Journal of Applied Informatics and Computing (JAIC)
Vol.9, No.3, June 2025, pp. 571~582
e-ISSN: 2548-6861

Evaluation of Scalability and Resilience of Hyperledger Fabric in Blockchain Implementation for Diploma Management

Cahyani Pebriyanti [1]*, Galura Muhammad Suranegara [2]*
* Departement of Telecommunication system, Universitas Pendidikan Indonesia
cahyanifbri@upi.edu [1], galurams@upi.edu [2]

Article Info

Article history:

Received 2025-03-10
Revised 2025-04-09
Accepted 2025-04-14

Keyword:

Blockchain,
Hyperledger Fabric,
Performance,
Evaluation.

ABSTRACT

This research aims to evaluate the performance of a Hyperledger Fabric-based blockchain system implemented for digital diploma management. The system is tested using the Caliper benchmark tool in various network and scalability scenarios, including normal conditions (baseline), network delay of 50ms, 100ms, 200ms, and 500ms; packet loss of 1%, 5%, 10%, and 15%; bandwidth limitation of 5 Mbps; high transaction load (scalability standard and scalability optimized); and extreme conditions in the form of Byzantine attacks with malicious nodes of 10%, 30%, and 50%. The evaluation was conducted using four key metrics: transaction success rate, failure rate, average transaction latency, and throughput (TPS). The system recorded high performance under normal network conditions with a success rate of 99.8%, latency of 0.89 seconds, and throughput of 9.7 TPS. Network disruptions such as delay, packet loss, and bandwidth limitation had only a minor impact, with the success rate remaining above 99% and latency gradually increasing. High load in the scalability scenario caused latency to increase to 27.21 seconds and failure rate to rise, but improved significantly after chaincode optimization. Meanwhile, the Byzantine scenario showed a drastic drop in performance with the success rate decreasing to 12.83% and the failure rate increasing to 87.17%. These results show that the Hyperledger Fabric-based digital diploma management system is resilient to common network disruptions and reliable at medium scale, but still requires strengthening the consensus mechanism to deal with extreme conditions and maintain reliability in environments that are not fully trusted.

I. INTRODUCTION

The first generation of blockchain technology began with the introduction of Bitcoin by Satoshi Nakamoto in 2008 [1], which proposes a peer-to-peer-based digital currency system without the involvement of a centralized authority. The aim is to reduce transaction costs in the conventional financial system [2]. This innovation brings the concept of blockchain to the forefront and encourages its use in cryptocurrency-based digital payment systems [3]. Over time, the application of blockchain has expanded to areas such as financial transactions, healthcare, data collection, asset management, and supply chain [4]. One promising application is the management of important documents, including education certificates.

A diploma issued by an educational institution is an important document that proves a person's graduation and eligibility to continue their studies to a higher level or enter the workforce [5]. However, printed certificates face significant challenges regarding their legitimacy and security, such as the risk of forgery, physical damage, and error-prone [6]. Diploma forgery is a serious problem in many countries [7]. In the UK, for example, the high number of counterfeiting institutions led the University of Wales to shut down its diploma validation system and prompted the resignation of the university's registrar [8]. In the United States, a scandal unfolded in which more than 7,600 fake nursing diplomas were issued, allowing thousands of nurses to work illegally [9].

In Indonesia, falsification of academic credentials violates professional ethics and is a criminal offense under Article 263 of the Criminal Code as well as Article 69 of Law No. 20 of 2003 on the National Education System. Case No. 20/Pid.B/2024/PN Tmg, in which SZ, a legal practitioner, was found guilty of using fake diplomas, exposed loopholes in the credential verification system, allowing unqualified individuals to enter the legal profession. According to the Ministry of Education, in the past five years there have been more than 1,200 cases of falsification of academic documents [10]. These cases show how vulnerable the conventional diploma system is to forgery.

One solution that can be applied to overcome this problem is Blockchain. Blockchain is a decentralized ledger that is capable of recording transactions between two parties without the need for an intermediary [11]. Blockchain has the properties of decentralization, immutability, transparency, and distributed consensus [12]. Each transaction is stored in blocks that are interconnected and distributed across the network. Each block contains a hash of the previous block and its own hash, which ensures that all data is stored chronologically and validly [13].

There has been a lot of research related to the use of blockchain for verification and issuance of academic certificates to overcome the problem of diploma forgery and efficiency in the academic validation process. Research [14] proposed a blockchain-based digital certificate system using Hyperledger Fabric. The system aims to prevent certificate forgery and reduce the cost and time required for certificate issuance. Research [15] researched the application of Hyperledger Fabric in higher education institutions to improve security, transparency, and efficiency in the remote assessment process, with results showing that permissioned blockchains are more suitable than cryptocurrency-based solutions. Research [16] discussed a distributed data sharing system at a university using Hyperledger Fabric and IPFS, with results showing improved security and efficiency in sharing public and private data.

The implementation of blockchain in digital diploma certification systems faces technical challenges related to resilience and scalability. Resilience refers to resistance to network disruptions and non-ideal operating conditions, especially in institutions located in regions with limited infrastructure. In addition, the threat of malicious (Byzantine) nodes is also a serious concern, as they can disrupt the consensus process and drastically reduce transaction success rates. Scalability, on the other hand, encompasses the system's ability to handle transaction spikes without performance degradation. Some previous research has addressed this aspect in Hyperledger Fabric.

Research [17] evaluated the scalability of blockchain networks by varying the number of transactions, workers, channels, block size, and block interval. The results show that increasing channels and workers can increase latency, while larger block intervals can decrease latency without reducing throughput. Research [18] analyzed Hyperledger Fabric by considering block size, transaction arrival rate, and number of users. Throughput increases with block size, but decreases when the system is overloaded; transaction failures also increase as the number of users increases. Research [19] conducted a thorough analysis of Hyperledger Fabric performance, including aspects of network architecture, hardware configuration, transaction parameters, workload, and network conditions. It was found that the main bottleneck occurs in the transaction validation phase, and performance degrades drastically if latency exceeds 100 ms or bandwidth is below 50 Mbps. Meanwhile, Research [20] examined fault tolerance and Byzantine attacks, showing that disturbances such as jitter and packet loss have a non-linear impact on latency, with packet loss above 10% causing significant latency spikes.

While the application of blockchain, particularly Hyperledger Fabric, in digital diploma certification systems has been widely researched, most of the previous studies have been limited to design aspects or testing in simulated environments. This research broadens the scope by developing a more comprehensive and contextualized performance evaluation scenario, including gradual variation of network conditions (50-500 ms delay, 1-15% packet loss, and bandwidth limitation), scalability testing, and resistance to Byzantine attacks. In addition, the tests were conducted in the context of a real implementation at an educational institution, which has its own geographical and operational challenges. The main contribution of this research is to present a more comprehensive and realistic evaluation model of Hyperledger Fabric performance and provide technical insights into the impact of network conditions and transaction load on system success, latency, and throughput, including the effectiveness of chain code optimization in load scenarios.

II. METHODS

This research uses a type of empirical testing research (experiment) with development using the Software Development Life Cycles (SDLC) method in the creation and development of a blockchain-based Higher Education Institution Certificate system. SDLC has three development models, namely Waterfall, Spiral, and Incremental/Iterative. Of the three models, this research chose the Waterfall model because of its structured and sequential approach, in accordance with the research objectives which require systematic analysis at each stage of system development and testing. By going through five stages-Requirement, Design, Implementation, Testing, and Evaluation [21], this model ensures that Hyperledger Fabric performance testing can be done in a step-by-step and well-documented manner, thus providing more accurate and measurable results under various network conditions. The stages are depicted in Figure 1.

