Demochain - Framework destinado a criação de redes blockchain híbridas para dispositivos IoT

Lorenzo W. Freitas[1], Carlos Oberdan Rolim[2]

[1]Departamento de Engenharias e Ciência da Computação
Universidade Regional Integrada do Alto Uruguai e das Missões – Santo Ângelo, RS – Brazil

lorenzofreitas@aluno.santoangelo.uri.br, ober@san.uri.br

Resumo. *O uso da tecnologia Blockchain no contexto da IoT (Internet of Things) está sendo cada vez mais explorado pela comunidade acadêmica e a indústria. No entanto, essa implantação pode ser custosa ou inviável pois a Blockchain pode exigir recursos computacionais que não são obtidos facilmente com o uso de dispositivos IoT. Assim, esse trabalho apresenta o framework Demochain cuja função é auxiliar no desenvolvimento de plataformas blockchains híbridas no contexto de IoT.*

1. Introdução

Uma importante área de pesquisa na Computação que está apresentando um rápido crescimento é a Internet das Coisas (IoT). Segundo Gartner (2017), o gasto total em dispositivos e serviços de IoT atingiu quase US $ 2 trilhões em 2017, e haverá mais de 20 bilhões de "coisas" conectadas em todo o mundo até 2020.

Entretanto, apesar de possuir a natureza distribuída, a maioria das soluções IoT ainda dependem de arquiteturas que seguem o modelo cliente servidor. Embora esse modelo arquitetural possa funcionar hoje, o crescimento da IoT sugere que novas arquiteturas deverão ser propostas no futuro [Caramés e Lamas 2017]. Uma alternativa que vêm sendo explorada pela comunidade acadêmica e a indústria é o emprego de blockchains devido a sua, capacidade de manter os registros imutáveis sem perder segurança, com algoritmos que tratam nodos maliciosos.

Com esse cenário e a falta de padronização dos dispositivos IoT devido a seus diferentes objetivos existe uma dificuldade na implementação e modelagem de uma rede blockchain personalizada para dispositivos IoT [Conoscenti et al, 2016]. Assim, o presente trabalho apresenta o Demochain, um framework voltado para a criação de redes blockchain híbridas para dispositivos IoT. Um dos diferenciais do Demochain é a sua capacidade de oferecer opções para mesclar e combinar diferentes níveis na arquitetura e também funcionalidades da blockchain pura (totalmente descentralizada), variando seus protocolos e criptografias, construindo assim, uma blockchain híbrida. Além disso, pode-se ressaltar que a maior contribuição do Demochain é a possibilidade de facilitar a modelagem e a prototipação de novas redes blockchain em ambientes variados.

2. Framework Proposto

O framework construído foi chamado de Demochain e foi pensado para diminuir a complexidade na conexão dos dispositivos. Portanto, vários conceitos utilizados em

blockchains tradicionais foram simplificados. Ele foi desenvolvido com a linguagem Go [Google 2009] e foi utilizada uma abordagem de orientação a objetos com o padrão Decorator em sua implementação [Schmager, Frank 2010]. A arquitetura que foi implementada é multicamadas, baseada no trabalho de Li e Zhang (2017). Nessa arquitetura, cada nodo na rede pode assumir dois papéis: ser um nodo centralizador (Edge Node), que serve para controlar o acesso dos nodos abaixo na camada, ou ser um nodo de alto nível (High-Level Node), também chamado de nodo filho, onde são adicionado blocos e executado o protocolo de consenso.

Com isso tem-se um modelo descentralizado com alguns níveis de centralização, porém não totalmente distribuído. A Figura 1 apresenta a arquitetura do Demochain.

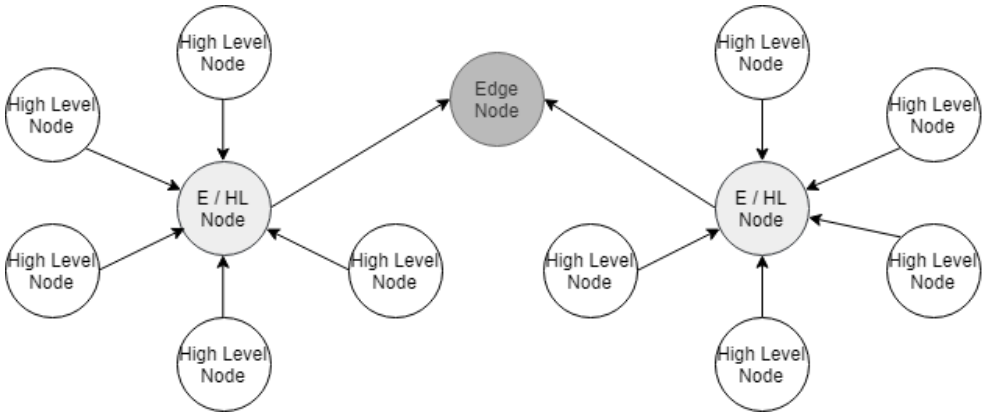

Figura 1. Arquitetura implementada.

