Blockchain-enabled Real-time SLA Monitoring for Cloud-hosted Services

Kashif Mehboob Khan[a], Junaid Arshad[b], Waheed Iqbal[c], Sidrah Abdullah[a], Hassan Zaib[c]

[a]*Department of Software Engineering, NED University of Engineering & Technology, Karachi, Pakistan*
[b]*School of Computing and Digital Technology, Birmingham City University, Birmingham, UK*
[c]*Punjab University College of Information Technology, University of the Punjab, Lahore, Pakistan*

Abstract

Cloud computing is an important technology for businesses and individual users to obtain computing resources over the Internet on-demand and flexibly. Although cloud computing has been adopted across diverse applications, the owners of time-and-performance critical applications require cloud service providers' guarantees about their services, such as availability and response times. Service Level Agreements (SLAs) are a mechanism to communicate and enforce such guarantees typically represented as service level objectives (SLOs), and financial penalties are imposed on SLO violations. Due to delays and inaccuracies caused by manual processing, an automatic method to periodically verify SLA terms in a transparent and trustworthy manner is fundamental to effective SLA monitoring, leading to the acceptance and credibility of such service to the customers of cloud services. This paper presents a blockchain-based distributed infrastructure that leverages fundamental blockchain properties to achieve immutable and trustworthy SLA monitoring within cloud services. The paper carries out an in-depth empirical investigation for the scalability of the proposed system in order to address the challenge of transparently enforcing real-time monitoring of cloud-hosted services leveraging blockchain technology. This will enable all the stakeholders to enforce accurate execution of SLA without any imprecisions and delays by maintaining an immutable ledger publicly across blockchain network. The experimentation takes into consideration several attributes of blockchain which are critical in achieving optimum performance. The paper also investigates key characteristics of these factors and their impact to the behaviour of the system for further scaling it up under various cases for increased service utilization.

Keywords: Permissioned Blockchain, SLO Monitoring, Smart Contract, Scalability, Web Service Monitoring

1. Introduction

Cloud computing has emerged as an important technology for businesses and individual users to obtain computing resources over the Internet. The main features of cloud computing, including pay-as-you-go, dynamic resource provisioning, high availability, and scalability, attracted the customers to migrate their services from traditional in-house data centers and computing facilities to remote cloud data centers. However, the owners of time-and-performance critical applications require some guarantees from cloud service providers about their services. To address customer concerns, cloud providers offer service level agreements (SLAs).

A typical SLA documents a set of service level objectives (SLOs) in terms of different quality of service (QoS) metrics and mentions financial penalties on SLO violations for the cloud provider. Nowadays, a typical SLA offered by the cloud providers guarantees their services' availability and performance [1]. For example, Amazon Web Services (AWS), a leading cloud service provider, offers 99.5% uptime guarantees for its Relational Database Service (RDS) and reduces 10% services charges in the customer bill in case of violations as a penalty. However, RDS customers still have to submit compensation requests manually. Most customers hosting services and solutions on the cloud platforms ensure the QoS for their end-users by relying on the cloud provider guarantees. Therefore, it is important for customers to automatically and periodically verify the SLA terms in a transparent and trustworthy method. An independent third party can help to enforce the agreed SLA terms. However, the third party requires to ensure transparency and trustworthiness for both parties.

Automatic SLA monitoring, verification, and enforcement are challenging for both parties (cloud providers and customers) to avoid trust deficits. Therein, blockchain brings new opportunities to address the challenges involved in automating SLA monitoring and enforcement. A Blockchain system exemplifies distributed ledger technologies, enabling a digital transaction's data to be stored in a chained block where each participant stores transaction records in these connected blocks. The transactions are included with the consensus of verifying nodes, also known as *miners*. A blockchain-based system has inherent decentralization properties, transparency, trustless-trust, immutability, and traceability since it allows each participating node to have equal opportunity to influence the ledger. The transparency and immutability are enforced by enabling the nodes to view and maintain the ledger, which can only be altered with other mining nodes' consensus. Some initial work has been done in the blockchain for SLAs, such as a blockchain-based framework for the negotiation of cloud services [2], implementation of SLAs in supply chain management system [3], validation of SLA violations [4], payment of the penalties in case of SLA violations [5, 6], the integrity of SLA [7] and secure monitoring of SLA [8]. However, further work is required to investigate challenges to achieve end to end SLA monitoring and enforcement in a trustworthy manner. In this respect, this paper aims to address this challenge through the design and development of a blockchain-based system to monitor and enforce SLA terms.

Specifically, we propose and evaluate an independent third party service for automatic monitoring and enforcement of SLA terms, precisely web application response time, agreed between application owners and cloud providers for ensuring transparency and trustworthiness for both parties. Our proposed solution is based on a private blockchain system that creates an immutable history of records that monitors and evaluates the level of SLA enforcement between customers and service providers. The proposed solution is evaluated for multiple SLA settings with a varying number of web application endpoints required to ensure specific response time SLO requirements. Our experimental evaluation investigates the effects of blockchain's different parameter settings, including block size, transaction throughput, and block generation rate, for optimizing the available infrastructure for running the proposed solution and avoid any wastage of computing resources. The proposed solution can monitor web applications' response time running on the cloud to ensure transparency and trustworthiness for application owners and cloud providers. The main contributions of this paper include:

i. Propose a blockchain-based automatic SLA monitoring system for ensuring transparency and trustworthiness between cloud providers and consumers. We leverage decentralised characteristics of blockchain technology to achieve fault tolerance and therefore achieve a resilient SLA monitoring infrastructure.
ii. A verifiable approach to SLA monitoring which can facilitate trustworthy SLA enforcement. We leverage blockchain's inherent capabilities to achieve an immutable record of web service monitoring which can be helpful for complete audit data trial and log violations against SLA in the future.
iii. Evaluate the proposed system for satisfying SLA, specifically for web application response time, for a varying number of endpoints.
iv. Analyse the impact of different blockchain system parameters to avoid wastage of computing resources to monitor and enforce SLA requirements.

The rest of the paper is organized as follows. Background about SLA and blockchain is presented in Section 2. We discuss the related work in Section3, highlighting state of the art within SLA monitoring. The proposed blockchain-based SLA monitoring system is explained in Section 4 along with its implementation details. Experimental setup and performance analysis of the proposed system are provided in Section 5 and Section 6, respectively. Finally, the conclusion and future work are presented in Section 7.

2. Background

2.1. SLA/WS monitoring

A Service Level Agreement (SLA) is a contract between a service consumer and a service provider that identifies what services will be provided, the goals that will be met, and penalties if expected QoS metrics are not met by the service provider [9]. It also defines the expected level of services that service providers must provide the customer. Every SLA between the SP and customer has a list of Service Level Objectives (SLOs) containing definitions and measurable values of QoS metrics. SLO guarantees that an SLA parameter will be provided in a specified time, and it will adhere to QoS metrics [10]. Every SLA has a lifecycle that it must adhere to. However, the terminologies may vary across different SLAs [11]. There are typically 6 stages in the lifecycle, which are described below:

1. **Discover service provider:** The customer identifies a service provider that provides the required services.
2. **Define SLA:** Both parties negotiate and try to reach acceptable definitions regarding the service level objectives.
3. **Establish agreement:** The terms and services are defined and deployed in this phase. The customer is able to access and utilize the deployed services.
4. **Monitor SLA violation:** The provided services are monitored by both the parties independently to assess compliance with the agreed SLA and potential violations.
5. **Terminate SLA:** The SLA is terminated in case no violation of service level agreement has been detected.
6. **Enforce penalties for SLA violation:** If the specified requirements are not met by service provider, penalties for SLA violation are enforced Such policies are envisaged to have been negotiated prior to service provision.

The general working of service level agreement management process is depicted in Figure 1. This flow chart demonstrates the SLA management process emphasizing the SLA monitoring phase which is the focus of this paper. There are several technical requirements in an SLA, which include the responsibilities of customer and service providers, procedures, pricing and discount policies, service description and description of QoS metrics, and reporting.

2.2. Blockchain and underpinning concepts

Blockchain is a distributed ledger technology that aids decentralized, distributed computing in a trustless environment. Blockchain has witnessed adoption across diverse domains including finance, e-voting [12], and public sector [13]. The attention attracted by blockchain is primarily due to its most popular application, i.e., Bitcoin [14], which effectively seeks to conduct financial transactions in a peer-to-peer manner without the support of the conventional banking system. Although Bitcoin was established in 2009, the concepts and technologies which underpin blockchain in general and Bitcoin, in particular, have evolved over the last few decades. For instance, the concept of anonymous transactions that can be traced back to the sender was introduced by David Chaum in 1983 [15] and today serves as one of the elementary concepts within Bitcoin.