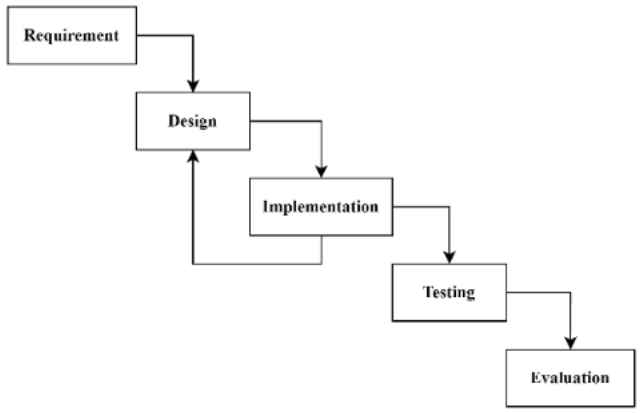

Figure 1. Flowchart of the Waterfall Model

A. Requirement

Requirement is the earliest stage to prepare requirements or research needs such as Software and Hardware used to support research. In Table 1 are the Software and Hardware that need to be prepared.

TABLE I
SOFTWARE AND HARDWARE

Software	Hardware
Windows 11 Operating System Ubuntu 22.03.04; Virtual Box Hyperladger Fabric v.2.2 Node js, Golang Docker, Docker Compose Caliper	Processor: AMD A4-9125 Radeon R3 2 CPUs 2.3GHz RAM: 8GB SSD: 240GB Network: Wi-Fi Hotspot (Telkomsel, 4G LTE)

B. Design

This research evaluates the effectiveness and efficiency of Hyperledger Fabric in a blockchain-based diploma management system. The network architecture consists of orderer nodes, peer nodes, certificate authority (CA), and channels as communication paths. Orderer nodes manage consensus, peer nodes execute and validate transactions, while certificate authority ensures security by managing digital certificates for authentication and authorization of entities in the network.

The network used in testing consists of 2 peer nodes, 1 orderer node, and 1 certificate authority (CA). The system runs in a Docker container environment on the Ubuntu 22.03.04 operating system.

In addition to the basic configuration, this research includes various test scenarios to evaluate the performance of the blockchain network under different conditions. The scenarios include the addition of latency (50ms, 100ms, 200ms, 500ms), packet loss (1%, 5%, 10%, 15%), and bandwidth limitation (5 Mbps), which are simulated using Traffic Control (tc) in Linux to represent unstable network conditions. In addition, a scalability scenario was also conducted to test the system's ability to handle an increasing number of transactions per second (TPS) in one batch workload. To test the system's resilience to potential attacks or failures, a Byzantine node scenario was applied, in which some nodes act invalidly by either sending incorrect transactions or not responding at all with 10%, 30%, and 50% disruption levels.

C. Implementation

In the implementation stage, the system that has been designed will be built and integrated. Figure 2 is the flow of the system implementation stage

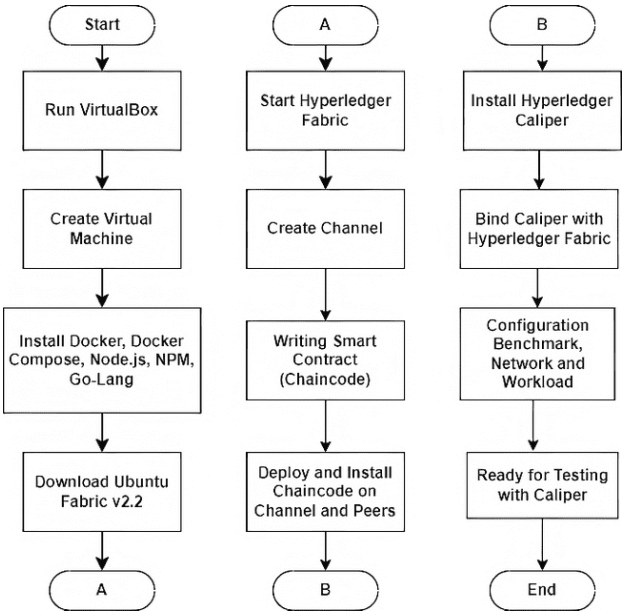

Figure 2. Implementation Flowchart

1) Prarequisites: At the initial stage, the process starts by running VirtualBox and creating a Virtual Machine (VM). Next, the prerequisite installations are Docker, Docker Compose, Golang, Node.js, and NPM. Hyperledger Fabric installation is performed using the curl -sSL https://bit.ly/2ysbOFE | bash -s -- 2.2.0 1.4.9 command, which automatically downloads and installs Fabric docker images and Fabric Certificate Authority. In addition, this command also installs fabric-samples, which contains a test network with two organizations (Org1 and Org2) and one orderer.

2) Start Hyperledger Fabric: To start and activate the Fabric network using the provided test network, run the command./network.sh up. This command initializes and activates the Fabric network consisting of two organizations and one orderer, and generates the necessary certificates and private keys.

3) Create Channel: By running the command time ./network.sh createChannel -ca -c mychannel -s couchdb, a new channel named mychannel will be created on the previously configured network. In a Hyperledger Fabric network, a channel serves as a communication path that regulates transaction access, so that only certain nodes can see relevant transactions. Once the channel is successfully

created, peer 1 and peer 2 will join it. During the channel creation process, the creation time will be recorded, which aims to analyze the difference in channel creation duration in various test scenarios. Figure 3 shows the process of successfully creating a channel.

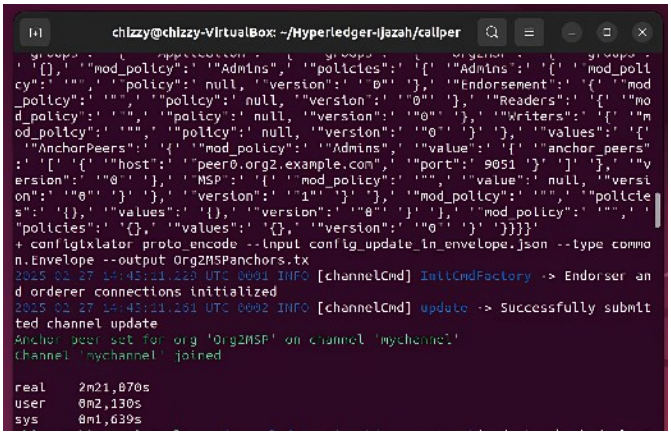

Figure 3. Channel Creation

4) Write and Deploy Chaincode: The chaincode used is written in the Go programming language and is designed to manage diploma data in a blockchain-based system using the Hyperledger Fabric platform. Diploma data is represented in a Diploma structure, which includes attributes such as DiplomaID, StudentName, University, Degree, and GraduationYear. To deploy the chaincode, use the command ./network.sh deployCC -ccn diploma -ccp ../Diploma/chaincode-go -ccl go. Figure 4 is a successfully deployed Hyperledger network.

