UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA
Autorizada pelo Decreto Federal nº 77.496 de 27/04/76
Recredenciamento pelo Decreto n°17.228 de 25/11/2016
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
COORDENAÇÃO DE INICIAÇÃO CIENTÍFICA

XXIX SEMINÁRIO DE INICIAÇÃO CIENTÍFICA DA UEFS
SEMANA NACIONAL DE CIÊNCIA E TECNOLOGIA - 2025

Emulação de um Sistema de Reputação Distribuído Baseado em Blockchain

Silvio Azevedo de Oliveira[1]; Antonio Augusto Teixeira Ribeiro Coutinho[2]
1.Silvio Azevedo de Oliveira, PIBIC/FAPESB, ENGENHARIA DE COMPUTAÇÃO, e-mail: silviozv.oli@gmail.com
2.Antonio Augusto Teixeira Ribeiro Coutinho, DEPARTAMENTO DE TECNOLOGIA, e-mail: acoutinho@uefs.br

PALAVRAS-CHAVE: Emulação; Névoa; Blockchain; Reputação;

INTRODUÇÃO

A Internet das Coisas (IoT) conecta bilhões de dispositivos inteligentes através da Internet (ATZORI; IERA; MORABITO, 2010). Essa quantidade massiva de nós conectados gera desafios significativos relacionados à segurança, confiabilidade e gerenciamento de dados. O conceito de Computação em Névoa (*Fog Computing*) (BONOMI et al., 2012) apresenta-se como uma arquitetura ideal para sistemas IoT que demandam baixa latência e gerenciamento eficaz no processamento distribuído.

Além das vantagens alcançadas com a arquitetura em névoa, torna-se também fundamental estabelecer mecanismos que permitam avaliar a confiabilidade e o comportamento de cada entidade participante. Sistemas de reputação emergem como uma solução promissora para esse desafio, permitindo que dispositivos avaliem a confiança de outros nós com base em interações passadas (JØSANG; ISMAIL; BOYD, 2007). No entanto, abordagens centralizadas de sistemas de reputação apresentam vulnerabilidades inerentes, como pontos únicos de falha, possibilidade de manipulação e falta de transparência. Tecnologias de livro-razão distribuído (*Distributed Ledger Technology* - DLT), como Blockchain, oferecem características para superar essas limitações: descentralização, imutabilidade, transparência e segurança criptográfica (GREVE et al., 2018). O Hyperledger Besu[1] é uma DLT de código aberto que usa protocolos de consenso robustos, como o *Istanbul Byzantine Fault Tolerance* (IBFT) 2.0 e o *Quorum Byzantine Fault Tolerant* (QBFT). Outra tecnologia usada na implementação de redes IoT confiáveis é Tangle da IOTA (POPOV, 2018), cuja premissa consiste na construção de uma estrutura de transações interligadas em grafo acíclico direcionado (DAG). Isso permite escalabilidade e rapidez na inserção de dados.

Este trabalho tem como objetivo desenvolver um protótipo de um sistema de reputação distribuído para IoT usando as tecnologias Besu e Tangle no ambiente de emulação Fogbed (COUTINHO et al., 2023). Os objetivos específicos incluem: (1) implementar o ambiente para criação de blockchain privada; (2) desenvolver um manual de replicabilidade para a reprodução dos experimentos; (3) analisar o desempenho da arquitetura distribuída proposta; e (4) implementar e otimizar o algoritmo *K-Means* para agregação de avaliações de reputação.

[1]https://www.hyperledger.org/use/besu

MATERIAL E MÉTODOS

O desenvolvimento deste trabalho foi estruturado em quatro etapas principais. A primeira consistiu na disponibilização do plugin FogBesu, que permite a criação de uma blockchain privada baseada em Hyperledger Besu no emulador Fogbed, uma extensão do Mininet para emulação de ambientes de fog computing e IoT. A arquitetura implementada permite a configuração de uma rede blockchain com múltiplos nós distribuídos em diferentes instâncias virtuais. A integração entre Besu e Tangle (POPOV, 2018) viabilizou a criação de cenários complexos para comunicação dos nós em IoT e modelos de reputação em ambientes de névoa.

