IoT com Blockchain

OWFS (simulação, docker)

IoT com Blockchain

motivação:

* manter um registro seguro e imutável dos dados coletados pelos sensores.
* gerenciar o controle de acesso (conceder acesso a dispositivos, dados e serviços de sistemas IoT com base em políticas específicas) (O BC serve como armazenamento seguro e canais de comunicação em sistemas BC-IoT ao manter os dados e eventos do mundo real relatados por dispositivos IoT, por exemplo, Xie et al., 2017; Hossain et al., 2018; Kak et al., 2018; Zinonos et al. , 2019).

Modelos de implementação (deploy):

* **Na nuvem** (Cloud-based deployment, implantado em uma infraestrutura externa)
  + Exemplos:
    - Ethereum (rede BC pública);
    - Blockchain-as-a-service:
      * IBM Block-chain Platform

<https://www.ibm.com/blockchain/platform>

* Microsoft Azure Blockchain

<https://azure.microsoft.com/en-au/services/blockchain-service/>

* Blockchain on AWS5

<https://aws.amazon.com/blockchain/>

* Vantagens:
  + Facilidade de operação porque elimina o gerenciamento da infraestrutura.
* **Fog nodes** (nós localizados entre a nuvem e os sensores)
  + Exemplos:
    - Afanasev et al. (2018, <http://dx.doi.org/10.23919/FRUCT.2018.8468296> ) implantou uma **rede Ethereum privada** em controladores de um sistema IoT industrial para controlar dispositivos com contratos inteligentes e manter dados de sensor à prova de violação;
    - Agrawal et al. (2018, <http://dx.doi.org/10.1109/ICASSP.2018.8462513> ) implantou uma rede privada **Hyperledger Fabric (**[**https://www.hyperledger.org/use/fabric**](https://www.hyperledger.org/use/fabric) **)** BC em fog nodes que gerenciam diferentes zonas de um edifício inteligente para permitir autorização contínua;
    - Yang et al. (2018, <http://dx.doi.org/10.1109/JIOT.2018.2836144> ) implantou uma rede BC personalizada em unidades de beira de estrada (RSU, roadside units) para armazenar e calcular classificações de reputação de veículos inteligentes.
  + Vantagens:
    - A implantação em “névoa” permite que os desenvolvedores modifiquem os parâmetros de infraestrutura de uma rede BC, como taxa de blocos e dificuldade do hash, para se adequar a seus sistemas IoT.
* **Na borda** (Edge-based deployment, menos comum, desafio de ajustar os nós BC a dispositivos de borda com capacidade limitada de computação e armazenamento)
  + Yang et al (2017, <http://dx.doi.org/10.1109/PIMRC.2017.8292724>) implantou uma rede BC personalizada diretamente em computadores de bordo de veículos para manter as classificações de reputação de outros veículos. Essas classificações ajudam os veículos a avaliar a confiabilidade de anúncios vindos de terceiros sem consultar backends em nuvens ou fog nodes na beira da estrada

Vários sistemas BC-IoT utilizam nós completos e leves para sua rede BC. A configuração mais comum é ter nós completos na nuvem e nós leves em fog nodes. O objetivo dessas configurações é trazer as redes BC para mais perto da borda sem trazer a demanda de recursos delas.

Esses sistemas mantêm as operações de mineração intensivas em nós com muitos recursos e trazem apenas o livro-razão (ledger) para os dispositivos com recursos limitados.

Quanto ao conteúdo:

* Dados on-chain: Os dados que uma rede BC armazena podem ser:
  + codificados como transações
  + armazenados nos dados da transação ou
  + mantidos em variáveis ​​internas de contratos inteligentes.

Um tipo de dados comum on-chain são as leituras de sensores e seus hashes. Esses tipos de dados são geralmente reservados para dados de alto risco e propensos à adulteração.