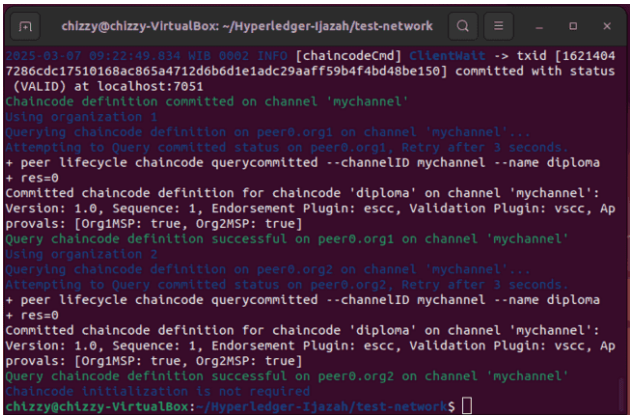

Figure 4. Deploy Successfully

5) Caliper Configuration: After successfully building the Blockchain network, continue with running Caliper. Caliper-CLI is installed via NPM (Node Package Manager) with version 0.4.2. Caliper is then connected to the SUT to be tested, in this case the Hyperledger Fabric, via the npx caliper bind --caliper-bind-sut fabric:2.2 command. Next, we prepare 3 configuration files, namely network, benchmark, and workload configurations.

6) Network configuration: This configuration file is used to connect Caliper to the Hyperledger Fabric network, a channel definition named mychannel that runs a smart contract with a diploma ID, and an Org1MSP organization that has a user identity with a certificate and private key. In addition, there is a connection profile that connects Caliper to the Fabric network. Figure 5 is the network configuration script.

```yaml
name: Caliper test
version: "2.0.0"

caliper:
  blockchain: fabric
  sutOptions:
    mutualTls: true

channels:
  - channelName: mychannel
    contracts:
      - id: diploma

organizations:
  - mspid: Org1MSP
    identities:
      certificates:
        - name: "User1"
          clientPrivateKey:
            path: "../test-network/organizations/peerOrganizations
/org1.example.com/users/User1@org1.example.com/msp/keystore/
cc9a7dba724ac4a3247aacce6ca9f033b0eac96a13aea682635e12a356540042_sk"
          clientSignedCert:
            path: "../test-network/organizations/peerOrganizations
/org1.example.com/users/User1@org1.example.com/msp/signcerts/cert.pem"
    connectionProfile:
      path: "../test-network/organizations/peerOrganizations/
org1.example.com/connection-org1.yaml"
      discover: true
```

Figure 5. Network Configuration Script

7) Benchmark Configuration: This file defines the performance testing of a blockchain-based diploma management system using Caliper, with one worker executing transactions. Transactions are executed for 30 seconds with a fixed-rate of 10 transactions per second (TPS). Figure 6 is the benchmark configuration script.

```yaml
test:
  name: "Diploma Management Benchmark"
  description: "Benchmark untuk manajemen ijazah"
  workers:
    number: 1
  rounds:
    - label: createDiploma
      description: "Membuat ijazah"
      txDuration: 30
      rateControl:
        type: fixed-rate
        opts:
          tps: 10
      workload:
        module: workload/createDiploma.js
        arguments:
          diplomaCount: 100
```

Figure 6. Benchmark Configuration Script

8) Workload Configuration: This script defines a workload for testing diploma creation on Hyperledger Fabric using Caliper. Each transaction generates a unique diploma ID and calls the createDiploma function in the diploma chaincode with parameters such as student name, university, degree, and graduation year. Figure 7 is the Workload configuration script.

```
'use strict';
const { WorkloadModuleBase } = require('@hyperledger/caliper-core');
class CreateDiplomaWorkload extends WorkloadModuleBase {
  async initializeWorkloadModule(workerIndex, totalWorkers, roundIndex, roundArguments,
sutAdapter, sutContext) {
    // Menyimpan jumlah diploma yang akan dibuat
    this.diplomaCount = roundArguments.diplomaCount;
    this.sutAdapter = sutAdapter;
  }
  async submitTransaction() {
    // Menghasilkan ID Diploma unik
    const diplomaId = `DIP-${Math.floor(Math.random() * 100000)}`;

    // Mendefinisikan request untuk transaksi
    const request = {
      contractId: 'diploma',  // Nama contract yang sudah dideploy
      contractFunction: 'createDiploma',  // Fungsi chaincode yang dipanggil
      contractArguments: [
        diplomaId,  // ID Diploma yang unik
        'John Doe',  // Nama mahasiswa
        'XYZ University',  // Nama universitas
        'Computer Science',  // Gelar yang diperoleh
        2024,  // Tahun kelulusan
      ],
    };
    // Mengirim transaksi ke jaringan
    await this.sutAdapter.sendRequests(request);
  }
}
module.exports.createWorkloadModule = () => new CreateDiplomaWorkload();

```

Figure 7. Workload Configuration Script

D. Testing

After the Hyperledger Fabric implementation is complete and Hyperledger Caliper has been installed, the next step is to conduct testing to evaluate the performance of the blockchain network. Figure 8 is the flow of testing.

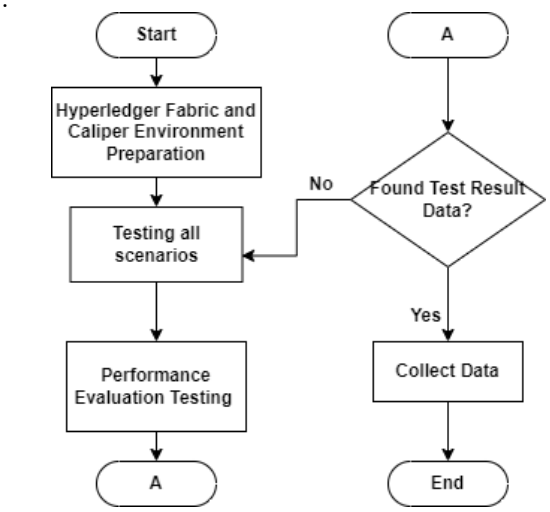

Figure 8. Testing Flow Chart

The flowchart in Figure 8 illustrates the process of testing the performance of Hyperledger Fabric using Hyperledger Caliper. Testing begins with environment preparation, including Fabric and Caliper configuration. Next, the performance of Hyperledger Fabric was evaluated through a variety of testing scenarios. After the test, the system checks whether Caliper successfully displays the data. If not, the test will be repeated, but if results are available, the data will be collected for further analysis before the test ends.

The evaluation was conducted using four key metrics: transaction success rate, failure rate, average transaction latency, and throughput (TPS). Throughput represents the number of successfully processed and confirmed transactions per second during the testing period. Average latency refers to the average time between when a transaction is submitted and when it is confirmed in the blockchain network, as reported by Caliper. The success and failure rates are calculated based on the proportion of successful or failed transactions relative to the total number of submitted transactions, as defined in Equations (1) and (2).

$$\%\ SR = \left|\frac{Number\ of\ Transactions\ Success}{Total\ Transactions}\right| \times 100\% \quad (1)$$

$$\%\ FR = \left|\frac{Number\ of\ Failed\ Transactions}{Total\ Transactions}\right| \times 100\% \quad (2)$$

In this test, several different scenarios were applied including:

1) Scenario 1 Baseline (Normal): In this scenario, Hyperledger Caliper is configured to measure the performance of the blockchain network with predetermined workload parameters as shown in Figure 7. Testing is done with one worker whose job is to send createDiploma transactions at a speed of 10 transactions per second (Transaction Per Second/TPS) for 30 seconds. Testing is carried out under normal network conditions, without any interference.