Na segunda etapa, foi desenvolvido um manual de execução detalhado para garantir a reprodutibilidade dos experimentos. A verificação da funcionalidade do sistema foi feita através de dados escritos em arquivo CSV, usando o módulo *write-csv*, responsável por registrar dados relacionados à requisição de serviços entre gateways.

Na terceira etapa, foram conduzidos experimentos em uma máquina com processador Intel Core i3-8100 (quatro núcleos, 3.60 GHz), 16 GB RAM, SSD NVMe 256 GB e Ubuntu 20.04. Foram executados três nós da DLT Tangle Hornet e a ponte ZMQ (HINTJENS, 2013), antes dos gateways. O experimento foi realizado com cinco gateways, monitorados por uma hora. Os resultados revelaram uso médio de CPU de 73,62% (desvio padrão 43,27) e uso médio de memória de 7,37% (desvio padrão 0,1).

Na última etapa, foi implementado o algoritmo *K-Means* (AHMED; SERAJ; ISLAM, 2020), método de aprendizado não supervisionado para particionamento de dados em clusters. No sistema de reputação, o *K-Means* agrupa avaliações dos nós com maiores credibilidades. A implementação em Java utilizou o método *K-Means++* (ARTHUR; VASSILVITSKII, 2007), que favorece centroides iniciais bem distribuídos. O processo segue quatro passos: (i) escolha aleatória do primeiro centroide; (ii) cálculo da distância ao quadrado de cada ponto ao centroide mais próximo; (iii) escolha probabilística de novo centroide proporcional à distância calculada; (iv) repetição até selecionar todos os centroides. Os experimentos compararam a implementação em Java com script Python (*ProcessBuilder*), coletando cem amostras de tempo por execução. A análise estatística utilizou intervalo de confiança de 95% com distribuição *t* de *Student* (MONTGOMERY; RUNGER, 2018).

RESULTADOS

Analisando os resultados do consumo de processamento, é possível deduzir que a emulação do sistema de reputação avaliado é predominantemente limitada pela CPU. O consumo médio de 73,62% é alto, e considerando um cenário com cinco gateways, estaria utilizando quase a totalidade dos quatro núcleos disponíveis na máquina utilizada. Esse fato explicaria a limitação da máquina em executar mais do que seis gateways. Os dados de CPU também se mostram voláteis devido ao altíssimo desvio-padrão de 43,27. Esse fator é explicado pela forma de operação do sistema, com picos de maior e menor atividade. Os picos maiores podem estar relacionados aos cálculos de credibilidade e à comunicação com a DLT. O consumo de memória, por outro lado, é estável e previsível, com uma média de 7,37% e desvio-padrão de 0,1. Isso demonstra que o gateway aloca uma quantidade de memória e utiliza de forma consistente, deixando evidente que a memória não é um gargalo para essa arquitetura.

Para cada um dos dois cenários, o executado pelo script em Python e outro pelo *KMeans* em Java, o sistema foi submetido a quinze execuções independentes, coletando-se cem amostras de tempo do algoritmo *K-Means* de cada uma. A análise estatística, baseada nas médias das execuções e nos cálculos do intervalo de confiança, produziu os dados apresentados na Tabela 1, arredondados para quatro casas decimais.

Tabela 1: resultados dos tempos para os dois cenários de execução do algoritmo *K-Means*.

Métrica	**Implementação em Java**	**Script em Python (*ProcessBuilder*)**
Médias das amostras ($\bar{x}$)	0,1847 ms	1076,6506 ms
Desvio-padrão das médias (s)	0,0805 ms	46,2923 ms
Intervalo de confiança (95%)	[0,1401 ms, 0,2293 ms]	[1051,0147 ms, 1102,2864 ms]
Largura do intervalo	0,0892 ms	51,2717 ms