With respect to the ledger, a *transaction* represents the fundamental concept within the blockchain. A transaction in the blockchain is a piece of information that moves something of value (a digital token which may represent a unit of currency, a vote, etc.) from one public address (belonging to the sender) to the receiver's address. Therefore, a transaction saves and tracks the state of the blockchain over a period of time. These transactions become part of a blockchain forever through blocks that move them into the chain. A *block* is primarily a collection of transactions that are integrated and organized in such a way that each block computes and keeps its own *blockhash* (using the individual hashes of all the transactions as its source) in the block along with the blockhash of its preceding block. In this way, a chain of blocks is generated, which grows with time. Since these blocks are connected through their hashes (computed through the transactions within that particular block), this data structure makes the records of blockchain immutable where a slight change in a single transaction would produce an entirely new hash resulting in a mismatch of this hash with the neighboring block. This prevents any suspicious block from being accepted by the blockchain network and therefore mitigates against illegitimate tampering of blockchain state. Fig 2 demonstrates this linkage between different blocks to achieve a tamper-resistant ledger.

All the nodes of a typical blockchain network store an identical copy of blockchain locally, which is frequently synchronized with the main blockchain (also known as *consensus blockchain*). Each new block of a blockchain is accepted and added by its peers through a process known as *mining*. The process of mining is essential in developing consensus among participating nodes and can take up different forms depending upon the type of application. Proof of work is the most common consensus algorithm in blockchain applications owever, other variants such as Proof of Stake have also emerged. Proof of Work algorithm relies heavily on the computational capabilities of the mining hardware to solve a non-trivial mathematical problem.

2.2.1. Public vs. private blockchain

From the perspective of nodes' participation, a blockchain can be divided into two broad categories; public and private blockchains. *Public blockchain* [16] adopts a public model for participation and, therefore, may be joined by any node without any restriction. Such networks of

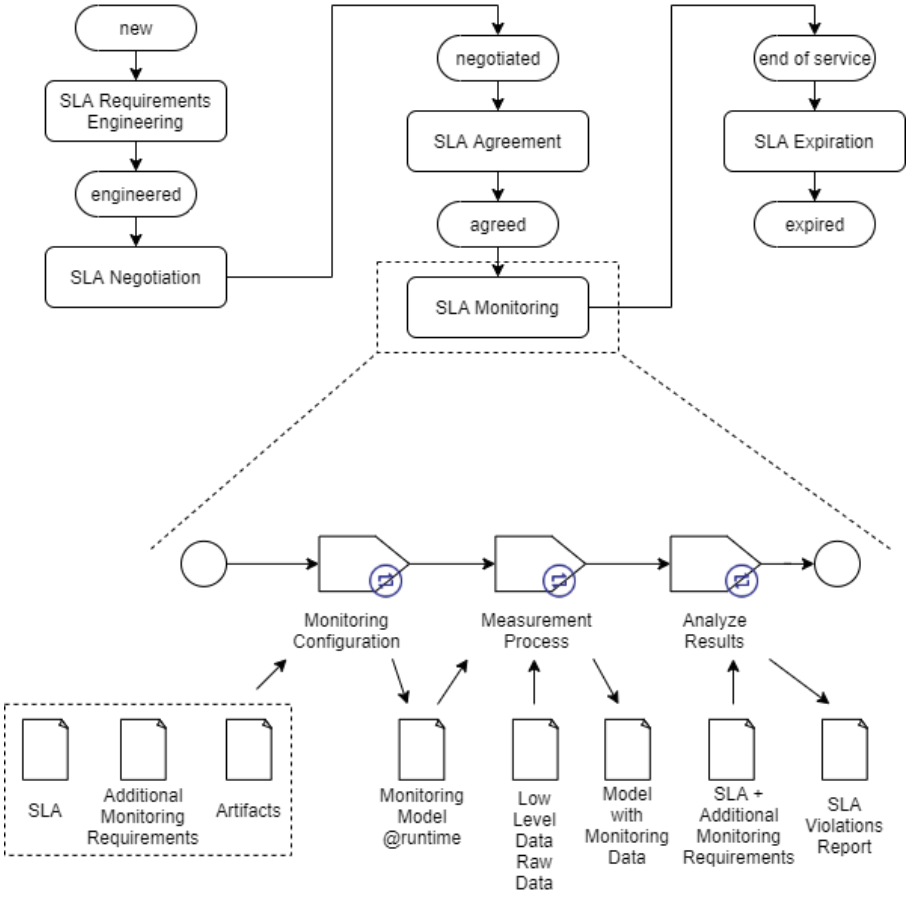


Figure 1: Service Level Agreement Management

blockchain do not require any permission for a user to join or participate in the network. As mentioned earlier, it is a permissionless open-ended blockchain, and that is why the network size is usually bigger than the permissioned blockchain. *Private blockchain*, also called *permissioned blockchain*, on the other hand, is a more controlled form of blockchain which is not publicly accessible. In a private blockchain [17], nodes must seek permission to join the network. Such networks usually require an authenticated node to perform according to a predefined role in the system.

3. Related Work

Recently, blockchain is used in the cloud, edge, fog, and IoT services and solutions [18]. For example, Zhang et al. [19] proposed a solution to use blockchain to improve computation ability for edge computing facilities. Rehman et al. [20] proposed a secure mechanism for IoT devices connected over the cloud through edge nodes using blockchain. Cao et al. [21] proposed a reliable system for the eHealth system to protect electronic health records for authorized and illegal access and modifications using Ethereum blockchain. Savi et al. [22] present initial investigations of using a blockchain-based decentralized system for resource allocation for fog computing infrastructure. Blockchain is also used for automated processes in different domains. For example, in [2], the authors presented a blockchain-based solution for the negotiation process for the provisioning of the cloud resources being provided by any cloud service provider. The proposed solution offers tamper-proof automated negotiation terms dynamically. Blockchain-enabled e-voting systems automatically ensure the security and reliability of the system [23, 12].

There have been several efforts to use blockchain to ensure SLA in different applications as studied by Gai et al. [24]. For example, [3] et al. proposed and evaluated a blockchain integrated logistics and supply chain management using IoT devices. Their proposed system provides a trustful and transparent solution to track deliveries of high value across different organizations. It ensures SLAs for different objectives, including on-time deliveries, appropriate packaging, and asset damages during the transition. Zhou et al. [25] proposed a solution to satisfy SLA using game theory and smart contracts. The authors proposed a witness model to ensure the credibility of the information to record on a public blockchain to detect and report SLA violations. Neidhardt et al. [26] presented SLA monitoring using blockchain to ensure customer trust in provider services. Taha et al. [27] presented an approach to validate the compliance of SLA offered by cloud providers and automatically compensate the customers for SLA violations

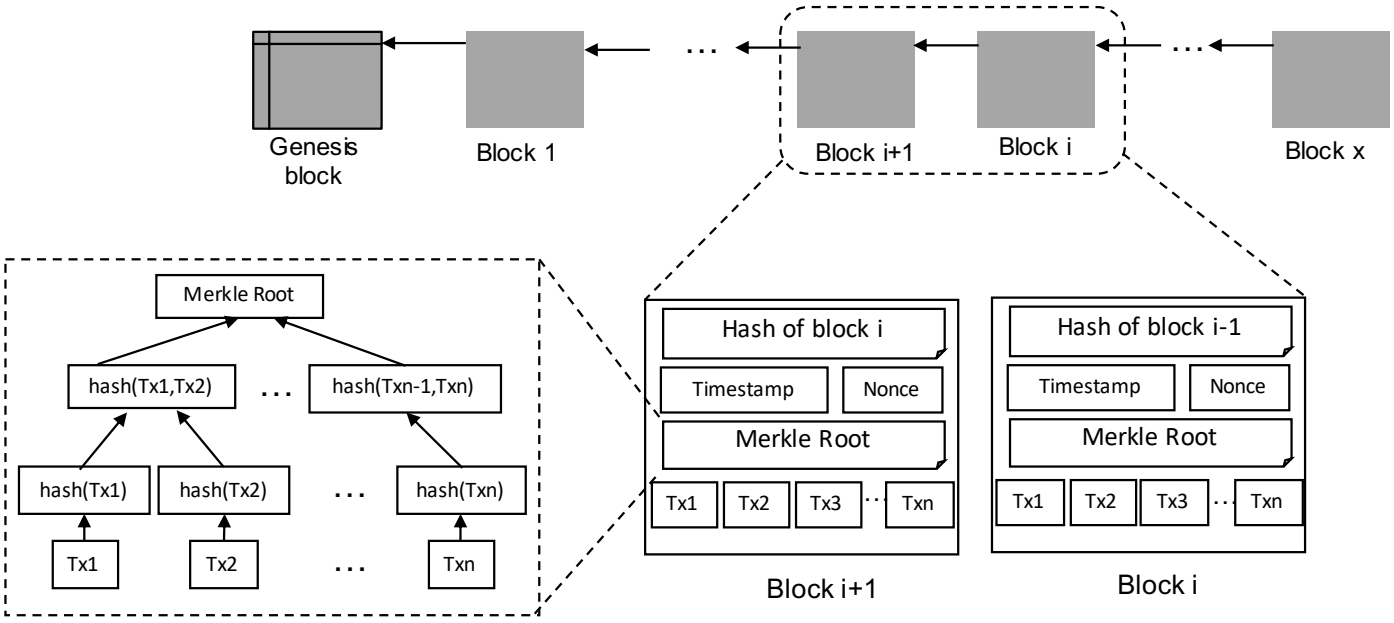


Figure 2: An in-depth view of conventional blockchain

using smart contracts and the Ethereum blockchain system.

