Uma Abordagem Baseada em *Brokers* para Registro de Transações em Múltiplos Livros Razão Distribuídos

Mateus Pires[1], Daniel Souza[1], Rostand Costa[1], Guido Lemos[1]

[1]Laboratório de Aplicações de Vídeo Digital (LAVID)
Centro de Informática – Universidade Federal da Paraíba (UFPB)
Caixa Postal 58051-900 – João Pessoa – PB – Brasil

{mateus, daniel, rostand, guido}@lavid.ufpb.br

Abstract. *The use of distributed ledgers technologies (DLTs) to development of decentralized applications is an area of research that has recently gained prominence due to the popularization of such technologies. Through the use of DLTs, such as blockchains, it is possible to record data in order to guarantee two basic properties for this stored information: proof of existence from the moment of registration and the immutability of that record. Due to differences in technologies, blockchains differs from each others and require adaptations in the process of registering data when the goal is perform this registry in multiple ledgers. This work aims to present a broker-based approach to record data into multiple distributed ledgers avoiding issues related to particular characteristics of each technology.*

Resumo. *O uso de livros-razão distribuídos (Distributed Ledger Technologies - DLTs) no desenvolvimento de aplicações descentralizadas é uma área de pesquisa que tem ganhado destaque recentemente devido a popularização de tal tecnologia. Por meio do uso de DLTs, como por exemplo blockchains, é possível registrar dados de forma a garantir duas propriedades básicas para essa informação armazenada: a prova de existência dos dados a partir do momento do registro e a imutabilidade desse registro. As diferenças existentes entre as diversas tecnologias de blockchain que se apresentam atualmente demandam adaptações no processo de registro de dados quando se deseja lidar com persistência em vários livros-razão distribuídos ao mesmo tempo. Este trabalho tem como objetivo apresentar uma abordagem baseada em broker para registro de dados em múltiplos livros-razão distribuídos de forma transparente e integrada com o objetivo de abstrair as características diversas de cada tecnologia.*

1. Introdução

Com os avanços no desenvolvimento de tecnologias de livros-razão distribuídos ou DLTs (do inglês *Distributed Ledger Technologies*) foi possível a definição e uso de aplicações descentralizadas que incorporam tal metáfora. As DLTs permitem o registro organizado de informações diversas a depender da aplicação e semântica que seja dada a tais dados. Um caso especial de tecnologias dessa natureza são as *blockchains*, que

implementam a idéia de DLTs por meio de um mecanismo organizacional baseado em encadeamento de blocos.

Dessa forma, uma *blockchain* podem ser entendida como uma tecnologia de propósito geral que oferece uma base de dados altamente transparente, segura e resiliente contra falhas [Davidson et. al. 2016]. A propriedade de resiliência está ligada diretamente ao fato de que a *blockchain* está replicada em cada nó da rede que a processa. Além disso, o uso de poderosos mecanismos de consenso entre os computadores que compõe a rede garantem a integridade do livro razão e, consequentemente, atestam a confiabilidade da tecnologia. Duas das mais famosas *blockchains* em operação atualmente são a *Bitcoin* [Nakamoto 2008] e *Ethereum* [Wood 2014].

Blockchains são principalmente utilizadas na área financeira como base para grande parte das criptomoedas. A despeito de seu uso mais popular como tecnologia base para definição de criptomoedas, as *blockchains* podem ser utilizadas para aplicações diversas onde o registro de informação, a criação de plataformas de aplicações descentralizadas e a necessidade de prova de existência da informação são a base do modelo de negócio que se deseja explorar.

De fato, como apresentando em [Underwood 2016], *blockchains* têm sido estudadas e aplicadas dentre outros campos, na área financeira, comercial e desenvolvimento de políticas de transparência. McConaghy et. al. (2016) apresentam uma proposta para gestão de grandes bases de dados distribuídas incorporando características intrínsecas a *blockchains* à estes sistemas. Em [Huckle 2017] os autores analisam a aplicação combinada de *blockchain*, aplicações de economia compartilhada e Internet das Coisas (*Internet of Things - IOT*). Axon (2015) explora o uso de *blockchain* para apresentar um modelo de infraestrutura de chave pública baseado em tal tecnologia. Além disso, no trabalho desenvolvido por [Liang 2017] é proposto o uso de *blockchain* como uma base de dados para armazenamento de metadados relacionados a criação e operação de objetos de dados em nuvens, garantindo um carimbo de tempo para os registros, bem como a imutabilidade dos dados.

A vida útil dos dados armazenados em uma DLT está sensivelmente ligado a quantidade de nós, uma vez que cada nó possui uma cópia da cadeia. Dessa forma, uma boa alternativa a garantia de redundância das informações armazenadas seria sua persistência em diferentes DLT. Outra demanda para o uso de múltiplas DLTs refere-se a aplicações que tenham a necessidade de operar em um conjunto de livros-razões ao mesmo tempo por meio do intercâmbio de registros entre múltiplas redes. Um cenário onde isso deverá ser comum está relacionado a comunicação entre DLTs federadas.