2) Scenario 2 Delay Addition: In this scenario, tests were conducted to analyze the effect of network delay on Hyperledger Fabric performance by applying delays of 50 ms, 100 ms, 200 ms, and 500 ms using the sudo tc qdisc add dev enp0s3 root netem delay <value>ms command. Figure 9 shows the application of a 50 ms delay using the tc command, which is then verified with the ping command to ensure successful configuration and an increase in response time. Tests with other delay values were performed using the same method.

```
chizzy@chizzy-VirtualBox:~$ sudo tc qdisc add dev enp0s3 root netem delay 50ms
chizzy@chizzy-VirtualBox:~$ sudo tc qdisc show dev enp0s3 root
qdisc netem 8004: root refcnt 2 limit 1000 delay 50ms
chizzy@chizzy-VirtualBox:~$ ping 192.168.222.93
PING 192.168.222.93 (192.168.222.93) 56(84) bytes of data.
64 bytes from 192.168.222.93: icmp_seq=1 ttl=127 time=53.6 ms
64 bytes from 192.168.222.93: icmp_seq=2 ttl=127 time=53.8 ms
64 bytes from 192.168.222.93: icmp_seq=3 ttl=127 time=65.7 ms
^C
--- 192.168.222.93 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2013ms
rtt min/avg/max/mdev = 53.613/57.701/65.733/5.679 ms
chizzy@chizzy-VirtualBox:~$
```

Figure 9. Adding a delay

In addition, to ensure the latency effect is applied thoroughly in an isolated environment, delay is also configured on Docker containers running Hyperledger Fabric components. Figure 10 shows the implementation of a 50ms delay on a Docker container by checking the bridge interface, then adding the delay via sudo tc qdisc add dev <interface-name> root netem delay 50ms, and verified using tc qdisc show which displays the output qdisc netem ... delay 50ms as evidence that the configuration was successfully implemented.

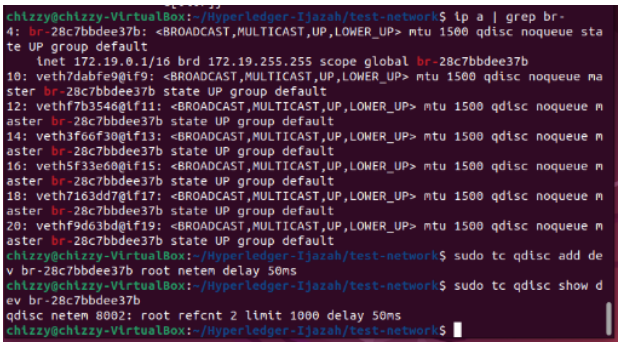

Figure 10. Delay Implementation on Docker Container

Testing using Caliper is done with the same benchmark configuration as in Figure 6 and the same workload as Figure 7, which includes a fixed transaction rate of 10 transactions per second (TPS) for 30 seconds.

3) Scenario 3 Testing with Packet Loss: In this scenario, tests were conducted to analyze the impact of packet loss on the reliability of the blockchain system built using Hyperledger Fabric. Packet loss of 1%, 5%, 10%, and 12% is applied using the sudo tc qdisc add dev enp0s3 root netem loss <value>% command to simulate network quality degradation. Figure 11 shows the application of a packet loss of 10% using the tc command, which is then verified through a ping command to the destination IP, showing that out of 10 packets sent, only 9 are received, in accordance with the applied configuration. Tests with other packet loss values were performed with the same method.

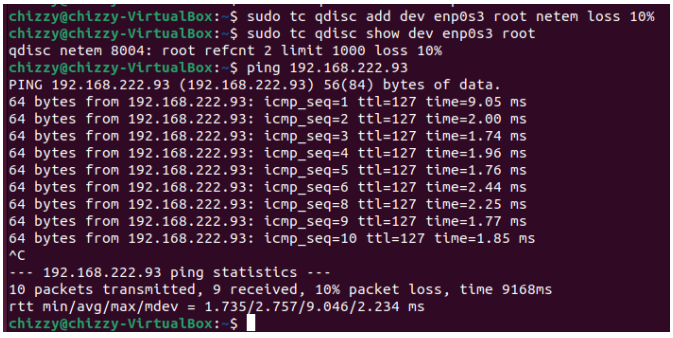

Figure 11. Adding Packet Loss

To ensure the effect of packet loss is also applied to Hyperledger Fabric components running in a container environment, a similar command is run on each Docker container involved in the network using: sudo tc qdisc add dev <container_id> root netem loss <value>%. Figure 12 shows the application of a packet loss of 10% on a Docker container through the bridge interface br-28c7bbdee37b. Verification is done with the tc qdisc show command, which displays a 10% netem loss configuration, indicating that the packet loss simulation was successfully implemented in the container environment.

Figure 12. Proof of Delay Implementation on Docker Containers

Testing using Caliper is done with the same benchmark configuration as in Figure 6 and the same workload as Figure 7, which includes a fixed transaction rate of 10 transactions per second (TPS) for 30 seconds.

4) Scenario 4 Bandwidth Limitation 5 Mbps: In this scenario, tests were conducted to analyze the impact of bandwidth restrictions on the performance of the blockchain system built using Hyperledger Fabric. The restriction is done by limiting the data transfer rate to 5 Mbps using the tc qdisc add dev enp0s3 root tbf rate 5mbit burst 32kbit latency 400ms command. Figure 13 shows the results of verifying the application of this configuration using the TBF algorithm, which is then validated through the speedtest service and shows that bandwidth restrictions are successfully applied according to the configuration.

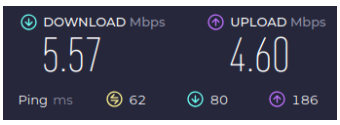

Figure 13. Proof of Bandwidth Deployment

In order for bandwidth limitation to also apply to Hyperledger Fabric components running in containers, the same command is applied to each Docker container using sudo tc qdisc add dev <container_id> root tbf rate 5mbit burst 32kbit latency 400ms. Figure 14 shows the application of the configuration on the network interface in the Docker container through the br-28c7bbdee37b bridge, which was successfully verified with the tc qdisc show command and displays the output tbf rate 5Mbit burst 4Kb lat 400ms, indicating that bandwidth limitation has been successfully implemented in the container environment.

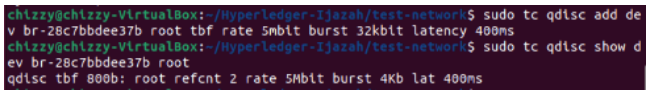

Figure 14. Bandwidth Deployment on Docker Containers

Testing using Caliper is done with the same benchmark configuration as in Figure 6 and the same worklod as Figure 7, which includes a fixed transaction rate of 10 transactions per second (TPS) for 30 seconds.

5) Scenario 5 Scalability with Increased Transaction Load: Testing in this scenario is done by modifying the benchmark configuration in the Caliper folder to evaluate the system's ability to handle increased transaction loads. The changes included increasing the transaction duration (txDuration) from 30 seconds to 40 seconds, as well as increasing the transaction rate per second (TPS) from 10 TPS to 40 TPS. There were no changes to the network configuration; adjustments were only made to the parameters

in the benchmark file. In addition, the tests were conducted under two conditions: first, using the same chaincode as the previous scenarios, and second, using a modified version of the chaincode to reduce latency. Optimizations to the chaincode were made by adding the ability to process ten transactions at once in a single function call (batch transaction), thus reducing the processing overhead per transaction.