* Exemplos:
  + Niya et al. (2018, <http://dx.doi.org/10.1109/NOMS.2018.8406329>) usou redes BC para armazenar dados de detecção de poluição, incluindo Hidrogênio Potencial (Potential Hydrogen), turbidez, Monóxido de Carbono e Dióxido de Carbono. O BC garante a integridade desses dados, tornando-os evidências confiáveis ​​de poluição.
  + Uddin et al. (2018, <http://dx.doi.org/10.1109/ACCESS.2018.2846779>) armazena leituras fisiológicas de sensores corporais (body area sensors) no BC para permitir o monitoramento remoto do paciente.
  + Xie et al. (2017, <http://dx.doi.org/10.1109/BIGCOM.2017.43>) registrou o status do ambiente ao redor de produtos agrícolas à medida que eles se moviam pelas cadeias de abastecimento usando sensores IoT e os armazenavam em um BC. Esses registros ajudam a detectar problemas de segurança alimentar.
* Lógica on-chain: as redes BC podem executar programas de usuários como contratos inteligentes. Esses programas aceitam transações como entradas e geram o próximo estado do BC como saídas. Esse cálculo ocorre no processo de mineração e todos os nós completos podem verificar suas saídas. Portanto, a execução de código por meio de contratos inteligentes é verificável. A desvantagem é que os aplicativos distribuídos baseados em contratos inteligentes podem ser ordens de magnitude mais lentos do que os aplicativos centralizados.
  + Metade não utiliza nenhuma lógica on-chain, BC serve apenas como um mecanismo para armazenar e compartilhar dados. O sistema IoT não transfere a lógica para o BC.
  + Lógicas mais utilizadas:
    - Contratos de troca de recursos e
    - Controle de acesso
  + Lógica incomum:
    - Niya et al. (2018, <http://dx.doi.org/10.1109/NOMS.2018.8406329>) implementou a lógica de detecção de eventos com contratos inteligentes. Eles usaram contratos inteligentes on-chain para monitorar dados de sensores ambientais. As violações de limite acionam e emitem eventos, que permanecem imutáveis no log da rede BC.

A configuração de uma rede IoT com block considera os seguintes aspectos:

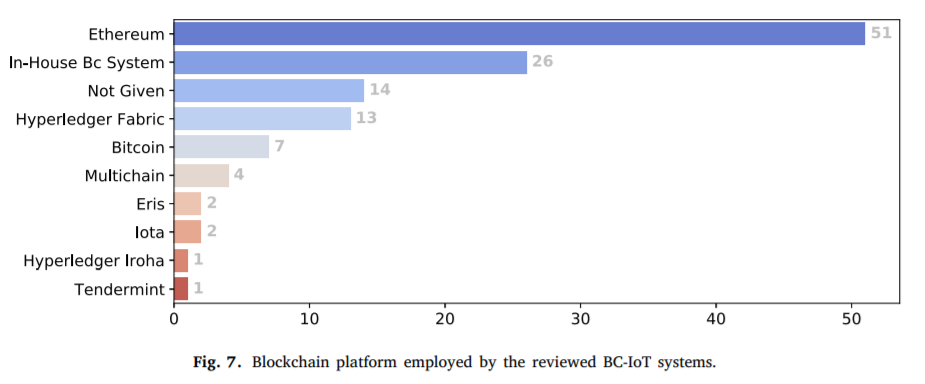
* o número de redes BC que um sistema IoT usa,
  + maioria usa uma
* o tipo de permissão dessas redes,
  + três tipos de permissão comuns são:
    - pública,
    - privada e
    - consórcio.

Os tipos de permissão privada e de consórcio restringem o acesso a uma rede BC. Essas entidades autenticam e autorizam participantes. Eles também podem rastrear a identidade dos participantes. Portanto, protocolos custosos de proof-of-work geralmente são desnecessários para essas redes BC. Em vez disso, eles optam por protocolos de consenso menos intensivos que abordam apenas Byzantine Fault (?) para acelerar seu processamento e finalização

* o tipo de consenso que usam e
  + ver gráfico no artigo

Um número esmagador de sistemas BC-IoT usam algumas variantes do protocolo de proof-of-work (PoW), como o protocolo Dagger-Hashimoto da Ethereum, o protocolo de consenso Nakamoto do Bitcoin ou protocolos desenvolvidos internamente. Nestes protocolos, os participantes da rede (ou seja, mineiros) correm para encontrar um número (também conhecido como nouce) para tornar o hash do bloco menor do que um valor limite. O primeiro mineiro a encontrar o nonce ganha o direito de anexar seu bloco à corrente e receber as recompensas de mineração. A probabilidade de um minerador ganhar uma rodada de mineração depende de sua taxa de hash. **PoW protege BCs públicos ao custo de um consumo de energia significativo**. Alguns dos sistemas BC-IoT revisados modificaram o PoW para resolver esse problema. Por exemplo, Uddin et al. propôs um PoW seletivo. Este protocolo remove a corrida entre os mineiros, selecionando um mineiro para trabalhar por vez. Os clientes classificam e selecionam os mineiros por uma classificação de confiança.