Devido a heterogeneidade entre diferentes DLTs, o armazenamento de informação em redes diversas exige a compreensão do funcionamento de cada tecnologia, bem como a especificação de elementos restritivos, como por exemplo espaço para salvamento de dados e tempo de confirmação de operações. Nesse sentido, esse trabalho tem como objetivo apresentar uma abordagem para registro de informações em múltiplos livros-razão distribuídos. Para tanto, os autores propõe uma estratégia baseada em *Broker* para acesso unificado e transparente às DLTs, bem como mecanismos para tomada de decisão automatizada do ponto de vista de escolha de qual cadeia, ou conjunto delas, deverá conter as informações armazenadas.

O restante do documento está organizado como segue. A Seção 2 apresenta o conceito de livro razão distribuído e *blockchain*. Além disso, a Seção 2 faz uma breve discussão sobre o processo de registro de dados em DLTs e os desafios inerentes ao manuseio de múltiplas DLTs. Na Seção 3 é apresentada uma estratégia para gestão de registros em múltiplas DLTs. Na Seção 4 é apresentado um estudo de caso utilizado para validar a solução proposta. Por fim, na Seção 5 são apresentadas as considerações finais e trabalhos futuros.

2. DLTs, *Blockchains* e Registro de Dados

DLTs podem ser vistos como uma base de dados distribuída ao longo de uma rede de computadores. Cada computador é um nó da rede e possui uma cópia idêntica da base de dados que é semanticamente definida no formato de um livro razão.

Uma das principais características de uma DLT é a propriedade de descentralização, ou seja, a base de dados distribuída não é mantida por nenhuma autoridade central. A inserção de novos registros no livro é feita de forma independente por cada nó. Por meio de um processo de votação os nós determinam qual versão do livro é a válida por meio de algoritmos de consenso. Uma vez determinada a nova versão do livro razão, as informações são atualizadas para todos os nós.

Todo o processo de consenso e manutenção de uma cópia fiel do livro razão pelos nós é feito de forma automatizada pela rede de computadores. O uso de DLTs reduz os custos com a manutenção de um estrutura que garante a integridade de registros sem a necessidade de autoridades centrais como bancos e governos [Mills et. al. 2016].

Uma *blockchain*, por sua vez, é um tipo de DLT onde os dados armazenados são agrupados e organizados na forma de blocos. Esta tecnologia tem como estrutura básica uma cadeia de blocos gerados linearmente e cronologicamente [Davidson et. al. 2016]. Esses blocos estão ligados de forma que cada um possui uma referência para o bloco anterior. Essa referência é o *hash* desse bloco e serve como prova de que nenhum dos dois foram adulterados ou corrompidos (ver Figura 1) [Nakamoto 2008].

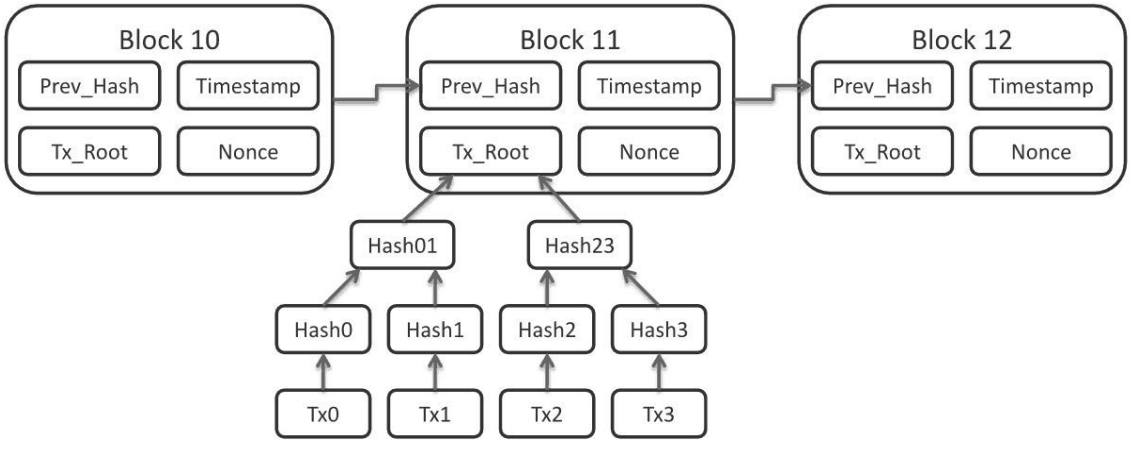

Figura 1. Representação de um conjunto de blocos interligados pelas referências de *hash* do bloco anterior. Internamente, cada bloco contém um hash raiz resultado da combinação dos recibos das transações.

Dentro de um bloco é armazenado também uma hash raiz (*Tx_Root*) fruto da combinação dois a dois dos IDs das transações realizadas no contexto daquele bloco. Para tanto, essa informação é condensada por meio do uso de uma estrutura de dados denominada *Árvore de Merkle* [Merkle 1987].