Another recent work by Scheid et al. [6, 5] proposed an automated solution based on blockchain and smart contracts to dynamically pay the cost of the penalties to the customers due to SLA violations observed in the services offered by the providers. The authors experimented and showed the SLA violations and penalties payment for a web server response time objectives. The solution is built on the Ethereum blockchain system and does not address the performance impact of hosting multiple smart contracts and monitoring overhead for multiple customers. Nguyen et al. [28] explored the implementation of SLA based blockchain which aims to address the issue of security of the agreements between a user and a service provider in the domain of tourism. The services are defined in detail in SLAs which are executed to enhance the reliability of the system. For this purpose, these SLAs run on architecture based on blockchain that not only monitors the entire process, but also minimizes the interaction between people The SLAs are monitored on the parameters of satisfaction, rule-abiding rates and cost.

There has been some work that uses oracle [29] to use off-chain services for enforcement of SLA. For example, a recent work by Taghavi et al. [30] presented multi-agent monitoring for cloud service SLA for the Ethereum blockchain system using a single oracle for monitoring the SLA constraints and objectives. Uriarte et al. [31] proposed SLA monitoring and penalty enforcement using the public blockchain platform Ethereum. For imposing penalties, the solution used smart contracts and exploited oracles to use off-chain services and data. Scheid et al. [6] also use oracle as a monitoring service to ensure SLA compliance. Ma et al. [32] provide a decentralized solution for verification and disputation for consumers and providers using oracles. Their proposed solution ensures the reliability of oracle for avoiding adding any malicious information to the blockchain.

In addition to using blockchain as a ledger, smart contracts are increasingly used to develop autonomous applications as highlighted by [33]. The authors in [34] explored the use of smart contracts in the field of blockchain for the purpose of implementing effectiveness in service-oriented computing. Further, This study emphasizes on reuse and increase in cost-effectiveness. Daniel and Guida [34] presented a study on the use of smart contracts from an SLA perspective in the paradigm of blockchain. They have also presented with the challenges that might arise in using smart contracts, such as cost awareness, interoperability and standardization, and cost awareness [34]. Smart Contracts in blockchain can also be used as a trusted component to maintaining the integrity of SLA provided by the service provider, as proposed by Wonjiga et al. [7]. SLA verification approach in cloud-based on blockchain technology is made possible to ensure trust between the involved parties. The authors stated that data integrity is a common issue in storage systems, and there have been several data integrity failures. So, this research has enabled tenants to verify the integrity of the data, and providers need to verify the claim [7]. Wonjiga et al. [8, 35] has proposed detailed infrastructure for the monitoring of the security of service level agreements in IaaS clouds [8, 35].

There has been an introduction of SLA in the complex Internet of Things (IoT) domain. Alzubaidi et al. [36, 37] presented the implementation of blockchain SLA in the environment of IoT. IoT devices continuously exchange data in the environment. SLA plays a vital role in the context of IoT applications as it enables concerned parties to conform

to the agreed QoS. Traditional SLA techniques are found to be inefficient for the complex IoT domain and applications. The presented conceptual framework focuses on the awareness of the IoT ecosystem, transparency, auditability and minimum human intervention [36]. Hang and Kim [38] proposed and evaluated an SLA based economy sharing service using Hyperledger Fabric. The system focuses more on the breach of the contract between the concerned parties. In case there is a breach of contract, the monetary compensation is applied automatically, and all the concerned entities are aware of this breach. The framework has been designed in a way to transform business solutions, improve efficiency, and bring automation in different economic situations [38].

Efficient resource provisioning is another significant challenge in cloud computing as it is critical to maximising utilisation and an lead to under or over provisioning. In this context, Ghobaei-Arani et al. [39, 40, 41] and [**?**] have presented recent efforts to achieve effective methods for efficient resource provisioning for cloud-based environments whilst minimising SLA violations. As with SLA monitoring discussed earlier, research community has investigated leveraging blockchain technology to aid resource provisioning in cloud environments. Xing [42] presents one of the most recent efforts in this respect, focusing on complexity due to the use of IoT systems.

The state of the art uses blockchain for various reasons, including SLA monitoring and compliance, but none of the work focused on building a service to enable consumers and providers to monitor web application response time requirements in a transparent and trustful environment. In this work, we have presented and evaluated the third party service using blockchain to address this issue. We also perform performance analysis for using different blockchain parameters to optimize computing resources required to run the proposed solution. Table 1 summarizes the exiting state of the art uses Blockchain for SLA.

4. A Blockchain-based SLA Monitoring System

In this section, we present details of the proposed blockchain-based SLA monitoring system, including the system architecture and our implementation testbed using the Multichain blockchain.

4.1. System architecture

Figure 4 presents the overall architecture of our proposed system. The overall monitoring architecture consists of a customer(s), a cloud service provider, and a monitoring authority. A customer is a user who requires the service provider's service and can be an automated service/program acting on behalf of a human user. The monitoring authority is a trusted third party that runs one or more instances of the monitoring service, each tasked with monitoring compliance of SLA for individual objectives. As presented in Figure 4, an SLA is generated through initial interaction between a customer and the service provider, which is envisaged to include specific SLOs such as availability, response time, and Round Trip Time (RTT). Upon the SLA generation, the monitoring authority monitors the compliance of service endpoints with the agreed SLOs reporting any violations using specific interfaces.

Within our system, a private blockchain is used to create an immutable history of records that monitors and evaluates the level of SLA enforcement between customers and service providers. Every endpoint has its own wallet address in the blockchain network, including monitoring service. Transactions are sent in real-time from the wallet address of the monitoring service to the end point's wallet addresses. All these transactions are encrypted using the OpenSSL library (through https) and authenticated by digital signatures generated in the coin base transaction. Blockchain network raises the system's overall performance by making it scalable through its decentralized network, where the mined transaction upon becoming the part of the consensus blockchain cannot be tampered. While logging the response time of endpoints, the monitoring service also checks the internet quality every time. Any degradation in the quality of the internet may not impact the observed response time of endpoints. If the internet is in an unresponsive state, its throughput is not recorded and compared with its SLO.

Figure 3 describes the interaction of entities through a sequence diagram. Here, the customer process activates upon receiving endpoints' data and registering their accounts on blockchain through their respective wallet addresses. Upon successful registration, SLOs (Service Level Objectives) are defined between customer and its service provider to formally develop a mutually decided service level agreement. This SLO based SLA is then provided to monitoring service to evaluate endpoints' performance benchmarks in accordance with the agreement between customer and service provider and maintain its immutable copies of records on blockchain based distributed ledger through its decentralized private network. The data logging for endpoints evaluation is frequently synchronized to continuously monitor the performance and enforcement of SLA.