Ao escolher o esquema de criptografia para o framework, foi levado em consideração não apenas a segurança fornecida de acordo com a carga computacional, mas também o consumo de energia, fazendo um trade-off entre segurança e consumo. Foram implementados quatro algoritmos de criptografia assimétrica: RSA, Ed25519, Secp256k1 e ECDSA. A identificação dos nodos ocorre utilizando a pilha TCP em sua forma tradicional. Portanto, um mesmo dispositivo pode estar rodando várias instâncias do Demochain, desde que em portas distintas.

No Demochain a blockchain é replicada para todos os nós. Entretanto o diferencial deste framework é que a execução do consenso é por nível. Protocolo consenso consiste em um mecanismo que determina as condições a serem alcançadas para concluir que um acordo foi alcançado em relação às validações dos blocos para ser adicionado ao blockchain [Zheng et al. 2017]. Foram implementados três protocolos de consenso: PoW (Proof of Work), PoS (Proof of Stake) e PBFT (Practical Byzantine Fault Tolerance).

2.1. Proof of Work (PoW)

Algoritmos de consenso PoW, ou Prova de Trabalho baseiam-se no fato de que um nó malicioso tem que executar muito trabalho do ponto de vista computacional para atacar a rede, e por isso, é menos provável que ele vá querer atacar. O trabalho realizado geralmente envolve fazer alguns cálculos até que uma solução seja encontrada, um

processo que é comumente conhecido como mineração. No caso da blockchain do Bitcoin, a mineração consiste em encontrar um número aleatório, chamado de número nonce que fará o hash SHA256 do cabeçalho do bloco para ter no início certo número de zeros. Portanto, os mineradores têm que demonstrar que eles realizaram certa quantidade de trabalho para resolver o problema. Uma vez resolvido o problema, é realmente fácil para outros nodos verificar se a resposta obtida é válida. No entanto, este processo de mineração faz a blockchain ineficiente em taxa de transferência, escalabilidade [M. Vukolić 2015], e também em termos de consumo de energia, o que não é desejável em uma IoT rede.

2.2. Proof of Stake (PoS)

PoS ou Prova de Participação é um mecanismo de consenso que requer menos recursos computacionais e potência do que PoW [BitFury Group 2015], por isso consome menos energia. Em uma blockchain baseado em PoS, assume-se que as entidades com mais participação na rede são os menos interessados em atacá-lo. Assim, os nodos precisam provar periodicamente que eles possuem certa quantidade de participação na rede (por exemplo, moeda ou quantidade de dados coletados a partir de sensores). Podemos comparar o PoS há uma loteria, onde é realizado um sorteio e escolhido um nodo que irá dizer se o bloco que está sendo enviado pela rede é válido ou não, por consequência, o nodo que tiver mais participação recebe mais "fichas" neste sorteio, e tem mais chances de validar o bloco. O nodo que realiza o sorteio pode ou não ser pré-definido.

2.3. Practical Byzantine Fault Tolerance (PBFT)

PBFT [M. Castro e B. Liskov 1999] ou Tolerância de Falhas Bizantinas Prática é um algoritmo de consenso que resolve o problema dos generais bizantinos, recomendado para ambientes assíncronos. PBFT assume que menos de um terço dos nós são maliciosos. Para cada bloco a ser adicionado à cadeia, um líder é selecionado para ser encarregado de encomendar a transação. Essa seleção deve ser apoiada por pelo menos 2/3 de todos os nós, que devem ser conhecidos pela rede.

2.4. Implementação do consenso

No Demochain o PoW funciona da maneira clássica, é imposta uma dificuldade e é realizada uma tentativa aleatória de encontrar uma Hash utilizando SHA256 que comece com o número de zeros da dificuldade. Na figura 2 temos um exemplo de Hash's geradas de acordo com a dificuldade do PoW, onde é possível verificar o número de zeros no início da Hash.

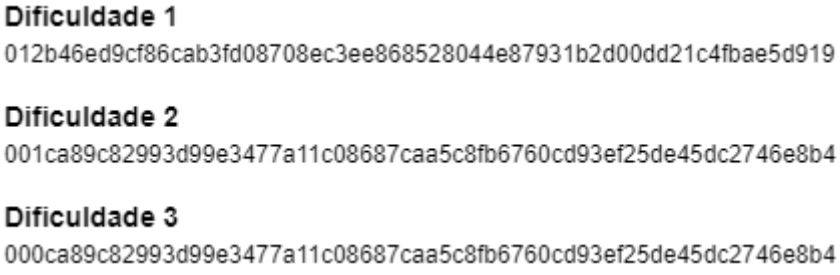

Figura 2. Dificuldade do Proof of Work.

Como o PoW processa os blocos apenas com os próprios recursos, sem depender da aprovação dos demais nodos, no Demochain utilizando o PoW é possível realizar a validação e geração de novos blocos independente da conexão com o seu Edge Node, portanto mesmo um dispositivo estando desconectado da rede, ao se reconectar ele consegue sincronizar os blocos validados, isso se ele estiver executando a mesma instância, pois esses blocos ficam armazenados em memória, entretanto caso sua execução seja interrompida e ele não estiver conectado na rede os blocos validados são perdidos, já que foi implementado apenas um arquivo físico de Blockchain que mantém o conteúdo atualizado conforme o conteúdo atual que está circulando na rede. Uma implementação com dois arquivos de Blockchain, um atualizado conforme a rede e outro com o conteúdo "local" é uma possível melhoria, conforme é sugerido em [Liang et al. 2017].