A prova de integridade de toda a cadeia está na validação de todos os pares de blocos dela e na concordância da rede em sua relação de blocos. A cadeia só pode ser considerada como verdadeira se ela é aceita de forma predominante na rede. Sendo assim, teoricamente, uma cadeia que não é acordada por mais de 50% dos nós da rede é considerada um *fork* e não faz parte da *blockchain* original.

Um exemplo de uso de *blockchain*, que é também uma de suas primeiras aplicações, é a criptomoeda *Bitcoin*. Ela é atualmente a maior rede *blockchain* existente, e isso traz uma grande disponibilidade da rede para realização de operações, segurança quanto à concordância dos blocos e, por consequência, resiliência a ataques e a corrupção de dados.

Transações na *Bitcoin* são realizadas por um sistema de *script*. Cada instrução é rotulada por um *opcode* e um campo especial denominado *OP_RETURN* pode ser utilizado para armazenamento de dados adicionais. Este campo é o único que permite armazenamento arbitrário de bytes e obedece o limite estabelecido de 80 *bytes*. Nesse sentido, caso se deseje utilizar a *Bitcoin* para aplicações que envolvem o registro de dados, deve-se observar essa restrição de espaço definida por sua estrutura.

A segunda rede *blockchain* mais conhecida, a *Ethereum*, tem algumas funcionalidades diferenciadas em comparação com a *Bitcoin*. De fato, a *Ethereum* é uma plataforma de aplicações, não uma criptomoeda unicamente. Para tanto, a *Ethereum* possui um mecanismo denominado *Contratos Inteligentes* (*Smart Contracts*) [Luu et. al. 2016] que possibilita a definição de acordos entre atores (pessoas, sistemas, etc) de uma negociação. Esses contratos são escritos em código-fonte e são avaliados e executados por meio da referida *blockchain*.

Assim como na *Bitcoin*, é possível armazenar dados na *blockchain* Ethereum. O armazenamento na *Ethereum* também é feito por meio da criação de uma transação. Para isso, os dados que se desejam persistir são colocados no atributo *data*. A quantidade máxima de dados que podem ser armazenados não é estabelecida, mas é limitada por outros fatores, como o preço a que uma transação poderia chegar, uma vez que o cálculo desse preço está diretamente relacionado ao número de bytes armazenados em uma transação. De fato, o preço pago por uma transação, dado a quantidade de dados que se deseje colocar, pode inviabilizar aplicações que necessitem de persistir muitos dados.

Existe ainda esforços da academia e indústria no desenvolvimento de outros tipos de DLTs com foco no suporte à aplicações nas áreas industrial, logística, financeira, de tecnologia da informação, dentre outras. Um exemplo de esforço dessa natureza é o projeto *Hyperledger* [Cachin 2016] fomentado por um consórcio de empresas e capitaneado pela *Linux Foundation*[1]. O objetivo do projeto, além de construir uma nova DLT para interoperabilidade entre instituições, é oferecer um

[1] http://www.linuxfoundation.org/

conjunto de ferramentas que promovam o acesso a essa nova rede, bem como facilite a criação de aplicações. Na mesma linha, outro projeto de DLT que tem ganho destaque é o R3 [Brown 2016]. A proposta da DLT é gerenciar e sincronizar acordos financeiros entre instituições reguladas. Essa DLT suporta um conjunto de algoritmos de consenso e não possui uma criptomoeda associada.

Independente das funcionalidades incorporadas por tecnologias mais modernas (a exemplo de *Contratos Inteligentes*), qualquer DLT permite o registro de operações entre pares de atores. Tais operações são armazenadas na forma de transações no livro razão. Cada ator participante de uma transação é de fato representado por um endereço. As transações ocorrem entre endereços. Uma transação pode ser constituída por um ou mais endereços, sendo que sempre temos um endereço base e um ou mais endereços alvo.

Transações são constituídas por um conjunto de propriedades que identificam a operação corrente. Dentre tais propriedades tem-se a referência temporal que pode ser vista como um carimbo de tempo que prova a existência da operação a partir do momento de sua criação. Além disso, como já mencionado,as DLTs podem oferecer suporte a inclusão de dados adicionais no registro.

Uma das características mais importantes no tocante ao processo de se registrar dados em uma DLT é a prova de existência. Tal característica está associada a capacidade de armazenar dados que façam uma referência inegável a um conteúdo digital e assim ter uma prova de que esse conteúdo existia no momento em que a transação foi realizada.

Um exemplo disto seria a prova de existência de um documento digital. Por exemplo, utilizando-se uma função criptográfica de *hash*, poderíamos obter uma referência única para um documento que poderá ser armazenada, por meio de uma transação, em uma dada DLT e, assim, estabelecer um registro que prove que aquele documento existia pelo menos a partir daquele momento.

Uma vez realizado o registro e, dada as características de imutabilidade do razão distribuído, temos algumas propriedades para esse registro, quais sejam: os dados de registro armazenados estarão disponíveis enquanto houver nós na rede e, além disso, como essas informações são imutáveis, elas podem servir como um carimbo que prova a existência desse dado.