This approach is envisaged to be very useful for a transparent evaluation and ranking of a service provider for business gain. In order to ensure the integrity of results, all the data becomes part of the records in real time, powered by blockchain technology. The system also takes into account the stability of the Internet and does not grade a service provider against any endpoints if the internet connection is not stable. Nowadays, most of the cloud service providers (such as Microsoft Azure and others) offer and get engaged with the customer through a minimum level of service agreement which is promised to fulfil by the vendor. Our system does have the tendency to build a strong platform for tracking and monitoring such kind of business model where the independent and instantaneous

Table 1: Comparison of related work uses Blockchain for SLA

References	Blockchain type / Platform	Utilized Techniques	Performance Metrics	Data set or type	Advantages	Disadvantages
Scheid et al. [5]	Permissionless	Network Function Virtualization	Not included/addressed	Not included/addressed	Automatic enforcement of the payments	No evaluation of the proposed work
Scheid et al. [6]	Permissionless	Blockchain-based SLA and Resource Description Framework	response time	Public Ethereum addresses	Dependency of billing handling removed	Monitoring solution only reports SC violations
Zhou et al. [25]	Permissionless	Nash equilibrium principle and unbiased random sortition algorithm	Complexity of the interface	Not included/addressed	Performs well in terms of feasibility	Unautonomous SLA
Neidhardt et al. [26]	Permissionless	Blockchain based SLA	Not addressed/included	Not included/addressed	Transparency of the billing process	No evaluation has been carried out
Nguyen et al. [28]	Permissioned	Blockchain based SLA	Satisfaction, rule abiding rates and cost	Data from Tourist Management Agency	Tracking of SLA violations	Security of the tracked data is not considered and no evaluation has been carried out for the select performance metrics
Alzubaidi et al. [43]	Permissioned	Accuracy Diagnostics	Latency and Success/Fail rates	e database provided by HLF	Performs well in terms of latency	Interoperability issue between Hyperledger Fabric and MVCC
Wonjiga et al. [7]	Permissioned	Blockchain Ledger	Time, verification and overhead	Users' data	Independent verification	No verification if the generated proof is lost by the user
Hang et al. [38]	Permissioned	Asymmetric cryptography	Throughput and latency	Wallet API	Multi-user collaboration and process automation	Security and privacy issues not addressed
Taghavi et al. [30]	Permissionless	Stackelberg differential game	Quality, price and capacity	Users' data	Less computing resources are reserved	Assumptions of perfect knowledge on players part
Taha et al. [27]	Permissionless	Decentralized monitoring over Ethereum	Availability, percentage of processed requests and secure-cookies	Customer data	Customer compensation in case of security breaches	Not focused on authentication management
Urirate et al. [31]	Permissioned and permissionless	Dynamic SLA management	Not addressed/included	Automates the SLA lifecycle and brings transparency	Reduced latency and low overhead	No evaluation of the framework was carried out

enforcement of agreement through independent evaluation of secured and immutable data is of utmost importance.

4.2. Implementation

Figure 5 presents the blockchain-based testbed implementation of the system described in Figure 4 to achieve a rigorous empirical evaluation of the proposed monitoring system. The testbed consists of three units working in coordination under their roles, i.e., Monitoring Server Unit (MSU), Administration Unit, and Monitoring Service Node. Monitoring Server Unit is responsible for hosting and running the monitoring service, which periodically checks, captures, and monitors the data related to SLO against each endpoint. SLO related information is shared with the monitoring server unit through the administration unit, which is composed of an administrator server and a file server. The administration unit's role includes registration of accounts for endpoints and monitoring service with the blockchain network (to facilitate transaction processing through respective wallet addresses). Admin server also co-ordinates with the file server (containing SLO and endpoints' data) and updates the records against each endpoint in accordance with their wallet addresses on the blockchain. This updated record is used by the monitoring service to process the captured data in accordance with SLO. Monitoring service then pushes the endpoint's observed data onto the SLA monitoring blockchain network via JSON RPC API using the OpenSSL library for secure communication.

Within our setup, the blockchain network contains one seed node and two connected nodes. Seed node in blockchain generates and registers all the addresses for endpoints and monitoring service. This node is also responsible for managing rights for endpoints and monitoring service along with the creation and utilization of tokens that are moved through transactions from monitoring service wallet address to respective end point's wallet address. The token represents an acknowledgment of a successfully recorded

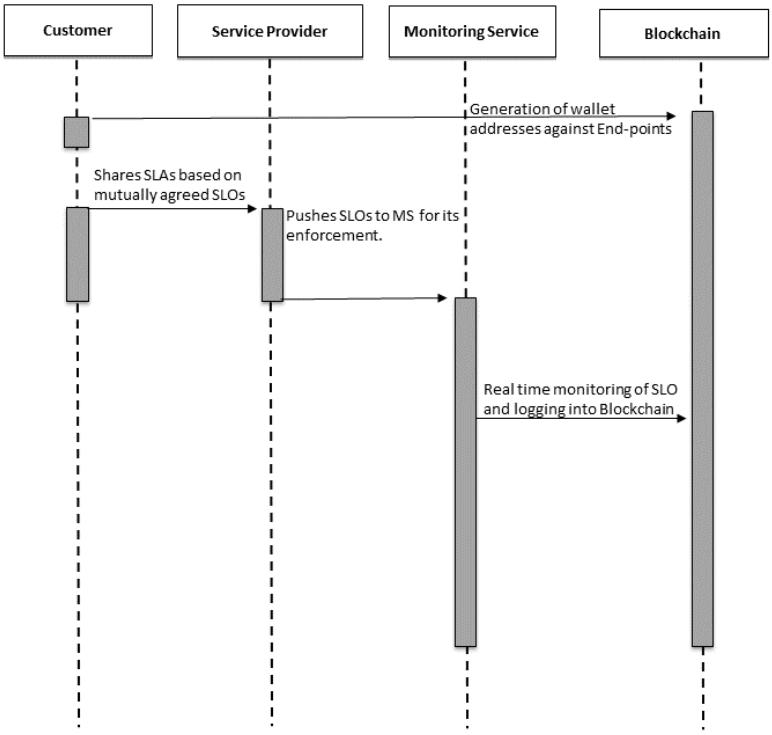


Figure 3: Sequence of events for the blockchain-based SLA monitoring process

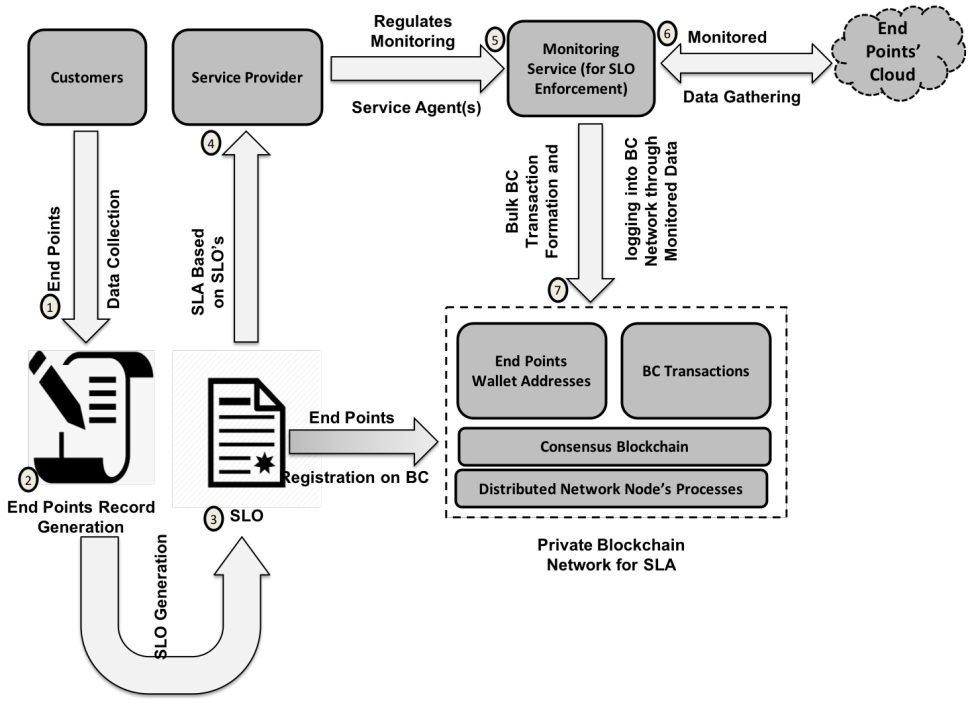


Figure 4: Blockchain-based SLA monitoring process model

transaction that is moved from the monitoring service account to the end point's wallet address. Connected nodes empower the decentralization of the distributed decentralized network by locally creating and maintaining their respective copies of the blockchain. The mining pool is shared among all the nodes to push the transaction data

into the blocks of consensus blockchain.

5. Experimental Setup

To evaluate the proposed system, we implemented the testbed presented in Figure 5 using Multichain within a private blockchain setup.

Here, monitoring service unit and blockchain network maintain end to end communication through multichain JSON based RPC API. These monitoring services have been specifically written to send transactions over the network using JSON based RPC APIs. They have support for SSL connections to its JSON-RPC API using Multichain platform [44]. The motivation to choose private blockchain is mainly due to our proposed model requiring the trusted third party monitoring infrastructure to run on a managed and controlled environment. The monitoring infrastructure maintains the ledger containing web services monitoring data, runs on a managed and controlled environment. Private blockchain also enables a permissioned model for accessing the data stored on the ledger, thereby adding a layer of security to achieve resilience against data tampering and leakage.