O primeiro ponto de destaque entre os dados é a grande diferença entre os tempos médios de execução. A implementação em Java se mostrou ser, aproximadamente, 5830 vezes mais rápida do que o script em Python. Para um sistema de reputação distribuída, que necessita de operações de baixa latência, a abordagem em Java é mais adequada. Em relação aos intervalos de confiança e ao desvio-padrão, é possível analisar a consistência dos resultados. A baixa largura do intervalo da implementação em Java indica uma variabilidade extremamente baixa, resultando em um desempenho mais estável e previsível. Por outro lado, o script em Python mostrou uma variabilidade 575 vezes maior, indicando que o seu tempo de execução é mais suscetível a flutuações e seu comportamento é menos previsível. Essa grande diferença pode ser justificada pela utilização da ferramenta *ProcessBuilder* no script Python. Essa ferramenta cria um novo processo no sistema operacional a cada chamada, sendo uma operação computacionalmente cara. A comunicação entre os dados do sistema no geral e do script ocorre por meio de mecanismos de comunicação, como *stdin/stdout*, o que é inerentemente mais lento do que simples chamadas de métodos em uma aplicação Java. Em suma, os resultados validam de forma conclusiva que, para a arquitetura desenvolvida (SANTOS; COUTINHO, 2024), a implementação nativa em Java é a escolha viável. A abordagem via o script em Python, mesmo que funcional, introduz uma sobrecarga de tempo e uma instabilidade de desempenho que a tornam inadequada para os requisitos necessários.

CONSIDERAÇÕES FINAIS

O trabalho atingiu com sucesso os objetivos relacionados ao uso da arquitetura de emulação na execução de DLTs utilizando modelos de reputação. Como trabalhos futuros, pretende-se utilizar a infraestrutura desenvolvida para implementar e avaliar diferentes modelos de reputação em cenários de névoa, bem como permitir a autenticação dos nós de IoT através do plugin FogBesu. A criação do manual de execução serve como base para outros casos de teste desejados, tornando a pesquisa acessível e verificável para a comunidade científica. A análise de desempenho

identificou o alto consumo de CPU pelos gateways, o que pode ser uma limitação no momento da execução. A mudança do script em Python pela implementação do *K-Means* em Java resolveu um grande problema de latência, deixando o sistema milhares de vezes mais rápido em relação ao cálculo da credibilidade dos nós, como também aumentou a previsibilidade e estabilidade do seu comportamento. Essa otimização valida a arquitetura do sistema de reputação como uma abordagem viável para um ecossistema IoT. A solução desenvolvida representa uma contribuição para a área de pesquisa em sistemas de névoa e IoT, fornecendo uma ferramenta que pode ser utilizada para a experimentação e a avaliação de diferentes modelos DLT em um ambiente de Névoa/IoT controlado e realista.

REFERÊNCIAS

AHMED, M.; SERAJ, R.; ISLAM, S. M. S. (2020). The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8), 1295.

ARTHUR, D.; VASSILVITSKII, S. (2007). k-means++: The advantages of careful seeding. *In*: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (pp. 1027-1035).

ATZORI, L.; IERA, A.; MORABITO, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15), 2787-2805.

BONOMI, F.; MILITO, R.; ZHU, J.; ADDEPALLI, S. (2012). Fog computing and its role in the internet of things. *In*: Proceedings of the first edition of the MCC workshop on Mobile cloud computing, 13-16.

COUTINHO, A.; DAMASCENO, U. D. C.; MASCARENHAS, E. D. S.; SANTOS, A. C. D. S.; DA SILVA, J. E.; GREVE, F. (2023). Rapid-Prototyping of Integrated Edge/Fog and DLT/Blockchain Systems with Fogbed. *In*: ICC 2023-IEEE International Conference on Communications. IEEE, 2023. p. 622-627.

GREVE, F. G.; SAMPAIO, L. S.; ABIJAUDE, J. A.; COUTINHO, A. C.; VALCY, Í. V.; QUEIROZ, S. Q. (2018). Blockchain e a Revolução do Consenso sob Demanda. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)-Minicursos.

HINTJENS, P. (2013). ZeroMQ: Messaging for Many Applications. O'Reilly Media.

JØSANG, A.; ISMAIL, R.; BOYD, C. (2007). A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2), 618-644.

MONTGOMERY, D. C.; RUNGER, G. C. (2018). Applied statistics and probability for engineers (7th ed.). *Wiley*.

POPOV, S. (2018). The Tangle. White Paper, IOTA Foundation.

SANTOS, A. C. S.; COUTINHO, A. A. T. R. (2024). Sistema de Reputação Distribuído com Blockchain para Ecossistemas IoT. Trabalho de Conclusão de Curso, Universidade Estadual de Feira de Santana, Feira de Santana.