A razão dessa funcionalidade é dar liberdade ao desenvolvedor dizer quando deve ser o envio dos blocos, não precisando ser necessariamente logo após a validação dos mesmos, o que em um cenário de concorrência de criptomoedas não faz sentido, mas em redes privadas pode fazer, dependendo dos objetivos de cada um.

A implementação proposta para o PoS é baseada no trabalho de Dan Larimer (2014), também chamado de DpoS (Delegate Proof of Stake). A diferença do DPoS para o PoS "clássico" é que são delegados representantes para a realização do sorteio para a governança da rede com critérios predeterminados. Nesse caso a cada bloco gerado será solicitado um sorteio ao Edge Node, que irá direcionar o bloco ao vencedor para realizar a validação. Esse sorteio acontece no escopo de apenas um nível de rede, porém após ser validado o bloco é passado aos demais níveis. A figura 3 demonstra o funcionamento do PoS.

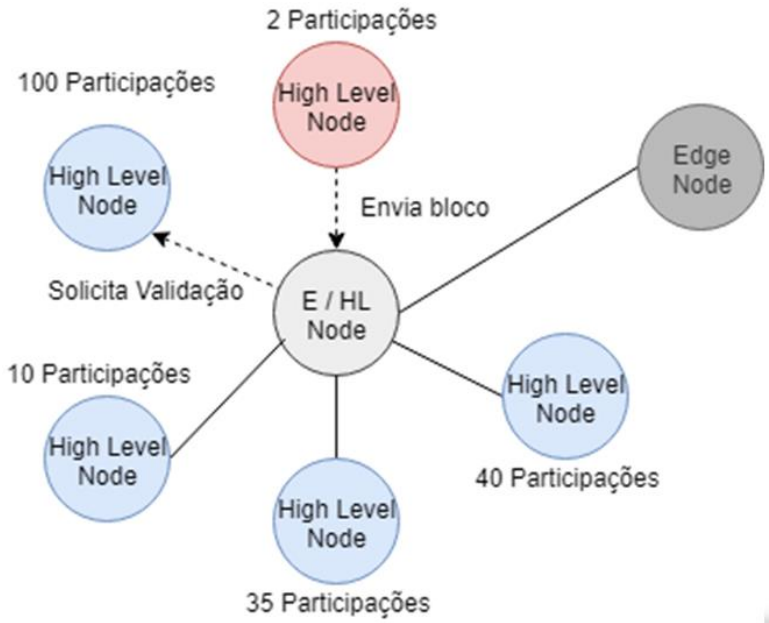

Figura 3. Funcionamento do PoS.

A implementação proposta para o PBFT também tem alterações. Assim como no PoS, toda validação deverá ser solicitado ao Edge Node da rede, porém ele irá enviar o bloco para todos os nodos do mesmo nível, ou seja, todos os High Level Nodes. Após o

mesmo irá contabilizar quem aprovou e quem desaprovou o bloco. Se pelo menos 2/3 de seus High Level Nodes aprovarem o bloco, então é decretado que o bloco é válido. Assim como o PoS o protocolo executa em um nível da rede, porém ao ser validado o bloco é passado aos demais níveis. A figura 4 demonstra o funcionamento do PBFT.

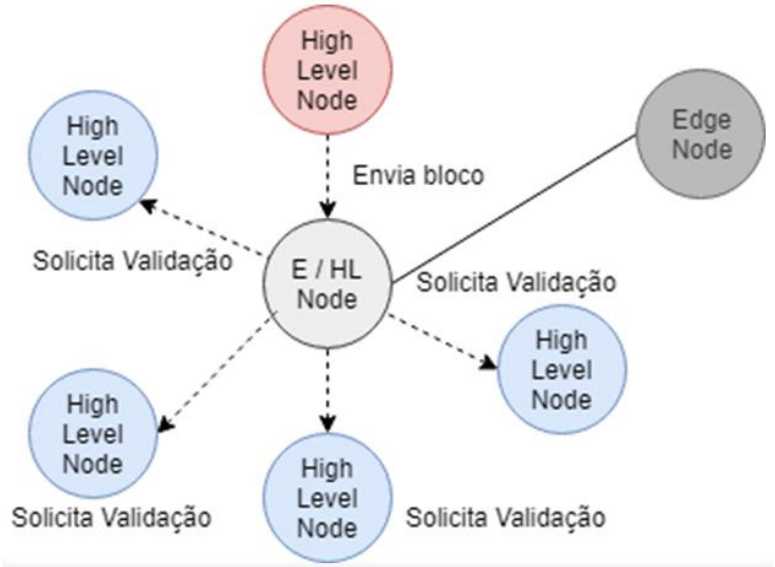

Figura 4. Funcionamento do PBFT.

Uma observação importante é que como os protocolos PoS e PBFT irão executar em apenas um nível da rede, uma preocupação que o desenvolvedor deve ter é a divisão dos Edge Nodes, pois quanto menos High Level Nodes vinculados a ele maior a chance de um nodo intruso realizar uma ação maliciosa, pois são protocolos probabilísticos.