Our experimental blockchain network consists of three nodes (a seed node and two client nodes) connected within a local area network. The specifications of these nodes are presented in Table 2. The blockchain network consists of 10 miners with a blockchain generation rate of 15 seconds and a maximum block size of 8 MB, as described in Table 3. Monitoring service(s) checks the status of the endpoints periodically (preset time interval) and stores the result of each monitoring event for an endpoint in the form of a transaction. As our setup is a private blockchain, we have used *round robin* algorithm for consensus to ensure the transactions are added to blocks based on their arrival times. Figure 7 presents a typical transaction whereas Figure 6 presents a typical block within our blockchain.

For our experiments, we have used sixteen web-based public resources to represent service endpoints. A description of these endpoints, along with their blockchain wallet addresses, is presented in Table 4. To achieve a thorough analysis of our proposed monitoring system, we have used different scenarios with a varying number of monitoring service instances (one, two, and three instances), each with different service endpoints (1, 5, 10, 15). The time interval to conduct periodic checks sets to five seconds across all these settings.

6. Performance and scalability analysis of the proposed approach

The experiments have been carried out using different configuration settings with respect to parameters such as number of active monitoring services, number of endpoints (hosted services), and time interval for monitoring the responses of endpoints as shown in Table 5.

6.1. Average block size for varying endpoints

Here, the average size of the transaction is 578 bytes. In order to determine the scalability limit of our system. Let's denote the size of the SLA transaction, which is being forwarded from monitoring service to blockchain is T_size, rate of generating new blocks by backend blockchain by B_rate, the maximum volume of data that can be accommodated by the block is $B_{data-max}$. The average size of the block under which the existing blockchain is currently being operated, is represented by is $B_{data-avg}$. We are interested in determining the maximum count of transactions that a block can carry with itself, represented as $T_{max-count}$ per block and the current average number of transactions in a block ($T_{avg-count}$), which may slightly vary with varying number of monitoring services due to associated metadata with each transaction. The size of the SLA-transaction in our system is 578 bytes that is $T_size = 578$ bytes. The computation for the size has been performed as per the method mentioned in the official documentation of Multichain.

Within this experimentation, Case 1 referred to the scenario when one monitoring service was operated to push SLA transactions to the blockchain. Similarly, cases 2 and 3 make use of two and three parallel running monitoring services, respectively. Table 5 shows the linear increase in the average data size of blocks with the increase in the number of endpoints along with monitoring services. It can be seen in Table 5 that B_rate is set to 15 seconds and $B_{data-max}$ at 8MB while $B_{data-avg}$ is 2581 bytes (from Figure 8.A). In order to obtain the maximum count of transactions that a block can carry, $T_{max-count/block}$, we need to divide the maximum size of block, $B_{data-max}$ (that may be utilized for allocating transactions) by T_size (size of SLA transaction);

Mathematically;

$$T_{max-count/block} = B_{data-max}/T_{size} \quad (1)$$

Substituting the values (in bytes), we get;

$\implies \mathrm{T}_{max-count/block} = 8000000 \ / \ 578$
$\implies \mathrm{T}_{max-count/block} \cong 13841$ transactions / block

Computing the average number of SLA transactions in a single block according to the case where $B_{data-avg}$ is 2581 bytes; We have;

$$T_{avg-count/block} = B_{data-avg}/T_size \quad (2)$$

Placing values for block and transaction size (in bytes), we get;

$\implies \mathrm{T}_{avg-count/block} = 2581/\ 578$
$\implies \mathrm{T}_{avg-count/block} \cong 4$ transactions / block

Figure 8.A confirms our above result, where some sample blocks of case 1 with EP_n=1 has been shown.

Analyzing the calculated values of $T_{max-count/block}$ and $T_{avg-count/block}$, it may be inferred that our system may

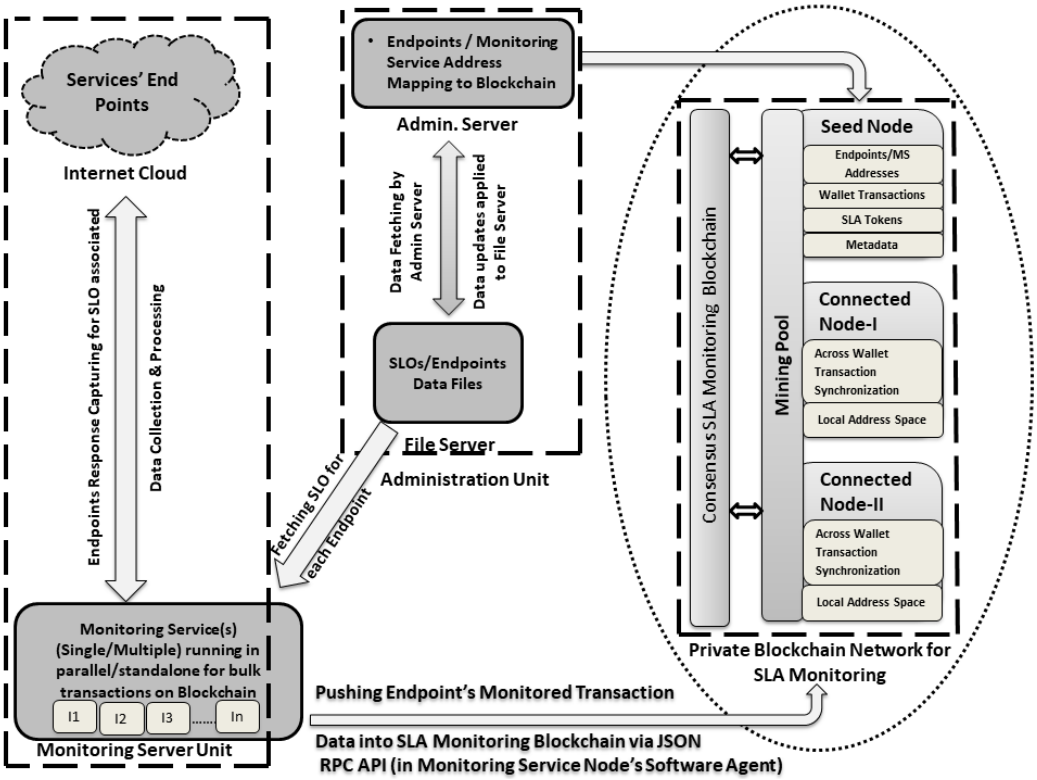


Figure 5: Testbed to implement blockchain-based SLA monitoring

Table 2: Hardware and software specifications

S. No.	Platform	Hardware Specification		
		Processor	Memory	Page File
01	Windows 10 Pro 64-bit	Intel Core i3-4005u CPU @ 1.70GHz (4CPUs)	4096MB RAM	5586MB Used 1887MB available
02	Windows 10 Home Single Lang. 64-bit (10.0, Build 17134)	Intel Core i7-7500U CPU @ 2.70GHz (4 CPUs) 2.9GHz	8076MB RAM	14346MB used 2836 available

```
E:\multichain-windows-2.0-alpha-4>multichain-cli.exe SLA_Monitoring listassets
{"method":"listassets","params":[],"id":"81630298-1585649184","chain_name":"SLA_Monitoring"}

[
    {
        "name" : "SLA_Validating_Tokens",
        "issuetxid" : "cb4c449b61fde7285eaa719dc2d01b54a29c28454dd687d64f6222a06be693ee",
        "assetref" : "60-266-19659",
        "multiple" : 1,
        "units" : 1,
        "open" : false,
        "restrict" : {
            "send" : false,
            "receive" : false
        },
        "details" : {
        },
        "issueqty" : 100000000,
        "issueraw" : 100000000,
        "subscribed" : false
    }
]
```

Figure 6: Sample block within the blockchain

further be scaled up to facilitate approximately three thousand four hundred and sixty times more transactions. One more thing that should be mentioned here is that the machines' computational strength (which are being operated at the monitoring service unit) may vary the result due to their processing strength.

6.2. Average transaction throughput

Now, considering the data from Table 5 where $B_{data-avg}$ has been the highest against different cases for monitoring services, we will perform the same computation for $T_{avg-count/block}$ for the data at row number 4,8 and 12 of the table. This will show us the existing operational upper limit of our proposed system under stress.