6) Scenario 6 Byzantine Node Effect: In this scenario, Byzantine behavior in the Hyperledger Fabric 2.2 network is simulated by modifying the chaincode of a smart contract that manages digital certificates. Since Hyperledger Fabric 2.2 does not have a Byzantine Fault Tolerance (BFT) mechanism by default, this approach is used to see how the system handles nodes that act improperly. Byzantine nodes in this study have three main forms of deviance: (1) altering diploma data before storage, (2) randomly rejecting transactions, and (3) returning manipulated diploma data when accessed. To measure the impact of these Byzantine behaviors, tests were conducted by gradually increasing the probability of deviation: the first scenario had a probability of 10%, the second scenario increased to 30%, and the third scenario reached 50%.

E. Evaluation

After the tests are completed, all results from Hyperledger Caliper will be analyzed to evaluate the performance of the blockchain network. This analysis aims to understand how the system handles various scenarios, including normal conditions, additional delay, packet loss, bandwidth limitation, increased transaction load, and byzantine nodes.

III. RESULTS AND DISCUSSIONS

Tests were conducted on different scenarios that represent various network and system conditions, such as normal (baseline) scenarios, additional delay, packet loss, scalability scenarios and Byzantine attacks. Each scenario was tested five times (Report 1-5) to obtain more representative and reliable results. As a detailed illustration, the complete data is presented in Table II Scenario 1 (Baseline). Meanwhile, the test results in other scenarios are presented in the form of summary averages and grouped by category, in order to facilitate reading and analysis of overall system performance.

TABLE II
COMPLETE RESULTS OF SCENARIO 1 (BASELINE)

Report Number	Success	Fail	Success Rate (%)	Failure Rate (%)	Avg Latency (s)	Throughput (TPS)
1	300	1	99.67	0.33	0.79	10.0
2	301	0	100	0	0.83	9.3
3	301	0	100	0	0.88	9.3
4	300	1	99.67	0.33	0.99	9.9
5	300	1	99.67	0.33	0.97	9.9
Average	300	0.6	99.80	0.20	0.89	9.7

Table II presents the complete results of testing on the baseline scenario, which is the condition of the system running normally without network interference or additional load. The results show that the system has a very stable performance, with an average transaction success rate of 99.80%, an average latency of 0.89 seconds, and a throughput of 9.7 TPS. This optimal performance is achieved because communication between Hyperledger Fabric components (endorser, orderer, and peer) does not experience obstacles, allowing the transaction process to run efficiently.

TABLE III
DELAY CATEGORY

Scenario	Avg Success	Avg Fail	Success Rate (%)	Failure Rate (%)	Avg Latency (s)	Throughput (TPS)
Delay 50ms	299.8	1.2	99.60	0.40	1.04	9.90
Delay 100ms	300.2	0.8	99.73	0.27	1.08	9.56
Delay 200ms	300.8	0.2	99.93	0.07	1.31	9.32
Delay 500ms	299.6	1.2	99.60	0.40	2.47	9.40

Table III summarizes the results of testing system performance with variations in network delay from 50ms to 500ms. Although all scenarios still show a high transaction success rate (above 99.6%), the latency increases as the network delay increases, from 1.04 seconds at 50ms delay to 2.47 seconds at 500ms delay. This increase is due to the increased propagation time between nodes. However, throughput remains relatively stable in the range of 9.32 to 9.90 TPS, this can be attributed to the transaction queuing and batch processing mechanisms in Hyperledger Fabric, which help mitigate the effects of network delay.

TABLE IV
PACKET LOSS CATEGORY

Scenario	Avg Success	Fail	Success Rate (%)	Failure Rate (%)	Avg Latency (s)	Throughput (TPS)
Packet Loss 1 %	299.8	1.2	99.40	0.40	0.876	9.90
Packet Loss 5%	300.4	0.6	99.60	0.20	0.886	9.70
Packet Loss 10%	300.2	0.8	99.74	0.26	0.972	9.90
Packet Loss 15%	300.0	1.0	99.67	0.33	1.33	9.42

Table IV shows the impact of network disruption in the form of packet loss with a packet loss rate of 1% to 15%. In general, the system is able to maintain high performance with a success rate above 99.4%. The average latency increases as packet loss increases, from 0.876 seconds at 1% to 1.33 seconds at 15%. This is due to the need for data retransmission due to packet loss. Despite this, throughput remained stable in the range of 9.42–9.90 TPS, indicating that the system remains efficient under less-than-ideal network conditions

TABLE V
BANDWIDTH LIMITATION

Scenario	Avg Success	Avg Fail	Success Rate (%)	Failure Rate (%)	Avg Latency (s)	Throughput (TPS)
Bandwidth 5 Mbps	300.4	0.6	99.80	0.20	1.752	9.48

Table V illustrates the system performance when the network bandwidth is limited to 5 Mbps. In this condition, the system still shows high performance with a transaction success rate of 99.80% and an average latency of 1.752 seconds. Although there is a slight increase in latency compared to normal conditions, throughput remains stable at 9.48 TPS. These results indicate that moderate bandwidth limitation did not significantly affect the performance of the blockchain system.

TABLE VI
SCALABILITY CATEGORY

Scenario	Avg Success	Avg Fail	Success Rate (%)	Failure Rate (%)	Avg Latency (s)	Throughput (TPS)
Scalability - Standard Chaincode	1097.8	8.2	99.26	0.74	27.21	17.21
Scalability - Optimized Chaincode	957.0	2.6	99.73	0.27	18.09	16.14

Table VI compares the system performance with two types of chaincode execution: standard and optimized. The use of optimized chaincode results in a significant decrease in latency, from 27.21 seconds to 18.09 seconds, with an increase in success rate from 99.26% to 99.73%. Although the throughput of both configurations was relatively equivalent (17.21 TPS vs. 16.14 TPS), optimization of the chaincode was shown to improve transaction processing efficiency through a batching approach and reduction of smart contract execution overhead.

TABLE VII
BYZANTINE ATTACK CATEGORY

Scenario	Avg Success	Avg Fail	Success Rate (%)	Failure Rate (%)	Avg Latency (s)	Throughput (TPS)
Byzantine 10%	200.2	100.8	66.51	33.49	1.30	9.54
Byzantine 30%	83.8	217.2	27.84	72.16	1.98	9.78
Byzantine 50%	38.6	262.4	12.83	87.17	1.98	9.74

Table VII illustrates the impact of Byzantine attacks on the system with the proportion of malicious nodes at 10%, 30%, and 50%. As the number of malicious nodes increases, the transaction success rate drops dramatically from 66.51% at 10% to only 12.83% at 50%. This decrease shows the difficulty of the system in achieving a valid consensus when the majority of nodes cannot be trusted. Although the throughput remained in the range of 9.5-9.7 TPS, the quality and validity of transactions decreased sharply. This finding confirms that the system is only able to tolerate Byzantine attacks up to a certain level before significantly losing functionality.