A tolerância prática a falhas bizantinas (**Practical Byzantine Fault Tolerance, pBFT**) é uma alternativa ao PoW. Byzantine Fault é uma condição em um sistema distribuído em que um componente pode falhar, mas a informação disponível é imperfeita para decidir se falhou. O protocolo pBFT garante que uma rede BC pode chegar a um consenso sobre o próximo estado, apesar das falhas bizantinas. Ele opera menos de 1/3 dos nós que são maliciosos (Cachin e Vukolić, 2017, <https://arxiv.org/abs/1707.01873> ). Diferente do PoW, o pBFT não exige uma solução de quebra-cabeças dispendiosa para conter os ataques Sybil. Portanto, é mais rápido, mas menos seguro. **BC privadas e de consórcio, como Hyperledger Fabric**, tendem a usar pBFT. Como essas BC examinam seus participantes, é improvável que 1/3 dos nós sejam maliciosos e que ocorram ataques Sybil.

* a tecnologia BC usada para construí-las.
  + ”Ethereum é a tecnologia mais comum para construir BCs integrados. A ampla adoção do Ethereum pode ser atribuída à sua chegada antecipada, suporte para contratos inteligentes programáveis ​​e pilhas de tecnologia relativamente maduras. **O Hyperledger Fabric também está emergindo no campo de pesquisa. Sua estrutura modular e suporte para redes privadas podem ser os principais fatores**. Depois, ​​temos tecnologias como Multichain, Monax e Eris, IOTA e Hyperledger Iroha. Finalmente, muitos trabalhos envolvem a implementação de BC proprietária. Nós os classificamos sob o rótulo de sistemas BC internos. Também extraímos outros recursos que caracterizam a arquitetura interna e a operação das redes BC, incluindo a estrutura de dados de seu ledger, o modelo de seu estado global e os tipos de seus contratos inteligentes. Não observamos nenhum desvio dessas características em relação às normas que as plataformas BC estabelecem. Por exemplo, se um sistema BC-IoT usa Ethereum, então seu BC geralmente usa uma estrutura de dados baseada em bloco, usa contas para modelar o estado global e segue o modelo de contrato inteligente na cadeia. Assim, por uma questão de concisão, não entraríamos em detalhes sobre esses recursos.”
  + https://www.ethereum.org
  + https://www.hyperledger.org/projects/fabric
  + https://www.multichain.com
  + https://monax.io
  + http://iota.org
  + https://www.hyperledger.org/projects/iroha

Arquétipo:

BC como um armazenamento seguro de dados para sistemas IoT

Neste arquétipo, BC atua como um banco de dados à prova de violação de entradas de dados ou ponteiros para entradas de dados fora da cadeia.

* Objetivos de melhoria:

melhorar a segurança em termos de integridade

* Problemas técnicos:

protegendo dados do sensor em repouso e registros de eventos ou protegendo índices que apontam para esses dados

* Posição Lógica do BC:

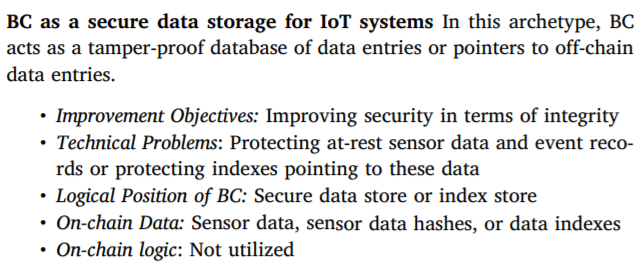
armazenamento de dados seguro ou armazenamento de índice

* Dados na cadeia:

dados do sensor, hashes de dados do sensor ou índices de dados

* Lógica on-chain:

não utilizado



<https://www.filipeflop.com/blog/containers-docker-com-raspberry-pi/>

<https://hackernoon.com/raspberry-pi-cluster-emulation-with-docker-compose-xo3l3tyw>

<https://raspberrypi.stackexchange.com/questions/165/emulation-on-a-linux-pc>

<https://owfs.org/index_php_page_owfs.html>

<https://github.com/lukechilds/dockerpi>

<https://www.boulderes.com/resource-library/building-raspberry-pi-disk-images-with-docker-a-case-study-in-software-automation>

<https://ieeexplore.ieee.org/abstract/document/9042950>

<https://www.computerworld.com/article/3191077/what-is-blockchain-the-complete-guide.html>

pesquisar Hyperledger Fabric na raspberry pi

<https://stackoverflow.com/questions/45800167/hyperledger-fabric-on-raspberry-pi-3>

docker + rasp - Rodrigo

rasp + owfs - Rodrigo

hyperledger fabric - Igor

módulos disponíveis e como utilizar (foco segurança e formato de dados)

ler Niya et al. (2018, <http://dx.doi.org/10.1109/NOMS.2018.8406329>) - Mateus

formato de dados no bc

pesquisa alternativa (plano b): nuvem