Therefore, using Eq. (2);

$$T_{avg-count/block} = B_{data-avg}/T_size \tag{3}$$

At $B_{data-avg}$ = 7185 bytes, B_{rate} = 15seconds, T_{size} = 578 bytes

Placing values for block and transaction size (in bytes), we get;

$$\implies \mathrm{T}_{avg-count/block} = 7185 \,/\, 578$$
$$\implies \mathrm{T}_{avg-count/block} \cong 12 \text{ transactions / block}$$

Table 3: Blockchain Parameters

Platform	Blockchain Parameters				
	Mining Diversity	No. of Miners	Block Gen. Rate	Max. Allowed Blk Size (MB)	Mining Turnover
Windows	0.3	10	15	8.3	0.5

Table 4: Web services monitored and their blockchain wallets

S.No	Endpoint Service Name	Endpoint Service Blockchain Wallet Address
01	Blockchain Research Group	182QwrqzWHmiBgjdC1pTQs1t1rdBGUHiQYm2F6
02	SLA Cloud service	14gtzt3XZVT3Mf61uAQoGEXxGgP6yc41awynzg
03	Yahoo	1UuPVjnTvewdSaw9Xe8XL1HzaTBYVi2yG8xZVD
04	Ebay	1VfSM1pC2qtFQaYHcYirGc8tMFUDLnJLjn5JPo
05	Google	17mY9QAim8i8XcQxVsUC6ficQTNYpKunmtwQQG
06	Example	1FjDokjwzYtT2mtWsk7irGY37hskaTtiXtjKWa
07	Paypal	1FX8jcXgN4dj36xvowFoo6ELkrv44rNS3k6AR9
08	Bing	1VTzdPGnDbX8e6L4W1iDn2uGryReC7LU5fKkWA
09	Techcrunch	1DYbGBPcrZYArYvyGhkrs1i2N59W8o1w2cENtn
10	Mashable	1JHcLQ4a8vtPb5UgVJkYyHrdAzX4KHimSaoEiQ
11	The next web	1K3A1qotfFo65HfSH8GARZLe8TK48eL8hj3Wbg
12	Wordpress.com	1T2QqCXp7o9GwixVbVnCQ8bm42pHUC5Sk4a7YH
13	Wordpress.org	1FkQApTwagEraRhzcTD3NEUfjyrVw96FeiSEKQ
14	NED University	1W5ZHwPvjbneXwJ3tC3g2UwHTotSB9LCZnLC84
15	Higher Education Comm.	1EBYDxDyvpJ8Au4Fnh9JUsGaGetdJqNMz9z4F7
16	Wikipedia	161r7RpfkYDraapBTibqnHXvBj5FGATuRo1SRY

```
E:\multichain-windows-2.0-alpha-4>multichain-cli SLA_Monitoring getblock 006972fbd3ea1b1f16205b4961ea042(
5bfa35f04b176a42847161e1c5f0b8f8
{"method":"getblock","params":["006972fbd3ea1b1f16205b4961ea04265bfa35f04b176a42847161e1c5f0b8f8"],"id":"
89230323-1586815244","chain_name":"SLA_Monitoring"}

{
    "hash" : "006972fbd3ea1b1f16205b4961ea04265bfa35f04b176a42847161e1c5f0b8f8",
    "miner" : "1GxMTAdaNg94QogeHB5P6wXWTReMhpVx3ZtMWE",
    "confirmations" : 10023,
    "size" : 2581,
    "height" : 1098,
    "version" : 3,
    "merkleroot" : "a13815427a7b08ddc697311e64a29587fbbb28ec4070694810cde815aca308e2",
    "tx" : [
        "e19f5b642b63c253b8b230cc05d3988b4c1129a8bc50bfa4cd0bdc0404907312",
        "0ec01c6a57ad11469487da79e2ea59161ce0d92476a252adf3bf33ee283f8dac",
        "1f866a5acd7a5a88bee0b36ca18e4d48769d3aeac1cf7e34a0cb71074b09f863",
        "ed8b5bc46df00092b9903cbea8798a91f5d8842ab1f87767739c1a394fe551a7",
        "322ca3df24b5d4b08445c85671659238a5a265b5d910276587037289e979358e"
    ],
    "time" : 1586214652,
    "nonce" : 774,
    "bits" : "2000ffff",
    "difficulty" : 5.96046447753906e-8,
    "chainwork" : "0000000000000000000000000000000000000000000000000000000000044b00",
    "previousblockhash" : "006dd466de314327e708b6894a87a039e9adb3676b81085a6486ed9583208ecb",
    "nextblockhash" : "008bc4687106292da47482fc7e12ed536c7d327d9b7e647b4b9578ee64cb99ce"
}
```

Figure 7: Sample transaction within the blockchain

Table 5: Average Block size with varying EP_n for Different Cases

S. No.	Case	Ep_n	D_{data_avg} (bytes)
1	1	1	2581
2	1	5	3499
3	1	10	4115
4	1	15	7185
5	2	1	3505
6	2	5	3933
7	2	10	9579
8	2	15	13009
9	3	1	6176
10	3	5	10399
11	3	10	15380
12	3	15	16108

At $B_{data-avg}$ = 13009 bytes, B_{rate} = 15seconds, T_{size} = 578 bytes

Placing values for block and transaction size (in bytes), we get;

$\implies T_{avg-count/block} = 13009 / 578$

$\implies T_{avg-count/block} \cong 23$ transactions / block

At $B_{data-avg}$ = 16108 bytes, B_{rate} = 15seconds, T_{size} = 578 bytes

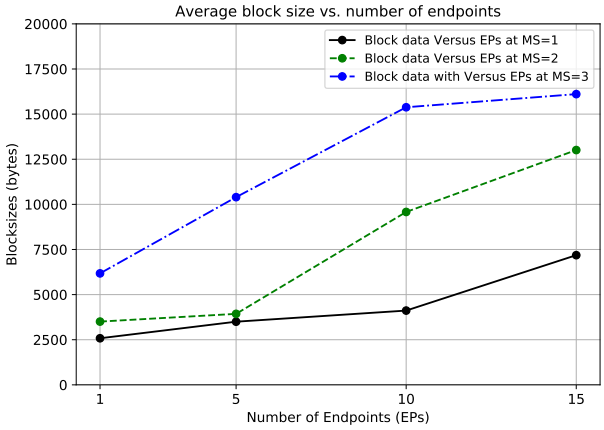
Placing values for block and transaction size (in bytes), we get;

$\implies T_{avg-count/block} = 16198 / 578$

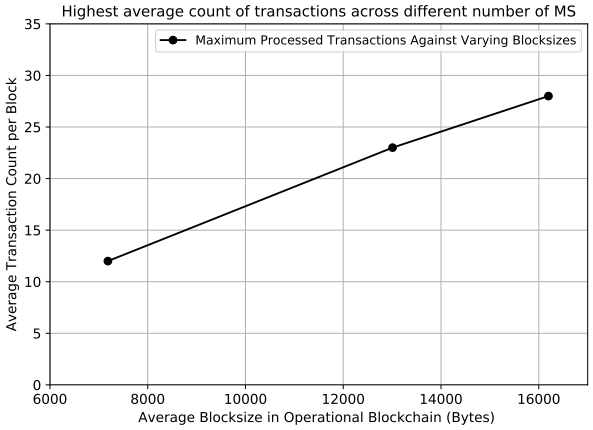
$\implies T_{avg-count/block} \cong 28$ transactions / block

Figure 8.B shows the linear increase towards average transaction count per block due to an increase in better utilization of the SLA-blockchain network. This also shows that the system at this stage may easily be further scaled up to meet further demands.

In this system, we have been pushing some metadata along



(a) EP_n vs. $B_{data-avg}$ with B_{rate}=15



(b) $T_{avg-count/block}$ vs $B_{data-avg}$ B_{rate}=15s

Figure 8: Experimentation to assess scalability of proposed system.

with each transaction through monitoring services. This metadata (which includes endpoint's URL, availability of hosted services, length of time it takes to respond, the number of times the service has been queried since its start, and the current timestamp) may cause some delays in transaction formation, which may ultimately affect the arrival time of the transaction to the blockchain but it does not impact the mining time of transaction. On the contrary, this situation can certainly affect the overall transaction throughput of the system and may lead to the under-utilization of backend Blockchain's capabilities and resources. Therefore, it is also equally important to investigate the scalability of our system by examining the maximum upper limit for the number of SLA-processed transactions per second.

6.3. Sensitivity analysis of transaction throughput vs. block-size

Lets T_{Tp} represents the transaction throughput of our proposed system. Hence, T_{Tp} can be mathematically expressed as;

$T_{Tp} = B_{data-max}$ / T_{size} / B_{rate}

Here we are keeping B_{rate}= 15seconds to determine the direct impact of block size on throughput. Applying values in Eq. (3), we get;

At $B_{data-max}$ = 8MB
$\implies$ T_{Tp} = 8000000 / 578 / 15
$\implies$ T_{Tp} = 922.722
$\implies$ $T_{Tp} \cong$ 923 transactions per second.