Atualmente, um nó da *Bitcoin* e da *Ethereum* precisa de cerca de 190 GB e 280 GB, respectivamente, para armazenar todos os blocos da cadeia. O alto volume se deve a grande quantidade de blocos, transações, e o longo tempo de operação das duas. A *Bitcoin*, apesar de ser mais antiga, tem um volume menor que a *Ethereum*. Isso se deve ao fato da *Bitcoin* permitir a adição de uma quantidade limitada de dados adicionais no registro de transação. A *Ethereum*, além de permitir armazenar grandes quantidades de dados, também suporta contratos inteligentes, que demandam uma quantidade maior de bytes para registro.

Pelo seu tamanho excessivo, ter uma cópia local de um nó para realizar operações na *blockchain* pode ser inviável ou inconveniente. Sem um nó, contudo, ainda é possível ter acesso e comunicação com a *blockchain* utilizando serviços remotos

que disponibilizam APIs, como a *Insight*[2] para *Bitcoin* e *Etherscan*[3] para *Ethereum*. Esses serviços permitem recuperar informações sobre blocos e transações, além de facilitarem a realização destas transações.

3. Um Modelo para Registro em Múltiplos Razões Distribuídos

Diferentes DLTs possuem finalidades diferentes e, consequentemente, formas distintas de operar, além de restrições específicas. Uma aplicação que necessita armazenar dados em múltiplas DLTs deve tratar esses casos, preferencialmente, de forma que eles não fiquem expostos ao usuário.

Algumas das diferenças e restrições estão listadas abaixo:

- Informações necessárias para realizar uma transação;
- Tamanho do espaço de armazenamento;
- *Contratos Inteligentes*.

Um exemplo de comportamento diferente é a forma como transações são geridas por diferentes DLTs:

- Uma transação em *Bitcoin*, por exemplo, precisa dos recursos do endereço, chamados de *unspent outputs*, que são as saídas de transações que ainda não foram utilizadas como entradas. Uma transação do endereço **A** para o endereço **B** é na verdade uma transformação das entradas do endereço **A** em duas saídas para os endereços **A** e **B**. O que **B** irá receber é uma saída com os recursos especificados para ele na transação. O que **A** irá receber é uma saída com o restante dos recursos enviados. Estas saídas podem então ser utilizadas como entradas para outras transações [Nakamoto 2008];
- *Ethereum*, por sua vez, implementa endereços que mantêm estados com informações sobre seus recursos, de forma que não necessita de informação similar dentro da transação [Buterim 2013].

Da mesma forma que um dado pode ser armazenado em uma transação comum na rede *Ethereum*, ela também pode ser armazenada utilizando um *Contrato Inteligente* que daria automaticamente outras características ao registro dos dados. A utilização de *Contratos Inteligentes* desta forma é mais um ponto de desencontro entre registros de dados em diferentes DLTs e até mesmo dentro da própria *Ethereum*, visto que há múltiplas formas de registro nela.

Além disso, uma aplicação que necessita lidar com diferentes *blockchains*, carrega o problema de ter que manter nós que podem não apenas ocupar muito espaço, como também precisam estar em constante sincronia com a rede. Conforme discutido na seção 2, utilizar serviços remotos é uma solução, mas também deixa a aplicação sob dependência da disponibilidade deles. Em resumo, a forma que será definida para acesso ao livro razão deve levar em consideração dois fatores: custo de manter um nó local e disponibilidade dos serviços que permitem o acesso remoto a tais nós.

Dada a heterogeneidade dos padrões, aplicações que desejem se comunicar com mais de uma DLT, devem implementar as especificidades de comunicação e gestão de

[2] https://insight.is/
[3] https://etherscan.io/

dados de cada padrão. Nesse sentido, como solução para lidar com a problemática relacionada à esta heterogeneidade inerente à diferentes DLTs, idealizamos um modelo de comunicação unificado baseado na ideia de um *Broker*, que funcionará como um provedor de acesso às DLTs, simplificando o processo e abstraindo detalhes e complexidade das aplicações usuárias.

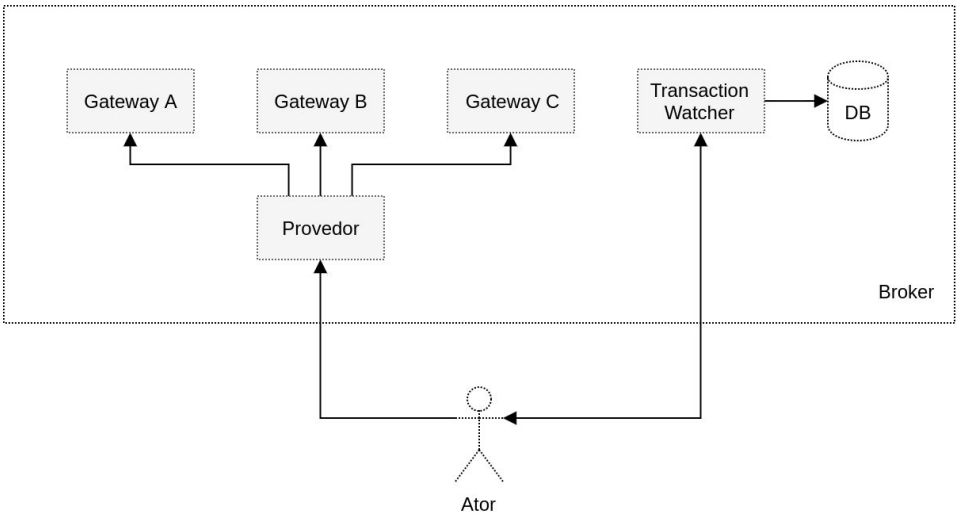

Figura 2. Arquitetura do *Broker*.