Será suportado apenas um protocolo de consenso rodando em todos níveis de rede. Uma possível melhoria seria permitir cada High-Level Layer estar rodando um tipo de protocolo de consenso, ou mesmo não ter nenhum implementado, podendo apenas estar inserindo os blocos e os compartilhando sem protocolo de consenso, entretanto isso exigiria um esquema mais complexo de armazenamento e uma compressão dos dados validados de cada nível da rede, fazendo com que cada High-Level Layer tivesse uma blockchain diferente da outra em seu formato.

2.5. Mensagens

A troca de mensagens se dá de maneira simples, podendo ter um ou dois fluxos em cada conexão, um de recebimento e outro para envio. Caso o desenvolvedor defina uma rede não permissionada sempre será aberto os fluxos de recebimento e de envio, porém é possível definir também uma rede permissionada. Isso permite com que o desenvolvedor defina nodos apenas de backup das informações para eventuais falhas. Na figura 5 temos um exemplo ilustrativo de uma rede permissionada com o Demochain.

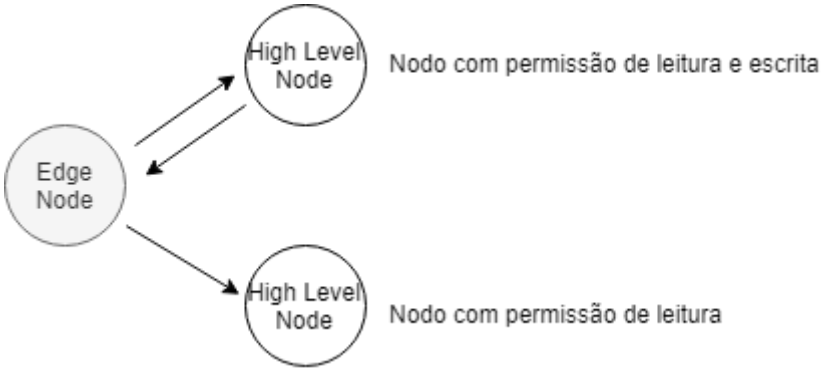

Figura 5. Rede permissionada com nodo de leitura.

Em uma primeira versão o demochain envia a Blockchain por inteira a cada mensagem trocada, o que a um longo prazo pode gerar gargalos na rede. Um controle mais complexo de troca de mensagens não foi implementado.

2.6. Estrutura dos blocos

A estrutura de blocos utilizadas pelo Demochain possui semelhanças e diferenças com as estruturas tradicionais utilizadas em criptomoedas. É importante ressaltar que normalmente em criptomoedas um bloco é formado de transações, sendo que toda transação tem um "recipient", podendo ou não ter um "sender". No Bitcoin por exemplo, em uma transação comum temos um "sender" que envia valor para o "recipient", porém quando um novo bloco é minerado é chamado de "Coinbase", pois não tem um "sender". Como esse framework não é voltado para criptomoedas foi utilizado um conceito mais simples somente com blocos, pois a princípio a validação que será aplicada nos dados coletados não deverá ser passada para outros Nodos. Abaixo temos a estrutura de bloco que o Demochain implementa:

- Index: Índice sequencial único que é incrementado a cada nova adição de blocos na blockchain.
- Timestamp: Carimbo de data e hora do momento da adição do bloco.
- Data: Dados que foram validados.
- Hash: Hash gerada para o bloco.
- PrevHash: Hash do bloco anterior.
- Consensus: Consenso utilizado para a validação do bloco.
- Nonce: Número Nonce é utilizado apenas para o consenso Proof of Work.
- Target: Target do nodo onde foi validado o bloco. Podemos dizer que corresponde ao "dono" do bloco.

2.7. Armazenamento

Para o armazenamento será gravado em um arquivo que servirá para manter os blocos já validados por todos os nodos. O arquivo será gravado com extensão ".bc" com um Json não identado com todos os dados da blockchain. Na figura 6 temos um exemplo de um

bloco gerado com a estrutura de bloco do Demochain, e na figura 7 temos o mesmo bloco armazenado no arquivo ".bc".


```
[
   {
      "Index":0,
      "Timestamp":"2018-11-17 19:02:25.834476 -0200 -02 m=+3.635126501",
      "Data":"0",
      "Hash":"f1534392279bddbf9d43dde8701cb5be14b82f76ec6607bf8d6ad557f60f304e",
      "PrevHash":"",
      "Consensus":{
         "TypeConsensus":1,
         "Difficulty":3
      },
      "Nonce":"",
      "Target":"QmbiS1uqyVVXzV2pYdife8vJeZZRtW4TQiNEP4Dj1ckNiG"
   }
]
```


Figura 6. Bloco json identado.


```
[{"Index":0,"Timestamp":"2018-11-17 19:02:25.834476 -0200 -02
m=+3.635126501","Data":"0","Hash":"f1534392279bddbf9d43dde8701cb5be14b82f76ec6607bf8d6
ad557f60f304e","PrevHash":"","Consensus":{"TypeConsensus":1,"Difficulty":3},"Nonce":""
,"Target":"QmbiS1uqyVVXzV2pYdife8vJeZZRtW4TQiNEP4Dj1ckNiG"}]
```


Figura 7. Bloco json não identado.