Using the same mathematical relationship between T_{Tp} and $B_{data-max}$; we can make a projected visualization for T_{Tp} based upon the experimental data of our system. Hence, determining scalability in context of T_{Tp} when available memory for block is increased to 80MB, we get;

At $B_{data-max}$ = 80MB
$\implies$ T_{Tp} = 80000000 / 578 / 15
$\implies$ T_{Tp} = 9227.220
$\implies$ $T_{Tp} \cong$ 9227 transactions per second.

Similarly, for 800 MB and 8000MB;

At $B_{data-max}$ = 800MB
$\implies$ T_{Tp} = 800000000 / 578 / 15
$\implies$ T_{Tp} = 92272.20
$\implies$ $T_{Tp} \cong$ 92272 transactions per second.

At $B_{data-max}$ = 8000MB
$\implies$ T_{Tp} = 8000000000 / 578 / 15
$\implies$ T_{Tp} = 9227220
$\implies$ $T_{Tp} \cong$ 922722 transactions per second.

This shows that the system bears the tendency to process 923 transactions at every second provided the nodes hardware/software configuration settings remains the same as shown in Table 5 along with the network connectivity strength (refer to Table 5) in accordance with existing settings while under same settings but by allocating more space for the block (at $B_{data-max}$ = 8000 MB), the same system may be scaled up to add as many as approximately nine lacs SLA-transactions per second which is considered adequate for a large scale global monitoring system. Figure 9 shows this relationship between T_{Tp} and $B_{data-max}$.

Figure 9 shows a big increase in transaction throughput when the block is allowed to accept more transactions to carry to the blockchain with a constant rate of 15 seconds for adding a new block. This assumes a continuous flow of incoming transactions from clients across the network; otherwise, the same situation may lead towards a disaster blockchain state if the resource utilization and responses are not properly analyzed due to frequent empty spaces within a block throughout the blockchain.

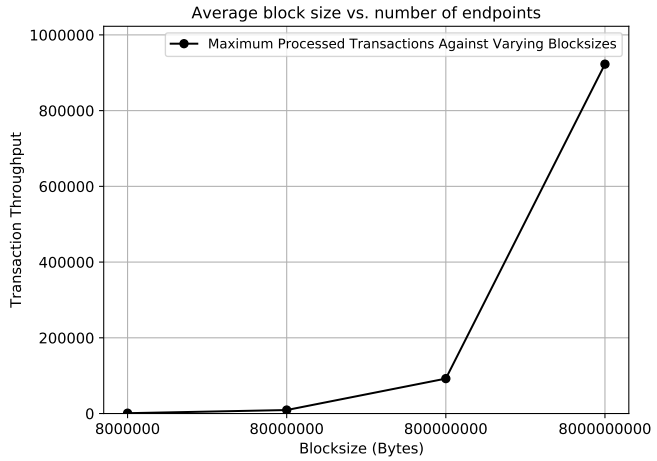


Figure 9: T_T vs $B_{data-max}$ (B_{rate}=15 seconds)

6.4. Sensitivity analysis of transaction throughput vs. block generation rate

Although transaction throughput is a vital attribute in assessing the scalability of the system, but there is another key factor which may affect the performance of the system by under or over-utilizing the available resources and refrain it to achieve an optimum level of performance. This key factor is the domain or application area, which sets up the benchmark performance for the system to deliver. Technically, at the blockchain level, this requires an optimum match of speed between the rate of incoming transactions and the rate of generating new blocks by the backend blockchain engine that is another area of investigation is T_{Tp} versus B_{rate}.

It is very important here to mention that the significance of the impact of maintaining an optimum level between an incoming flux of transactions and their additions through blocks to the blockchain. If this level is not maintained to meet the desired output, it may lead to a situation where monitoring service would likely to be able to push data to the blockchain at a slower rate (if the block size is configured to be more in size than the demand of the SLA system). At this moment, more transactions may likely to wait in the pool for their confirmations until the block is ready to be added to the blockchain (depending upon the block generation rate). This builds up a strong foundation for carrying out scalability analysis in the context of the number of incoming transactions versus its confirmation keeping the block memory size intact as it may dramatically increase or decrease the scalability strength of the system in terms of its performance for transaction throughput. Therefore, in order to assess the potential of our system in this context, we need to conduct a thorough stress testing analysis on the theoretical limits of transaction processing speed versus block generation rate.

Using Eq. (3), very exciting research findings may be obtained to investigate various dimensions of our system in the context of scalability. Figure 10 highlights the responses/behavior of our system when it is projected (using the mathematical relationship of Eq. (3)) to be stressed under the theoretical limits of one of the key scalability related attributes of our system, block generation rate. This mathematical projection is based upon the actual data obtained from the experimentation of the blockchain-enabled SLA system presented in this paper.

Performing computation using Equ. (3), we have;

$$T_Tp = B_{data-max}/T_{size}/B_{rate} \tag{4}$$

Now, here again, we will first consider the data from Table 5 where $B_{data-avg}$ has been the highest against different cases for monitoring services, and therefore we can use it for the maximum data consumed by the block against a particular monitoring service, which in our case is located at row number 4,8, and 12 of the table for monitoring service 1,2 and 3 respectively (can also be seen in Figure 8.B). The result will give us the existing operational upper limit of our proposed system under stress at different block generate rates. In the later part, we will carry out the same computation using the maximum block size allowed by our system for SLA blockchain to determine its potential peak performance.

6.4.1. Case 1: 1 monitoring service

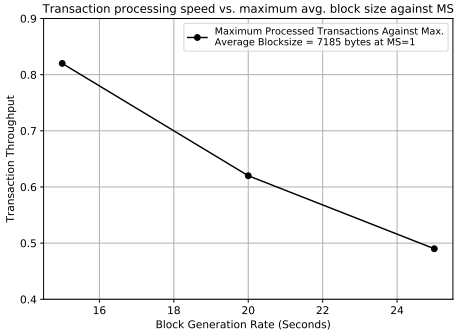
Applying values in Eq. (4) for system throughput T_{Tp} against maximum average blocksize achieved at EP_n=15, with one monitoring service.

At $B_{data-max}$ = 7185 bytes, B_{rate} = 15 seconds, T_{size} = 578 bytes, we get;
$\implies$ T_{Tp} = 7185 / 578 / 15
$\implies$ T_{Tp} = 0.82
$\implies$ $T_{Tp} \cong$ 1 transaction per second.

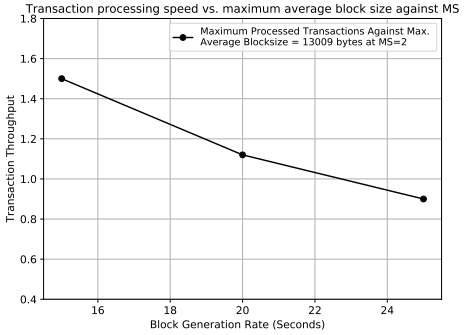
At $B_{data-max}$ = 7185 bytes, B_{rate} = 20 seconds, T_{size} = 578 bytes, we get;
$\implies$ T_{Tp} = 7185 / 578 / 20
$\implies$ T_{Tp} = 0.62
$\implies$ $T_{Tp} \cong$ ¿1 transaction every two seconds.

At $B_{data-max}$ = 7185 bytes, B_{rate} = 25 seconds, T_{size} = 578 bytes, we get;
$\implies$ T_{Tp} = 7185 / 578 / 25
$\implies$ T_{Tp} = 0.49
$\implies$ $T_{Tp} \cong$ 1 transaction every two seconds.

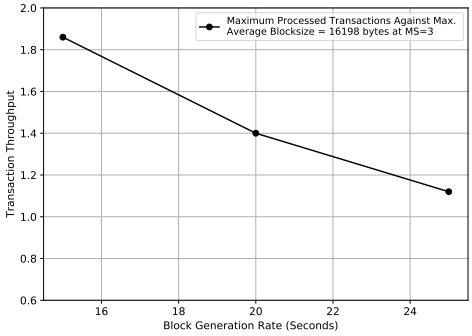
Figure 10.A shows the impact of increasing block generation rate to the existing operational strength of our system when 01 monitoring service is being used to observe 15 endpoints with block consumption of 7185 bytes. The trend in the graph discourages any further increase in the rate of generating new blocks as the system's current throughput is around 52 transactions per minute. Therefore, for one monitoring service, the graph shows a satisfactory and considerably stable status of the operation following the system's demand at the existing rate of the addition of new blocks.



(a) Avg. block size=7185B, MS=1



(b) Avg. block size=13009B, MS=2



(c) Avg. block size=16198B, MS=3

Figure 10: T_{TP} vs $B_{data-max}$ (consumed against each monitoring service) vs. B_{rate}.

6.4.2. Case 2: 2 monitoring service

Applying values in Eq. (4) for system throughput T_{Tp} against maximum average blocksize achieved at EP_n=15, with two monitoring services.