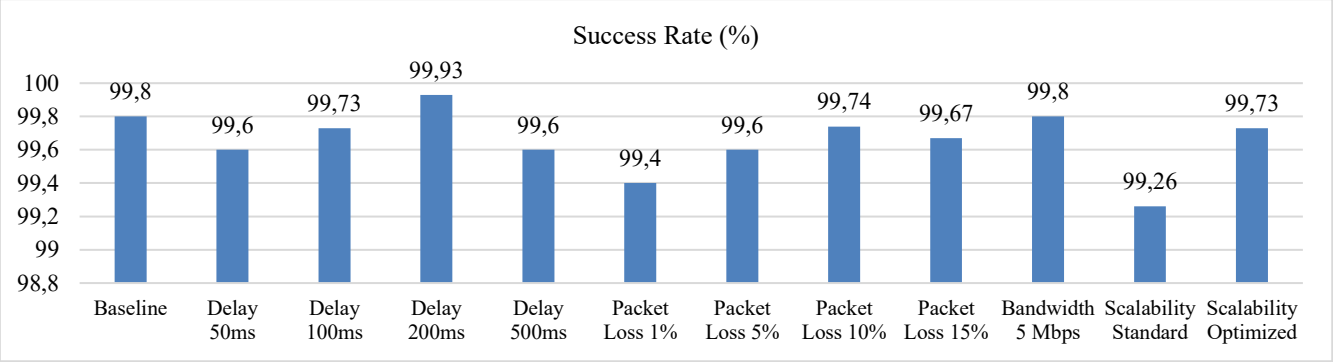

Figure 15. Success Rate Bar Chart (%)

Figure 15 shows the success rate of transactions in various scenarios testing the network and scalability of the Hyperledger Fabric-based system. Overall, the system performed very reliably, with all scenarios recording success rates above 99%. Under normal (baseline) conditions, the success rate reached 99.8%. Interestingly, scenarios with added delay showed high stability, with the highest value recorded at 200ms delay of 99.93%. Scenarios with packet loss of up to 15% have a relatively small impact, where the success rate remains in the range of 99.4%-99.74%. The only significant decrease occurred in the scalability scenario with standard chaincode, which amounted to 99.26%. However, after optimizing the chaincode, the success rate increased to 99.73%. These results confirm that the system is able to maintain high reliability even when faced with non-ideal network conditions, and shows significant performance improvement through the optimization approach.

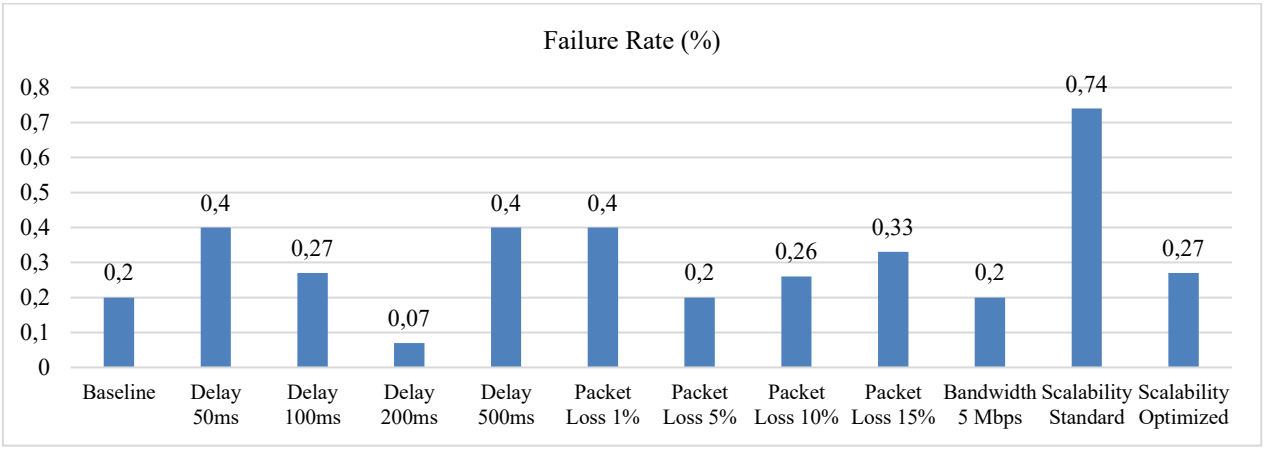

Figure. 16. Failure Rate Bar Chart (%)

Figure 16 presents the failure rate of transactions in various scenarios testing the network and scalability of the Hyperledger Fabric-based system. In general, the failure rate is very low, with most scenarios recording values below 0.5%. In the baseline condition, the failure rate was recorded at only 0.2%. The scenario with a delay of 50ms and a packet loss of 1% showed the highest failure rate outside of the scalability scenario, at 0.4% each. In contrast, a delay of 200ms resulted in the lowest failure rate of 0.07%, indicating that increasing latency does not always have a direct impact on increasing transaction failures. Failures started to increase gradually at 10% (0.26%) and 15% (0.33%) packet loss, indicating that larger packet losses started to have an effect on transaction success.

However, the most significant increase occurred in the scalability scenario with standard chaincode, with the failure rate reaching 0.74%. After optimization, this rate decreased drastically to 0.27%.

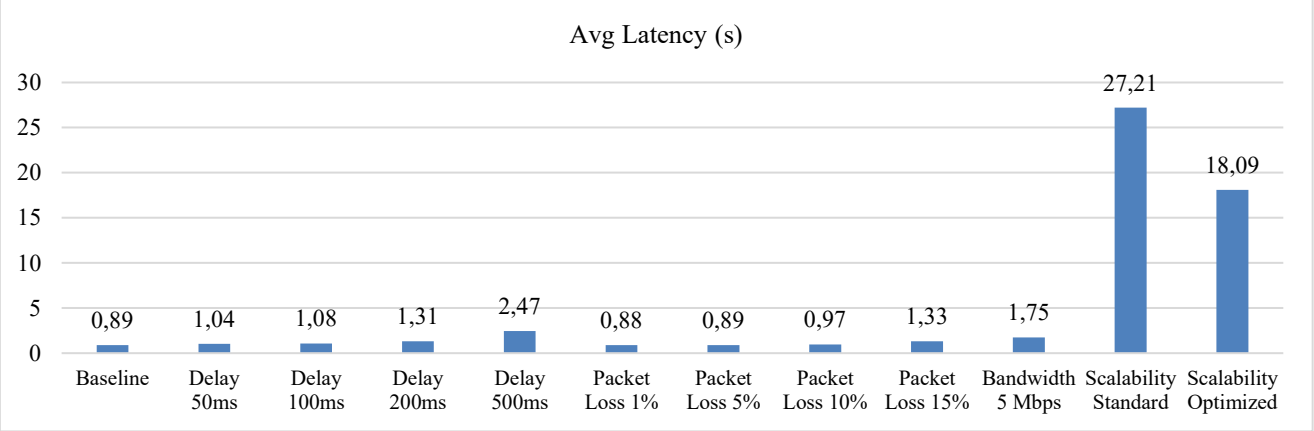

Figure. 17. Latency Bar Chart (s)

Figure 17 shows the average latency of transactions under various network condition and scalability scenarios. In general, latency tends to increase as network quality deteriorates. In the baseline condition, the latency is recorded at 0.89 seconds. When a delay of 50ms to 500ms is added, latency shows a gradual increasing trend, from 1.04 seconds to reach 2.47 seconds at a delay of 500ms. A similar pattern is seen in the packet loss scenario, where the greater the percentage of packet loss, the latency tends to increase, although with a less drastic increase-for example from 0.88 seconds at 1% packet loss to 1.33 seconds at 15% packet loss. Bandwidth limitation also resulted in an increase in latency to 1.75 seconds. The most significant increase occurred in the scalability scenario, where a high transaction load without optimization caused a drastic latency spike of up to 27.21 seconds. After optimization, latency was successfully reduced to 18.09 seconds, but still far above the baseline. These results show that deteriorating network conditions do have an impact on increasing latency, but the main challenge in keeping latency low is the system's ability to manage transaction loads efficiently.