A seguir serão apresentados os resultados de simulações de uso do framework com o objetivo de demonstrar a performance de cada protocolo de consenso sobre um mesmo desenho de arquitetura.

3. Resultados

Todos os códigos bem como testes realizados estão disponíveis em Freitas (2018). Os resultados foram obtidos com o uso de equipamentos para simular um ambiente IoT. A Tabela 1 demonstra a configuração do hardware utilizado para os testes do Demochain.

Tabela 1. Tabela de hardware para experimentação.

Código	Descrição	Processador	Memória RAM	Sistema Operacional
H01	Notebook	Intel Core i5 2x 2.20 GHz	4 Gigabytes	Windows 10 Pro
H02	Raspberry Pi	BCM2835	512 Megabytes	Raspbian Stretch Lite

Para testar o desempenho do framework foi utilizado as métricas descritas por S. Angelis (2017), que são comumente usadas para medir aplicações descentralizadas. A Tabela 2 descreve as métricas utilizadas.

Tabela 2. Tabela de métricas para experimentação.

Código	Nome	Descrição
M01	Taxa de validação (Throughput)	Mede o número de blocos gerados com sucesso por segundo.
M02	Latência (Latency)	Mede o tempo entre envio e confirmação do bloco relativo em milisegundos.
M03	Escalabilidade (Scalability)	Média em milissegundos das mudanças de latência, conforme é aumentado o número de nodos na rede.
M04	Tempo Total (Total Time)	Mede o tempo total em minutos de execução do programa de teste.

A fim de simular dados reais de ambientes IoT foi utilizado um dataset público [Ortiz, J. e Gottschlich, N. 2016]. A Tabela 3 descreve o dataset utilizado.

Tabela 3. Tabela de datasets para experimentação.

Código	Descrição	Referência
D01	Dados de consumo de eletricidade doméstica com 60.640 medições coletadas entre janeiro de 2007 e junho de 2007 (6 meses).	[Ortiz, J. e Gottschlich, N. 2016]

Para testar o framework foi definido cenários de experimentação, no qual para cada cenário foi criado um aplicativo de testes que importa o Demochain. Os cenários possuem dois objetivos principais, demonstrar exemplos de como utilizar o framework e analisar o seu desempenho. Nas subseções seguintes está descrito cada cenário, seu objetivo e seus resultados.

3.1. Cenário 1

O Objetivo deste cenário é comparar desempenho de geração de blocos, utilizando as métrica M01 e M04. Para isso foi implementado um nodo único para testar o protocolo de consenso Proof of Work com diferentes níveis de dificuldade utilizado o dataset D01 como entrada de dados. A cada execução do Proof of Work foi incrementada a sua dificuldade, variando de 1 a 5. A Tabela 4 demonstra os resultados obtidos deste cenário, e as figuras 8 e 9 demonstram os resultados retirados da tabela 4.

Tabela 4. Resultados da validação do PoW aplicando as métricas M01 e M04.

	Taxa de validação (blocos p/ segundo)	**Tempo Total (minutos)**
PoW 1	10 Blocos	434 min
PoW 2	8 Blocos	543 min
PoW 3	5 Blocos	869 min
PoW 4	3 Blocos	1.448 min
PoW 5	1 Bloco	4.344 min

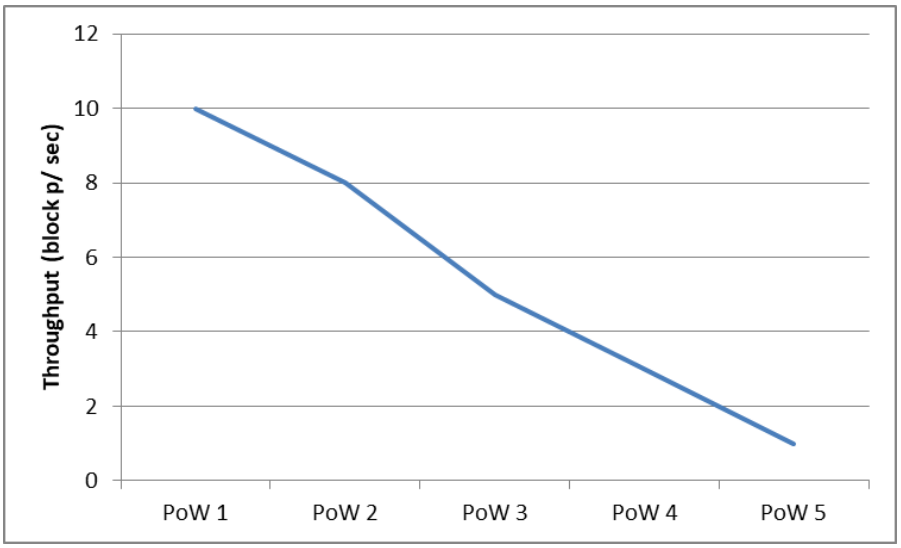

Figura 8. Aplicação da métrica M01 sobre o PoW.