A arquitetura do modelo proposto pode ser visualizada na Figura 2. A partir de um ponto de acesso unificado denominado *Broker*, as DLTs podem ser acessadas por meio de um identificador. A forma que se faz esse acesso é por meio de adaptadores, chamados de *gateways*, com uma interface para acesso unificado. Adicionalmente, o *Broker* possui um observador de transações, o *Transaction Watcher*, para indicar que de fato o registro foi confirmado como incluso em uma determinada rede, após a definição do algoritmo de consenso. Na prática, para DLTs do tipo *blockchain* a implementação do *Transaction Watcher* de fato verifica o número de confirmações de um bloco na rede para indicar o sucesso do processo de registro.

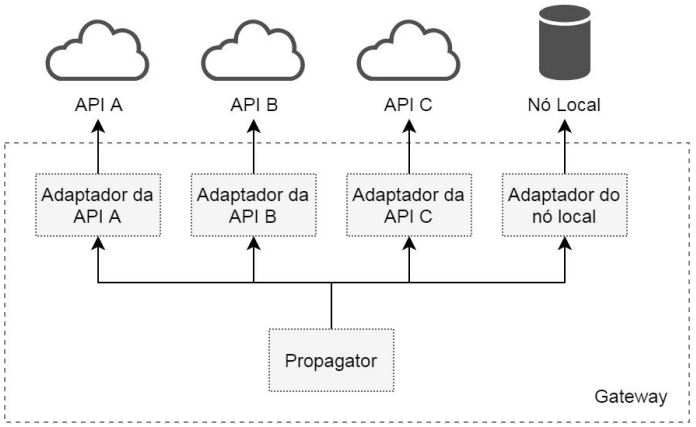

Figura 3. Arquitetura da Gateway. O propagador está representado pelo bloco com nome *Propagator* e cada nó pelas caixas ao lado de Adapters.

Um *gateway* específico para uma DLT (Figura 3) deve implementar a interface única com no mínimo três métodos: registrar, recuperar dados de transações e recuperar informações da DLT, como exemplificado no Quadro 1. É necessário também um

método para recuperar informações da DLT, uma vez que o *Transaction Watcher* utiliza estas informações para gestão de que transações foram ou não foram confirmadas.

Em um nível mais baixo, um *gateway* possui um propagador de requisições para APIs de acesso a DLT específica ou para acesso direto a um eventual nó dessa DLT. Para facilitar o entendimento, tanto um nó local quanto um serviço de API remota serão chamados nesse texto de nó.

```
interface Gateway {
       register(data);
       retrieve(transactionId);
       getInfo();
}
```

Quadro 1. Interface da Gateway.

O *propagador* é responsável por realizar o gerenciamento dos nós de acesso a DLT e tratar os casos de indisponibilidade que ocorrerem. O propagador decide qual nó ele usará para responder a requisição. Tanto o *Propagador*, quanto os *Adapters* para os nós devem implementar a mesma interface.

Essa interface depende do tipo de DLT que está sendo integrada ao *Broker*, de tal forma que seus métodos são dependentes da tecnologia empregada. No Quadro 2 é apresentado um exemplo de interface desenvolvida para gestão de transações utilizando a *blockchain Bitcoin*. O método *getUnspent* requisita os *unspent*. Estes são utilizados como entradas para criação da transação, que é enviada para a rede utilizando o método *broadcast*, obtendo ao final o identificador da transação. O identificador pode ser utilizado para requisitar os dados da transação com o método *getTransaction* ou para servir de método auxiliar, em conjunto com *getInfo*, para fornecer os dados de transação que serão solicitados pelo *Transaction Watcher* por meio da interface de acesso unificado do *gateway*.

```
interface BitcoinNode {
       getUnspent(address);
       broadcast(transaction);
       getTransaction(transactionId);
       getInfo();
}
```

Quadro 2. Interface para os nós e para o *propagador* do *gateway* para *Bitcoin*.

O propagador *m*antém os nós para acesso a DLT em uma fila, o primeiro nó é sempre o que será utilizado na próxima requisição, e quando este está indisponível, é passado para o final da fila e a requisição é automaticamente passada para o novo nó do topo. Pode ser definido o número de tentativas que se fará na fila inteira. Então se todos os nós falharem, a fila pode ser percorrida novamente.