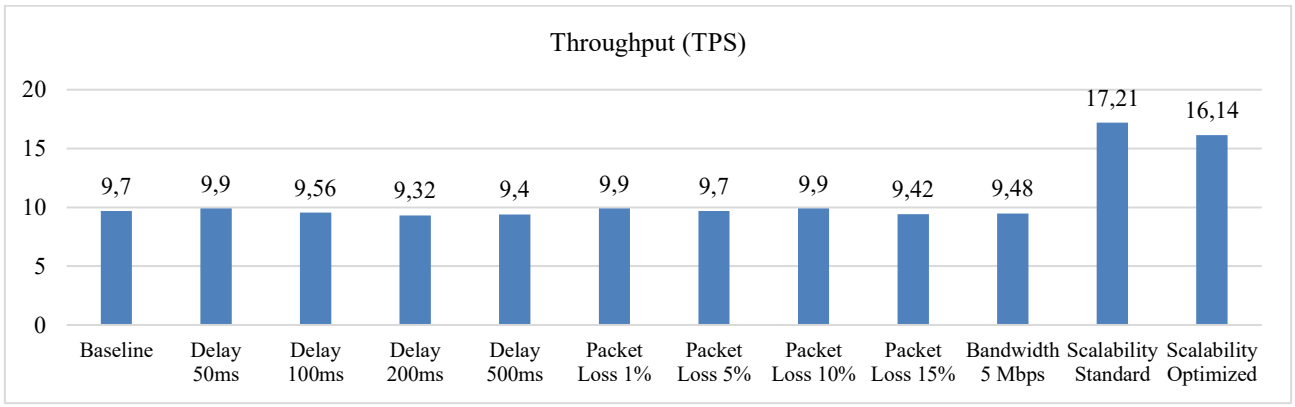

Figure 18. Throughput Bar Chart (TPS)

Figure 18 shows the throughput value or the number of transactions per second (TPS) in various scenarios of network conditions and scalability testing. In general, throughput performance under various network conditions such as adding delay, packet loss, and bandwidth limitation does not show significant changes. The throughput value in the baseline condition is 9.7 TPS, and remains stable in the range of 9.3-9.9 TPS in scenarios with delays up to 500ms, as well as at packet loss up to 15%. This shows that the Hyperledger Fabric system is quite resistant to mild to moderate network disruptions. Limiting the bandwidth to 5 Mbps also only decreases the throughput slightly to 9.48 TPS. In contrast, significant changes were seen in the scalability scenario. When the transaction load is increased (scalability standard), the throughput jumps dramatically to 17.21 TPS, which then slightly decreases to 16.14 TPS after the chaincode is optimized (scalability optimized).

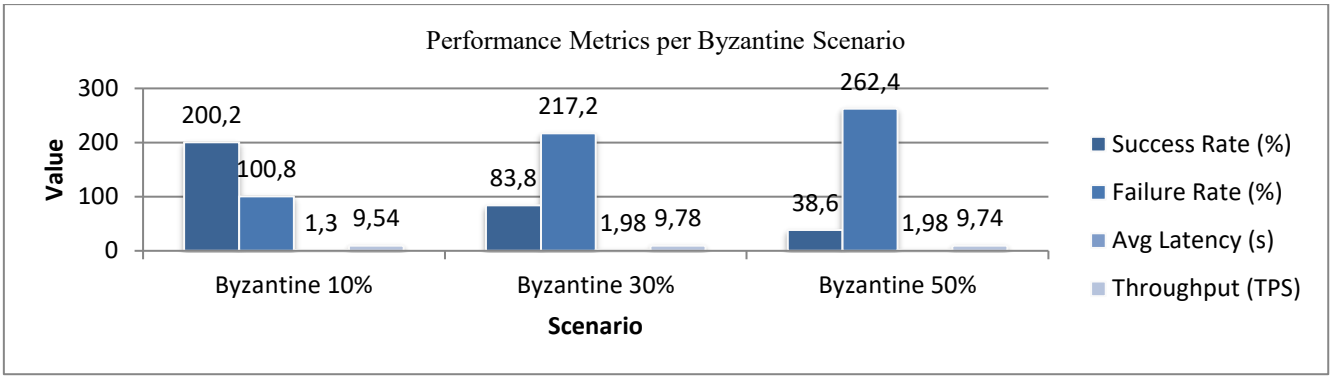

Figure 19. Evidence of 5 Mbps Bandwidth Restriction

Figure 19 is presented separately because the metric values in the Byzantine scenario tend to be extreme and much higher than the other scenarios. If combined in one visualization, the scale of the graph will distort the representation of the performance of other scenarios, making the pattern of performance variation in normal scenarios difficult to read. In the graph, it can be seen that the success rate has decreased dramatically, for example, in the 50% Byzantine scenario it only reaches an average of about 12.83%. The failure rate increases sharply to 87.17%, which reflects the large number of transactions that are not successfully processed. This is due to the presence of malicious (Byzantine) nodes that send invalid information in the consensus process, making it difficult for the system to reach an agreement and failing to process transactions properly.

The results of this research show that the Hyperledger Fabric-based system has a fairly high resilience to various network disruptions such as delay, packet loss, and bandwidth restrictions. In these scenarios, the system is still able to maintain a high success rate and relatively stable throughput.

However, transaction latency increases significantly with network disruption. This can be seen in the delay and packet loss tests, where the transaction delay increases as the communication process between nodes in the network becomes slower. Although this increase is still within acceptable limits, it shows that the system's performance in terms of response time is quite sensitive to network quality.

Challenges became apparent when the system was tested under scalability scenarios and Byzantine attacks. High transaction loads in the scalability scenario led to increased latency and failure rates, especially when using the standard chaincode configuration. However, after optimization, the system showed significant performance improvement in terms of latency, transaction success, and throughput stability. This confirms that chaincode execution efficiency plays an important role in maintaining system performance at scale.

Meanwhile, in the Byzantine attack scenario, the system showed a significant drop in performance. Although the throughput appears stable, the transaction success rate drops dramatically, and the failure rate increases sharply. This indicates that the system is unable to process transactions validly when there are nodes that behave improperly.

On the one hand, a decrease in the success rate may indicate that the system is successfully rejecting transactions from malicious nodes, indicating a limited form of resilience to manipulation. But on the other hand, a high number of failed transactions also indicates that the system becomes unusable in an untrusted environment.