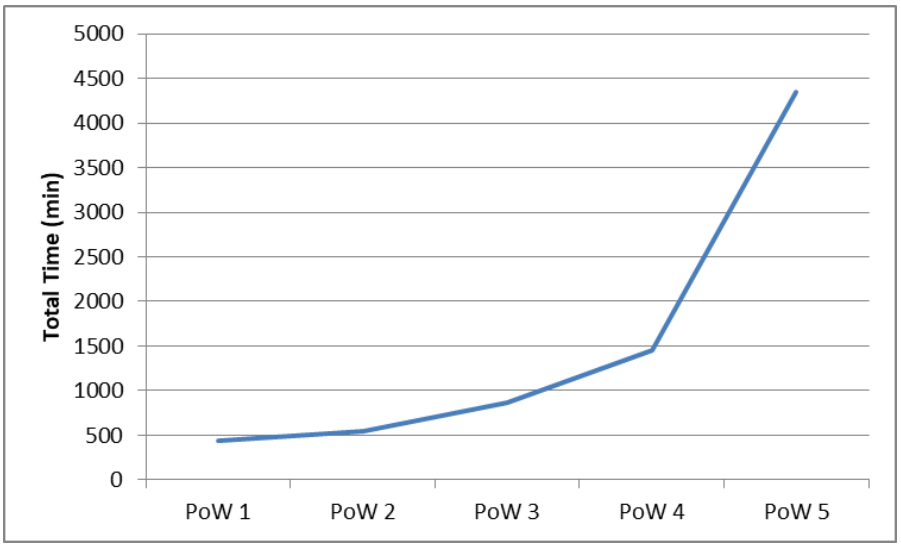

Figura 9. Aplicação da métrica M04 sobre o PoW.

Os resultados obtidos neste cenário demonstram que o Proof of Work tem um desempenho razoável na validação dos blocos até a dificuldade 3, portanto é uma boa alternativa para os nodos que não ficam conectados a todo momento, pois o seu processamento não depende da rede.

3.4. Cenário 2

O objetivo deste cenário é comparar desempenho de cada protocolo de consenso, utilizando as métricas M01, M02, M03 e M04. Para isso foi implementado um rede com até 15 nodos dispostos nos hardwares H01 e H02, utilizando a criptografia RSA, com o dataset D01 como entrada de dados, sendo que foi utilizado o Proof of Work apenas com dificuldade 3. Todos os nodos estavam rodando sobre um mesmo nível de rede, ou seja, todos respondiam para o mesmo Edge Node.

A Tabela 5 apresenta a aplicação da métrica M01 sobre os três protocolos de consenso, considerando a variação do número de nodos na rede, e a figura 10 demonstra com um gráfico de linhas os resultados desta tabela.

Tabela 5. Taxa de validação (Throughput) em blocos p/ Segundo.

	PoW 3	**PoS**	**PBFT**
1 Nodo	5 blocos	25 blocos	20 blocos
2 Nodos	4,7 blocos	21 blocos	15,7 blocos
5 Nodos	4,2 blocos	15 blocos	11 blocos
10 Nodos	3,5 blocos	10,5 blocos	6,5 blocos
15 Nodos	2,5 blocos	6,5 blocos	2,6 blocos

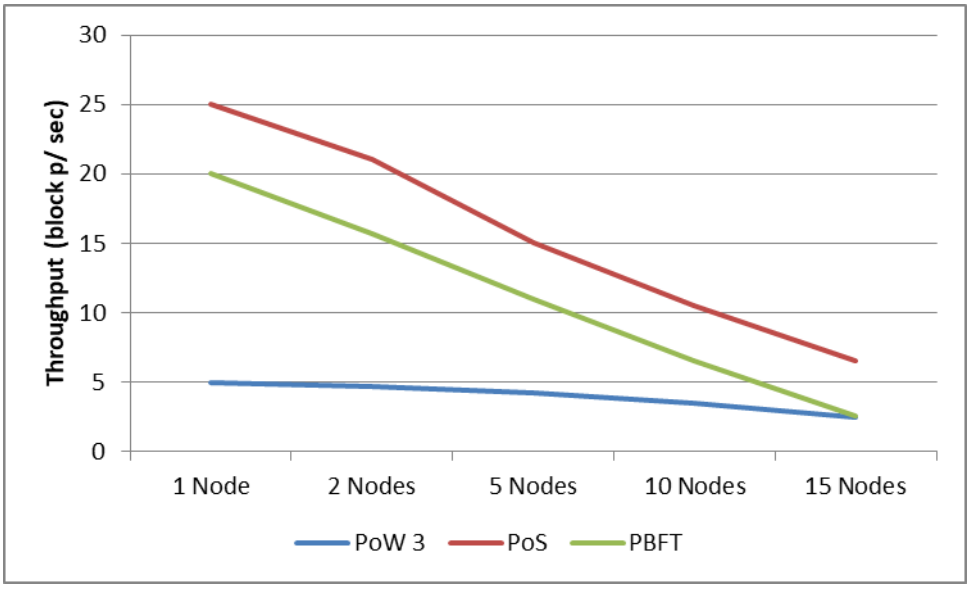

Figura 10. Aplicação da métrica M01.