O *gateway* retornada pelo *Broker* pode ser utilizada da mesma forma independente da DLT escolhida, uma vez que possui uma interface de acesso única. Quando um registro é feito, ele retorna a identificação da transação realizada e o identificador da DLT. Essa informação pode ser utilizada para recuperar os dados utilizando seu método de busca específico.

Já a informação retornada pela recuperação é constituída pelos dados registrados, o número de confirmações e a data do registro. No caso da data, pode haver diferenças no retorno dependendo do momento da consulta. Caso a consulta seja feita antes de qualquer confirmação, a data será a mesma da requisição de registro, caso a consulta seja feita após a confirmação de fato do registro, a data retornada será correspondente ao momento de criação do bloco em que a transação foi registrada.

Com os dados iniciais de registro retornados pelo *gateway*, procede-se com o registro da transação para ser observada pelo *Transaction Watcher*, que consulta os novos blocos de cada DLT que tem transações em espera e, quando há confirmação em alguma, é chamado um método de *callback* informando os seus dados. Este método de *callback* não é específico de cada transação, ele é um método único definido no *Broker*.

O reconhecimento de qual transação está sendo confirmada deve ser feito pelo usuário do serviço utilizando os identificadores da DLT e da transação. Isso possibilita maior segurança sobre quais transações estão esperando confirmação, pois todas são mantidas em um banco de dados até que sejam concluídas. Se o serviço que utiliza o *Broker* for desligado durante o processo, ele será retomado após a definição do método de *callback*. Para isso, as transações que já estão no banco de dados serão consultadas.

Para o caso de DLTs do tipo *blockchain*, as etapas executadas pelo *Transaction Watcher* estão listadas abaixo:

1. Receber método de *callback*;
2. Consultar transações no banco de dados;
    a. Se não houver transações, esperar por uma transação antes de continuar.
3. Requisitar o número do bloco atual e o intervalo de geração de blocos utilizando a *Gateway* que corresponde ao identificador da DLT da transação;
4. Calcular tempo previsto para geração do bloco com margem de segurança de 1 segundo;
5. Esperar a geração do próximo bloco pelo tempo calculado;
6. Requisitar o número do bloco atual e verificar se ele é maior que o último recebido;
    a. Se não for maior, esperar um tempo pré-definido e repetir o passo 6;
7. Requisitar os dados de todas as transações pendentes no banco de dados e verificar se estão confirmadas;
8. Chamar o método de *callback* passando as transações confirmadas e removê-las do banco de dados;
9. Voltar para o passo 2.

O passo 5, dependendo da *blockchain*, pode demorar até no máximo o tempo de intervalo de geração de novo bloco se a transação for confirmada no primeiro bloco, ou até múltiplos intervalos se confirmada nos próximos. Por exemplo, para a *blockchain*

Bitcoin, o intervalo de geração está em torno de 8.48 minutos[4], enquanto que para a *Ethereum*, o intervalo está em torno de 14 segundos[5].

Vale salientar que cada *gateway* precisa ser configurada para ter um endereço e chave privada para realizar as transações. Quanto a isso, o *Broker* pode ser implementado de duas formas: uma em que ele possui as chaves privadas e outra em que ele recebe as chaves durante sua inicialização. Esta última diminui os impactos das alterações, caso seja necessário substituí-las, por exemplo.

Embora o acesso aos *gateways* específicos possa ser feito por meio da indicação direta de uma DLT, para o modelo proposto foi pensado na possibilidade de uso de heurísticas para escolha da DLT a ser utilizada. O acesso por indicação direta de uma DLT demanda conhecimento prévio por parte do usuário do suporte a esta DLT no *Broker*. A comunicação por meio de um mecanismo de heurística oferece uma forma mais segura de acesso, visto que delega a escolha da DLT a ser utilizada para o próprio serviço. Outra característica interessante dessa abordagem é que o processo de escolha saí de uma visão meramente técnica para uma visão que captura requisitos não-funcionais de uma possível aplicação. Dois exemplos de requisitos que podem ser capturados por heurísticas são velocidade e custo de registro. Nesta linha, duas estratégias básicas foram definidas como heurísticas preliminares.

A primeira, denominada *FASTEST*, tenta escolher para registro a DLT com menor tempo para confirmação de atualização de livro razão (no caso de *blockchains*, confirmação de bloco). Para essa estratégia são necessárias informações como intervalo de geração de blocos e tempo previsto para a geração do próximo bloco de todas as DLT disponíveis. A segunda estratégia, denominada *CHEAPEST*, tenta escolher a DLT com menor custo de transação. Para tanto, é preciso analisar as últimas transações, definindo uma média para taxa de transação, verificando o preço médio e escolhendo a DLT com menor taxa.

Além das estratégias definidas e da indicação direta, é possível ainda indicar um conjunto de DLTs de forma que o registro seja feito em mais de uma. A forma como o modelo foi construído permite também a adição de novas heurísticas em função dos requisitos não-funcionais de aplicações que venham a utilizar essa dinâmica de registro em múltiplos livros-razão.