This finding is in line with some previous studies, such as those conducted by [19] which found that network disruptions such as latency and bandwidth greatly affect the performance of Hyperledger Fabric, and [20] which shows that packet loss above 10% can cause significant latency spikes. In addition, the findings on the scalability scenario are in line with [17] [18], where an increase in the number of transactions and users leads to transaction failures and overall performance degradation

On the other hand, the Byzantine scenario in this study underscores the importance of further research on the integration of BFT mechanisms into Hyperledger Fabric. As in the research of BFT mechanism into Hyperledger Fabric. As in the research [22] which concluded that consensuses such as kafka, raft no Kafka (used in the old Fabric) could not withstand Byzantine failure.

IV. CONCLUSION

This research shows that Hyperledger Fabric has good resilience to network disruptions such as delay, packet loss, and bandwidth throttling, with success rates and throughput remaining high despite increased latency. Chaincode optimization proved effective in improving performance in scalability scenarios, extending the findings of previous studies related to system efficiency at high loads. However, in the Byzantine scenario, the system performance degrades drastically, confirming the limitations of the current consensus in dealing with malicious nodes. These findings indicate the need for integration of Byzantine Fault Tolerant (BFT) mechanisms in the Hyperledger

Fabric and encourage further studies to explore system resilience in untrusted environments.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System".

[2] Y. Ucbas, A. Eleyan, M. Hammoudeh, dan M. Alohaly, "Performance and Scalability Analysis of Ethereum and Hyperledger Fabric," *IEEE Access*, vol. 11, hlm. 67156–67167, 2023, doi: 10.1109/ACCESS.2023.3291618.

[3] G. Tripathi, M. A. Ahad, dan G. Casalino, "A comprehensive review of blockchain technology: Underlying principles and historical background with future challenges," *Decis. Anal. J.*, vol. 9, hlm. 100344, Des 2023, doi: 10.1016/j.dajour.2023.100344.

[4] F. Anwar, B. U. I. Khan, L. B. M. Kiah, N. A. Abdullah, dan K. W. Goh, "A Comprehensive Insight into Blockchain Technology: Past Development, Present Impact and Future Considerations," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 11, 2022, doi: 10.14569/IJACSA.2022.01311101.

[5] SS.Rajkumari, Kapa Poojitha, V.Jhansi, Chedde Meghana, S.Rajiya Banu, dan R.Ayesha Siddikha, "Educational Certificate Verification System Using Block Chain," *Int. J. Eng. Technol. Manag. Sci.*, vol. 8, no. 2, hlm. 165–175, 2024, doi: 10.46647/ijetms.2024.v08i02.022.

[6] A. Rustemi, F. Dalipi, V. Atanasovski, dan A. Risteski, "A Systematic Literature Review on Blockchain-Based Systems for Academic Certificate Verification," *IEEE Access*, vol. 11, hlm. 64679–64696, 2023, doi: 10.1109/ACCESS.2023.3289598.

[7] S. R, handra P. G., A. K., dan P. S., "Blockchain based certificate validation system," *Int. Res. J. Mod. Eng. Technol. Sci.*, vol. 4, no. 7, hlm. 3816, Jul 2022.

[8] R. Abdelmagid, M. Abdelsalam, dan F. K. Alsheref, "A Blockchain Framework for Academic Certificates Authentication," *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 7, 2024, doi: 10.14569/IJACSA.2024.0150729.

[9] "Southern District of Florida | Fraudulent Nursing Diploma Scheme Leads to Federal Charges Against 25 Defendants | United States Department of Justice." Diakses: 2 Maret 2025. [Daring]. Tersedia pada: https://www.justice.gov/usao-sdfl/pr/fraudulent-nursing-diploma-scheme-leads-federal-charges-against-25-defendants

[10] D. J. Sistyawan, R. V. Neonbeni, M. Rizal, A. C. D. C. Kusuma, dan H. Husain, "Fake Diplomas, Real Consequences: Legal and Ethical Challenges in the Legal Profession," *Constitutionale*, vol. 5, no. 2, hlm. 137–156, Feb 2025, doi: 10.25041/constitutionale.v5i2.3608.

[11] M. Dabbagh, M. Sookhak, dan N. S. Safa, "The Evolution of Blockchain: A Bibliometric Study," *IEEE Access*, vol. 7, hlm. 19212–19221, 2019, doi: 10.1109/ACCESS.2019.2895646.

[12] A. S. Rajasekaran, M. Azees, dan F. Al-Turjman, "A comprehensive survey on blockchain technology," *Sustain. Energy Technol. Assess.*, vol. 52, hlm. 102039, Agu 2022, doi: 10.1016/j.seta.2022.102039.

[13] V. Jayadev, N. Moradpoor, dan A. Petrovski, "Assessing the Performance of Ethereum and Hyperledger Fabric Under DDoS Attacks for Cyber-Physical Systems," dalam *Proceedings of the 19th International Conference on Availability, Reliability and Security*, Vienna Austria: ACM, Jul 2024, hlm. 1–6. doi: 10.1145/3664476.3670927.

[14] A. El-Dorry, M. Reda, S. A. El Khalek, S. El-Din Mohamed, R. Mohamed, dan A. Nabil, "Egyptian Universities Digital Certificate Verification Model Using Blockchain," dalam *Proceedings of the 9th International Conference on Software and Information Engineering*, dalam ICSIE '20. New York, NY, USA: Association for Computing Machinery, Jan 2021, hlm. 79–83. doi: 10.1145/3436829.3436864.

[15] P. Dias, H. Gonçalves, F. Silva, J. Duque, J. Martins, dan A. Godinho, "Blockchain Technologies: A scrutiny into Hyperledger Fabric for Higher Educational Institutions"," *Procedia Comput. Sci.*, vol. 237, hlm. 213–220, 2024, doi: 10.1016/j.procs.2024.05.098.

[16] K. Marhane, F. Taif, dan A. Namir, "Secure Sharing of university Data Using Hyperledger Fabric and IPFS system," *Procedia Comput. Sci.*, vol. 224, hlm. 163–168, 2023, doi: 10.1016/j.procs.2023.09.024.

[17] L. Sadath, MSc, Mca, D. Mehrotra, dan A. Kumar, "Scalability Performance Analysis of Blockchain Using Hierarchical Model in Healthcare," *Blockchain Healthc. Today*, vol. 7, no. 1, Apr 2024, doi: 10.30953/bhty.v7.295.

[18] C. Wickboldt, "Benchmarking a Blockchain-based Certification Storage System".

[19] T. Guggenberger, J. Sedlmeir, G. Fridgen, dan A. Luckow, "An in-depth investigation of the performance characteristics of Hyperledger Fabric," *Comput. Ind. Eng.*, vol. 173, hlm. 108716, Nov 2022, doi: 10.1016/j.cie.2022.108716.

[20] T. Pang, Z. Ye, Z. Zhang, dan C. Jin, "Fault Tolerance Testing and Tuning for Consortium Blockchain," *Blockchain Res. Appl.*, hlm. 100267, Jan 2025, doi: 10.1016/j.bcra.2024.100267.

[21] A. A. Wahid, "Analisis Metode Waterfall Untuk Pengembangan Sistem Informasi," *J. Ilmu-Ilmu Inform. Dan Manaj. STMIK*, 2020.

[22] A. Barger, Y. Manevich, H. Meir, dan Y. Tock, "A Byzantine Fault-Tolerant Consensus Library for Hyperledger Fabric," 14 Juli 2021, *arXiv*: arXiv:2107.06922. doi: 10.48550/arXiv.2107.06922.