A Tabela 6 apresenta a aplicação da métrica M02 sobre os três protocolos de consenso, considerando a variação do número de nodos na rede, e a figura 11 demonstra com um gráfico de linhas os resultados desta tabela.

Tabela 6. Latência (Latency) em milisegundos.

	PoW 3	**PoS**	**PBFT**
1 Nodo	200 ms	40 ms	50 ms
2 Nodos	212 ms	47 ms	64 ms
5 Nodos	238 ms	66 ms	90 ms
10 Nodos	285 ms	96 ms	154 ms
15 Nodos	400 ms	153 ms	384 ms

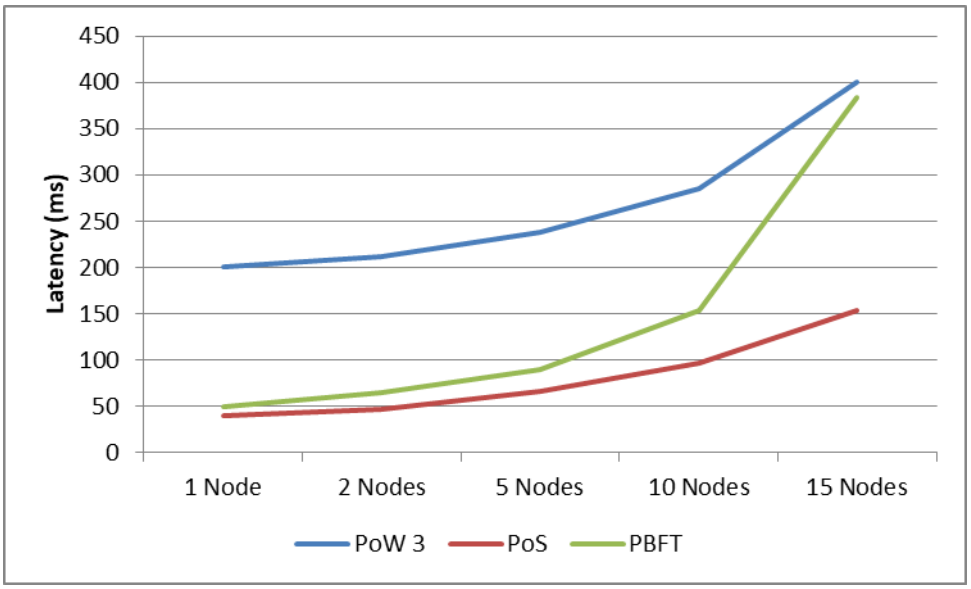

Figura 11. Aplicação da métrica M02.

A Tabela 7 apresenta a aplicação da métrica M03 sobre os três protocolos de consenso, considerando a variação do número de nodos na rede, e a figura 12 demonstra com um gráfico de barras os resultados desta tabela.

Tabela 7. Escalabilidade (milisegundos).

	PoW 3	**PoS**	**PBFT**
Média de Latência p/ Nodo	13,33 ms	7,53 ms	22,26 ms

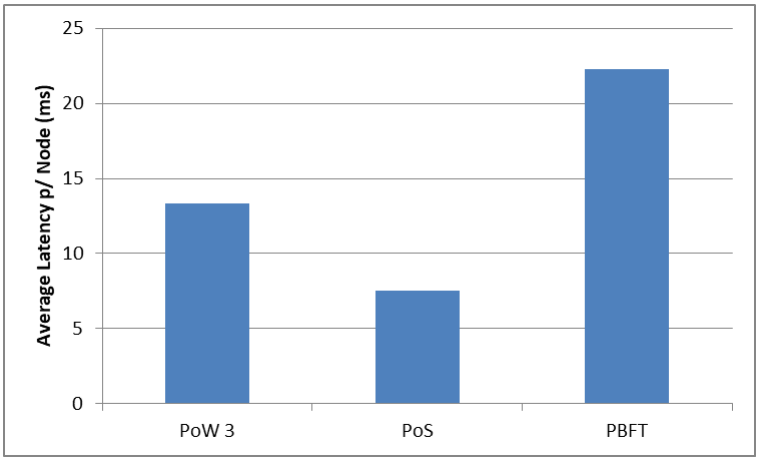

Figura 12. Aplicação da métrica M03.

A Tabela 8 apresenta a aplicação da métrica M04 sobre os três protocolos de consenso, considerando a variação do número de nodos na rede, e a figura 13 demonstra com um gráfico de linhas os resultados desta tabela.

Tabela 8. Tempo total da execução.