4. Estudo de Caso: Registro de Documentos Digitais

Como estudo de caso, foi desenvolvido um serviço de prova de existência de documentos digitais com suporte a múltiplas *blockchains* utilizando APIs públicas para realizar todas as operações relativas às *blockchains*.

O serviço, inicialmente, implementou o suporte a *Bitcoin* e *Ethereum*, ficando a critério do usuário em qual é realizado o registro. As APIs manipuladas para *Bitcoin*

[4] https://blockchain.info
[5] https://etherscan.io/

foram a *Insight*[6], *BlockExplorer*[7] e *BlockCypher*[8] e para *Ethereum* foi apenas a *Etherscan*.

Uma visão da forma como foi estruturado o estudo de caso é apresentado na Figura 4. O serviço responsável pelo registro e recuperação de dados e o *Broker* foram implementados utilizando as tecnologias *NodeJS* e *RabbitMQ* com o suporte de diversas bibliotecas e um módulo para geração de assinatura dos documentos digitais no formato PDF. O serviço disponibiliza uma API HTTP para consulta de transações e duas filas para o registro de documentos, uma de entrada e uma de saída. Esses meios são consumidos por uma aplicação Web que serve como interface para o usuário. O diagrama de fluxo de dados do estudo de caso está ilustrado na Figura 4.

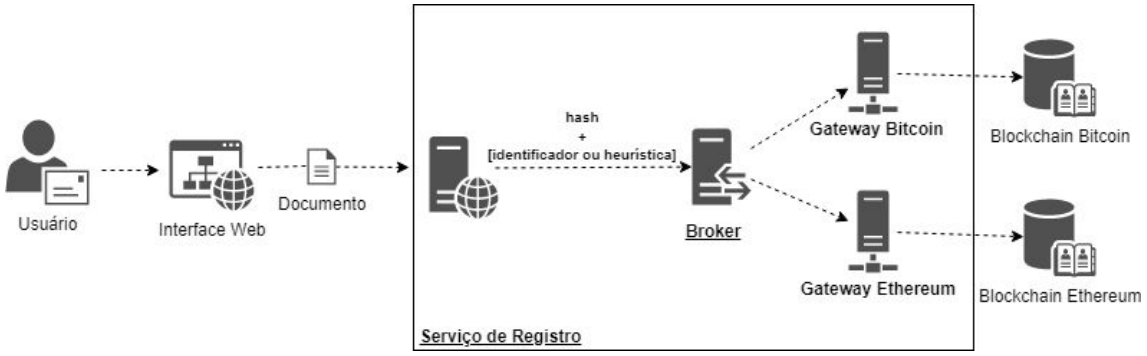

Figura 4. Diagrama estrutural do estudo de caso desenvolvido para validação do modelo proposto.

O *Broker* é um módulo independente que é ativado pelo serviço de registro. Ele se resume a métodos que realizam as seguintes tarefas:

- Acessar a *gateways* por identificador;
- Acessar a *gateways* por heurística;
- Definir o método de *callback* do *Transaction Watcher*;
- Cadastrar uma transação no *Transaction Watcher*.

Com o uso do *Transaction Watcher*, o serviço de registro é informado à respeito de transações confirmada de forma automatizada, por meio de um mecanismo de notificação que utiliza uma função *callback*.

Em resumo, o processo de registro tem as seguintes etapas:

- A partir do *Broker*, cadastrar uma função de callback no *Transaction Watcher*;
  - Inicialmente, o *Transaction Watcher* não inicializará seu processo, pois não há transações pendentes no banco de dados.
- Abrir a fila de entrada e continuamente esperar por dados;
  - Receber identificador da DLT, ou heurística, e documento PDF pela fila;
  - Gerar a *hash* do documento;;

[6] https://insight.is/
[7] https://blockexplorer.com/
[8] https://www.blockcypher.com/

- A partir do *Broker*, selecionar e acessar o *gateway* adequado utilizando o identificador ou heurística;
- Fazer o registro da *hash* utilizando o *gateway* selecionado e receber o identificador da transação;
  - Esta etapa, se feita utilizando a *Bitcoin*, terá 3 nós em alguma ordem inicial. O *propagador* escolherá o primeiro para fazer as requisições necessárias e, em caso de falha, colocará ele no final da fila, repetindo a requisição com o próximo. Se todos falharem, o processo será reiniciado no máximo mais uma vez, em decorrência do número de tentativas do propagador estar estabelecido como duas;
  - No caso da *Ethereum*, atualmente há apenas um único nó, então há um número configurável de tentativas para o mesmo (nos testes iniciais, esse valor foi setado para 2).
- A partir do *Broker*, cadastrar o identificador da transação no *Transaction Watcher*;
  - O *Transaction Watcher* inicializará seu processo de consultas que continuará enquanto houver transações pendentes.
- Esperar chamada do *callback* cadastrado com os dados da transação confirmada e enviar para a fila de saída.

