
```

Figura D.2: Retorno testes 1. Fonte: elaborado pelo autor

Solidity and Network Configuration				
Solidity: 0.8.28	Optim: false	Runs: 200	viaIR: false	Block: 30,000,000 gas
Methods				
Contracts / Methods	**Min**	**Max**	**Avg**	**# calls**	**usd (avg)**
CST					
approve	-	-	31,988	1	-
criarNovoToken	-	-	118,668	3	-
Enlace					
cederItem	-	-	118,987	1	-
criaItemCedivel	-	-	173,450	1	-
finalizaCessao	-	-	91,369	1	-
EnlaceCrossChain					
transferNFTInterChain	-	-	206,172	1	-
EnlaceCrossChain2					
mintNFT	-	-	272,209	1	-
Deployments				**% of limit**	
CST	-	-	2,056,850	6.9 %	-
CST2	-	-	2,056,922	6.9 %	-
Enlace	-	-	1,902,769	6.3 %	-
EnlaceCrossChain	-	-	2,869,582	9.6 %	-
EnlaceCrossChain2	-	-	2,869,618	9.6 %	-
Key					
{} Execution gas for this method does not include intrinsic gas overhead					
Δ Cost was non-zero but below the precision setting for the currency display (see options)					
Toolchain: hardhat					

Figura D.3: Consumo de gas testes. Fonte: elaborado pelo autor

Apêndice E

Amostra de Tabelas de Dados dos Experimentos

Tabela E.1: Amostra de dados do Experimento 1

	VM	Send Rate	Succ	Fail	Max Latency	Min Latency	Avg Latency	Throughput
0	Small	100	6001	0	0.250000	0.050000	0.090000	99.900000
1	Small	150	9001	0	0.210000	0.050000	0.100000	149.800000
2	Small	200	12001	0	0.280000	0.060000	0.110000	199.800000
3	Small	250	15001	0	0.330000	0.060000	0.120000	249.600000
4	Small	300	18001	0	0.660000	0.060000	0.150000	299.400000
5	medium	100	6001	0	0.250000	0.010000	0.080000	99.900000
6	medium	150	9001	0	0.220000	0.040000	0.090000	149.900000
7	medium	200	12001	0	0.200000	0.040000	0.090000	199.700000
8	medium	250	15001	0	0.220000	0.050000	0.100000	249.600000
9	medium	300	18001	0	0.340000	0.050000	0.110000	299.600000
10	xLarge	100	6001	0	0.080000	0.010000	0.060000	99.900000
11	xLarge	150	9001	0	0.120000	0.040000	0.060000	149.900000
12	xLarge	200	12001	0	0.140000	0.040000	0.060000	199.800000
13	xLarge	250	15001	0	0.130000	0.040000	0.070000	249.800000
14	xLarge	300	18001	0	0.170000	0.040000	0.070000	299.600000

Tabela E.2: Amostra de dados do Experimento 2

	VM	Send Rate	Succ	Fail	Max Latency	Min Latency	Avg Latency	Throughput	n
0	Small	300	18001	0	0.840000	0.140000	0.410000	298.600000	300
1	Small	300	18001	0	44.930000	0.090000	17.190000	172.800000	400
2	Small	300	4571	13430	85.020000	0.040000	11.130000	108.400000	500
3	Small	300	0	0	0.000000	0.000000	0.000000	0.000000	750
4	Small	300	0	0	0.000000	0.000000	0.000000	0.000000	1000
5	Small	300	0	0	0.000000	0.000000	0.000000	0.000000	1250
6	Small	300	0	0	0.000000	0.000000	0.000000	0.000000	1500
7	medium	300	18001	0	0.320000	0.020000	0.130000	299.500000	300
8	medium	300	18001	0	0.340000	0.090000	0.170000	299.400000	400
9	medium	300	18002	0	0.480000	0.030000	0.240000	299.000000	500
10	medium	300	10015	7874	79.030000	0.100000	17.210000	126.600000	750
11	medium	300	0	0	0.000000	0.000000	0.000000	0.000000	1000
12	medium	300	0	0	0.000000	0.000000	0.000000	0.000000	1250
13	medium	300	0	0	0.000000	0.000000	0.000000	0.000000	1500
14	xLarge	300	18001	0	0.260000	0.020000	0.120000	299.500000	300
15	xLarge	300	18001	0	0.220000	0.090000	0.150000	299.400000	400
16	xLarge	300	18001	0	0.270000	0.100000	0.190000	299.200000	500
17	xLarge	300	18001	0	0.580000	0.100000	0.390000	298.500000	750
18	xLarge	300	13770	0	7.480000	0.100000	5.420000	205.900000	1000
19	xLarge	300	11424	0	9.210000	0.110000	6.930000	167.200000	1250
20	xLarge	300	8251	1316	10.410000	0.100000	7.980000	138.000000	1500