	PoW 3	**PoS**	**PBFT**
1 Nodo	869 min	174 min	217 min
2 Nodos	462 min	103 min	138 min
5 Nodos	207 min	58 min	79 min
10 Nodos	124 min	41 min	67 min
15 Nodos	116 min	44 min	111 min

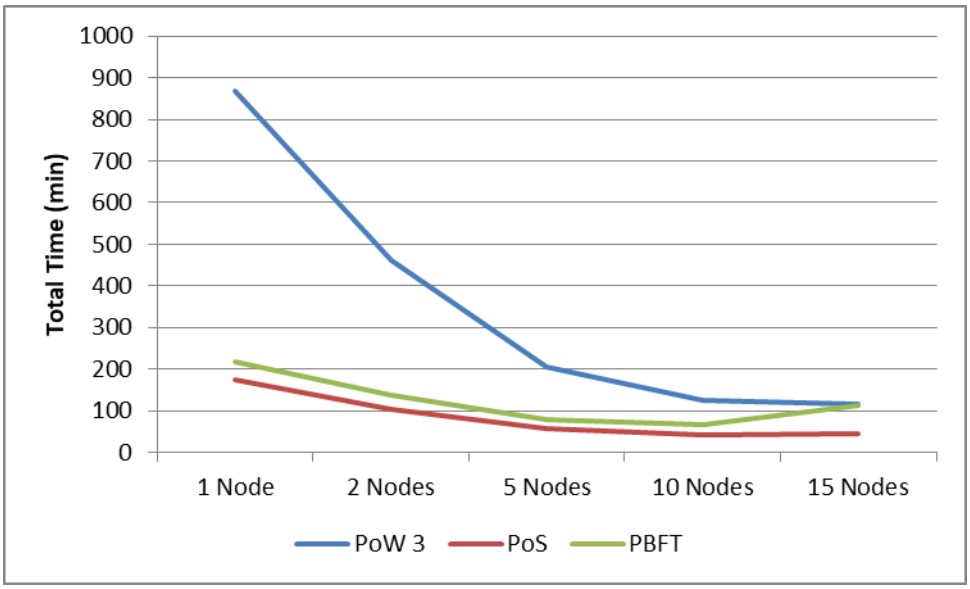

Figura 13. Aplicação da métrica M04.

Do ponto de vista de desempenho pode-se notar que o protocolo PoW é o menos performático, pois necessita de poder computacional para achar uma Hash específica com força bruta, porém é o mais escalável de todos já que a performance isolada de cada nodo não é influenciada pelos demais, apenas pelo seu próprio processamento, já para os protocolos PoS e PBFT a performance vai variar de acordo com a arquitetura modelada, já que a validação dos blocos é coletiva, depende da conexão com outros nodos, entretanto vale ressaltar que dependendo da disposição dos nodos, o nível de segurança varia, pois em um nível com apenas 2 nodos, por exemplo, caso um dos nodos se torne malicioso 50% do nível da rede está infectada, e como os outros níveis apenas "aceitam" os blocos deste nível infectado toda a rede pode ficar comprometida.

4. Conclusão

Esse artigo teve por objetivo apresentar o Demochain, um framework para a criação de redes blockchain híbridas para dispositivos IoT. O trabalho realizado até o momento demonstra que o framework proposto é capaz de auxiliar no desenvolvimento de redes blockchain híbridas para ambientes IoT. O estudo realizado e o framework desenvolvido propiciaram a abstração de diversos conhecimentos, possibilitando assim que trabalhos relacionados possam ser desenvolvidos utilizando este como base em diversos aspectos.

Assim, pode-se concluir que não existe uma solução única se tratando de redes blockchain para dispositivos IoT. No entanto, a adoção de um framework para auxílio no desenvolvimento abre uma ampla possibilidade de novas soluções cada vez mais performáticas em diferentes contextos.

Como trabalhos futuros pode ser apontado a implementação de novos protocolos de consenso nesta arquitetura híbrida, com a possibilidade de cada camada estar rodando o seu próprio consenso, além de novos controles de armazenamento e envio da blockchain, e também utilizar outras métricas de desempenho e segurança para testes com este framework, em diferentes ambientes IoT.

5. Refêrencias

Bitfury Group (2015). "Proof of Stake versus Proof of Work. White Paper". LINK DE ACESSO. Ultimo Acesso: 29/11/2018.

Castro, M. e Liskov, B. (1999). "Practical Byzantine fault tolerance".

Conoscenti, M., Vetrò, A. e De Martin, J. C. (2016). "Blockchain for the Internet of Things: A systematic literature review".

De Angelis, Stefano. (2017). "Assessing Security and Performances of Consensus algorithms for Permissioned Blockchains".

Fernández Caramés, M., Fraga Lamas, D. P. (2017). "A Review on the Use of Blockchain for the Internet of Things".

Freitas, L. (2018). Github com códigos do Demochain. Disponível em https://github.com/LorenzoWF/Demochain. Ultimo acesso em 29/11/2018.

Gartner (2017). "Leading the IoT – Gartner Insights on how to lead in a connected world".

Google (2009). Golang site oficial. Disponível em https://golang.org. Último Acesso: 29/11/2018.

Larimer, Dan (2014). “DPOS Description on Bitshares”.

Li, C. e Zhang, L. J. (2017). “A blockchain based new secure multi-layer network model for Internet of Things”.

Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K. e Njilla, L. (2017). “Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability”.

Ortiz, J. e Gottschlich, N. (2016). Base de dados “Household Power Consumption”. Disponível em https://data.world/databeats/household-power-consumption. Último Acesso: 29/11/2018.

Schmager, Frank (2010). “Evaluating the GO Programming Language with Design Patterns”.

Vukolić, M. (2018). “The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication”.

Zheng, Z., Xie, S., Dai, H. e Wang, H. (2017). “An overview of blockchain technology: Architecture, consensus, and future trends”.