Para fazer a confirmação da prova de existência, é necessário recuperar os dados da transação, este processo tem apenas algumas etapas:

- Receber requisição de recuperação de dados com os identificadores da DLT e da transação;
- A partir do *Broker*, selecionar e acessar o *gateway* adequado utilizando o identificador;
- Requisitar os dados utilizando o *gateway* selecionado e responder o retorno ao usuário.

O *hash* retornado no final deste último processo pode ser utilizada para comparar com o *hash* do documento que está sendo verificado, assim obtendo a prova de existência a partir da data que também está na resposta, caso sejam iguais.

5. Conclusões

Este trabalho apresentou uma proposta de arquitetura baseada no conceito de broker para viabilizar a comunicação com múltiplas DLTs de forma unificada. A comunicação do Broker com as DLTs é feita por meio de *gateways* que possuem uma interface, implementada para cada DLT que se deseje integrar ao sistema. Além disso, um mecanismo de notificação de registro de transações também foi pensado. Esse mecanismo possibilita o acesso unificado a informações sobre confirmação de inclusão de novos registros nas DLTs que se estão sendo geridas.

Um outro aspecto relevante da solução proposta, foi a definição de um modelo abstrato de acesso por meio de heurísticas que facilitam a escolha de uma determinada DLT para registro. Por meio do uso de heurísticas a decisão de escolha da DLT deixa de ser baseada no conhecimento técnico das tecnologias e passa a se basear em requisitos que por ventura estejam alinhadas com uma determinada aplicação que venha a utilizar

o serviço. Requisitos como velocidade ou custo são mais facilmente capturados por meio dessa abordagem. O modelo é flexível o bastante para permitir a inclusão de novas heurísticas à medida que aplicações demandem novos requisitos associados a escolha de uma DLT em detrimento a outra.

Por fim, com o objetivo de validar o modelo proposto, uma aplicação foi construída como estudo de caso. Por meio de um sistema para registro de documentos digitais, avaliamos a arquitetura proposta. Neste estudo de caso duas *blockchains* (Bitcoin e Ethereum) foram utilizadas.

O uso de filas assíncronas e de uma base de dados para armazenamento de transações ainda não confirmadas pelo *Transaction Watcher* adicionaram um componente de resiliência a falhas a arquitetura. Adicionalmente, esses dois componentes tornam o modelo proposto escalável, de forma a flexibilizar seu uso em ambientes com alta demanda de acesso e possibilidade de escalonamento de novas instâncias do serviço.

Como trabalho futuro, pretende-se avaliar o modelo proposto integrando um conjunto maior de DLTs. Nesse sentido, a ideia é buscar tecnologias que possuem implementações alternativas ao uso de cadeias de blocos, e/ou ainda, *blockchains* com uma proposta diferente das já integradas ao sistema, como por exemplo, a *HyperLedger* e a R3. Outra questão que demanda uma maior investigação é a especificação de um número maior de heurísticas, bem como a validação dessas heurísticas com um conjunto mais robusto de DLTs. Além disso, uma diversidade maior de estudos de caso podem permitir um refinamento do modelo e um processo de validação mais completo.

Agradecimentos

Os autores gostariam de agradecer a RNP (Rede Nacional de Pesquisa) pelo financiamento deste trabalho através do programa de Grupos de Trabalho (GTs).

Referências

Axon, L. (2015). Privacy-awareness in Blockchain-based PKI.

Buterin, V. (2013). Ethereum white paper. *GitHub repository*.

Brown, R. G. (2016). Introducing R3 Corda: A Distributed Ledger for Financial Services. *R3, April*, *5*.

Cachin, C. (2016, July). Architecture of the Hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*.

Davidson, S., De Filippi, P., & Potts, J. (2016). Economics of blockchain.

Davidson, S., De Filippi, P., & Potts, J. (2016). Disrupting governance: The new institutional economics of distributed ledger technology.

Huckle, S., Bhattacharya, R., White, M., & Beloff, N. (2016). Internet of things, blockchain and shared economy applications. *Procedia Computer Science*, *98*, 461-466.

Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K., & Njilla, L. (2017, May). Provchain: A blockchain-based data provenance architecture in cloud environment

with enhanced privacy and availability. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (pp. 468-477). IEEE Press.

Luu, L., Chu, D. H., Olickel, H., Saxena, P., & Hobor, A. (2016, October). Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 254-269). ACM.

McConaghy, T., Marques, R., Müller, A., De Jonghe, D., McConaghy, T., McMullen, G., Granzotto, A. (2016). BigchainDB: a scalable blockchain database. *white paper, BigChainDB*.

Merkle, R. C. (1987, August). A digital signature based on a conventional encryption function. In *Conference on the Theory and Application of Cryptographic Techniques* (pp. 369-378). Springer, Berlin, Heidelberg.

Mills, D. C., Wang, K., Malone, B., Ravi, A., Marquardt, J. C., Badev, A. I., ... & Ellithorpe, M. (2016). Distributed ledger technology in payments, clearing, and settlement.

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

Underwood, S. (2016). Blockchain beyond bitcoin. *Communications of the ACM*, *59*(11), 15-17.

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, *151*, 1-32.