At $B_{data-max}$ = 13009 bytes, B_{rate} = 15 seconds, T_{size} = 578 bytes, we get;
$\implies$ T_{Tp} = 13009 / 578 / 15
$\implies$ T_{Tp} = 1.50
$\implies$ T_{Tp} is approximately 3 transactions after every two seconds

At $B_{data-max}$ = 13009 bytes, B_{rate} = 20 seconds, T_{size} = 578 bytes, we get;
$\implies$ T_{Tp} = 13009 / 578 / 20
$\implies$ T_{Tp} = 1.12
$\implies$ T_{Tp} is just over one transaction per second.

At $B_{data-max}$ = 13009 bytes, B_{rate} = 25 seconds, T_{size} = 578 bytes, we get;
$\implies$ T_{Tp} = 13009 / 578 / 25
$\implies$ T_{Tp} = 0.90
$\implies$ T_{Tp} is closed to one transaction per second

Fig 10.B shows a slightly better utilization of blocks with respect to the system's existing throughput (90 transactions per minute). The reason is, of course, an increase in the number of parallel running monitoring services (in contrast to one monitoring service, as shown in Fig 10.A) and the availability of space in the block. Although space utilization has been improved to 13009 bytes, the system is still able to satisfy the demand of monitoring endpoints at different intervals of time ranging from 05 to 15 seconds. Another important point we need to keep during the analysis of such graphs is that the throughput may also be varied a bit in accordance with the response of the endpoints to the monitoring service. The earlier it responds, the lesser the time it takes to form and send a transaction to the blockchain. One of the monitoring services pushes once the transaction to the transaction mining pool, blockchain internal processes start to work.

6.4.3. Case 2: 3 monitoring services

Applying values in Eq. (4) for system throughput T_{Tp} against maximum average blocksize achieved at EP_n=15, with three monitoring services.

At $B_{data-max}$ = 16198 bytes, B_{rate} = 15 seconds, T_{size} = 578 bytes, we get;
$\implies$ T_{Tp} = 16198 / 578 / 15
$\implies$ T_{Tp} = 1.86
$\implies$ T_{Tp} is approximately 2 transactions per second

At $B_{data-max}$ = 16198 bytes, B_{rate} = 20 seconds, T_{size} = 578 bytes, we get;
$\implies$ T_{Tp} = 16198 / 578 / 20
$\implies$ T_{Tp} = 1.40
$\implies$ T_{Tp} is over one transaction per second.

At $B_{data-max}$ = 16198 bytes, B_{rate} = 25 seconds, T_{size} = 578 bytes, we get;
$\implies$ T_{Tp} = 16198 / 578 / 25
$\implies$ T_{Tp} = 1.12

$\implies$ T_{Tp} is just above one transaction per second

Fig 10.C highlights the extreme case of the existing system where 15 endpoints are being monitored by three parallel running monitoring services. The maximum utilization of memory space increases to 16198 bytes on average per block. This shows that the system is still capable of handling more endpoints within the existing setup. The existing block generation rate is working optimally to maintain a maximum demand of 112 transactions per minute. Increasing the rate of creating new blocks will lead to a situation where transactions would be required to wait more to be mined into the block, thereby decreasing the system's overall performance.

6.5. Sensitivity analysis of transaction throughput for max. blocksize

Since the blockchain in our system has the potential to carry almost 8MB of transactions in each block, therefore we also must need to find out how much boost in the performance we may be able to obtain if the blockchain is fully scaled up and we let the SLA-transactions occupy the maximum possible and available block space. For this purpose, we need to investigate the impact of transaction throughput when the block generation rate is varied, and the block memory size is wholly utilized.

Therefore, applying Eq. (4) for the above scenario, system throughput T_{Tp} for maximum available blocksize can be calculated as below.

At $B_{data-max}$ = 8000000 bytes, B_{rate} = 15 seconds, T_{size} = 578 bytes, we get;
$\implies$ T_{Tp} = 8000000 / 578 / 15
$\implies$ T_{Tp} = 922.722
$\implies$ $T_{Tp} \cong$ 923 transactions per second

Now, applying the projected values for B_{rate} = 15 seconds, 20 seconds, and 25 seconds to compute the constraints which may be observed while blockchain under further stress.

At $B_{data-max}$ = 8000000 bytes, B_{rate} = 20 seconds, T_{size} = 578 bytes, we get;
$\implies$ T_{Tp} = 8000000 / 578 / 20
$\implies$ T_{Tp} = 692.04
$\implies$ $T_{Tp} \cong$ 692 transactions per second

At $B_{data-max}$ = 8000000 bytes, B_{rate} = 25 seconds, T_{size} = 578 bytes, we get;
$\implies$ T_{Tp} = 8000000 / 578 / 15
$\implies$ T_{Tp} = 553.63
$\implies$ $T_{Tp} \cong$ 553 transactions per second

Fig 11 shows that our investigation for keeping SLA blockchain to operate optimally encourages the block generation rate to be kept at 15 seconds to maximize the throughput. If we would further decrease this rate, the chances are there that the blockchain may start to mine empty blocks as it happened during experimentation when

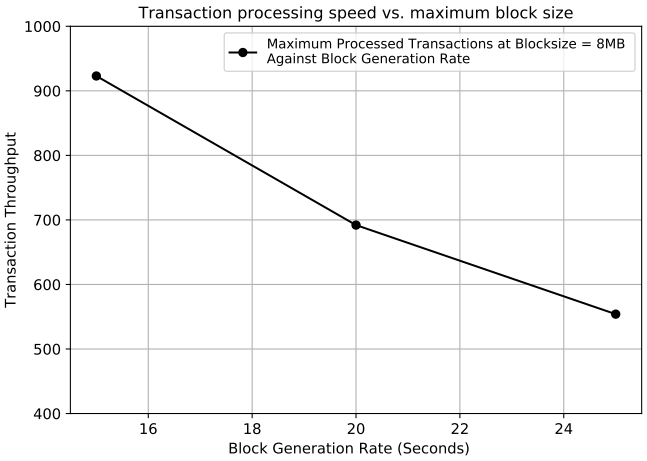


Figure 11: T_{TP} vs $B_{data-max}$ vs. B_{rate}

the checking interval by monitoring service is set at 15 seconds to observe the endpoints' responses. Fig 11 also reflects the true state of our blockchain as it works upon the actual transaction size of this system when the block is hypothetically completely filled by such transactions. At this stage, the system is capable of handling 923 transactions per second when the block generation rate is set to 15 seconds.

6.6. Scalability analysis for increasing monitoring services

Figure 12 represents the relation between block size with the variation of load through an increase in running the number of monitoring services. If we start analyzing the graph vertically with respect to different levels of interval in seconds (set by monitoring service to periodically check the status of the endpoints), we can see that overall; the trend is towards higher consumption of block size across cases 1, 2, and 3. A relatively unusual point at the graph can be found in cases 1 and 2 when monitoring services were being operated at a checking interval of 5 seconds, where the utilization of memory space of block is almost similar. This is most likely due to the fact the blockchain is one and common across all the cases. Although the monitoring services are running in parallel that is causing to generate multiple transactions from these two parallel running services, however, we must keep in mind that the block generation process is sequential, and these parallel running services, therefore, start to put pressure on the size of the block to scale it up. Therefore, more space is expected to be consumed as more transactions start to come in a shorter interval of time. The point to note here is that the difference in the average block size at this stage (when the checking interval is 5 seconds) is very little. This shows that the rate of incoming transactions was very close at that point in time when one and two parallel monitoring services had been running, respectively.

Another factor that affects the average block size here is the block generation rate (which is set to 15 seconds)

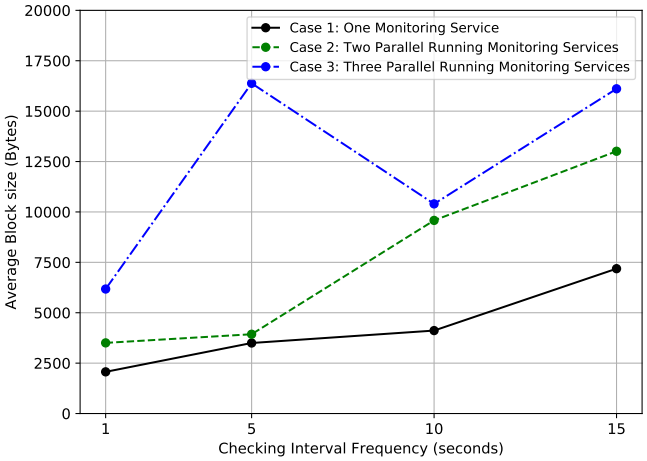


Figure 12: Block size variation with increasing load

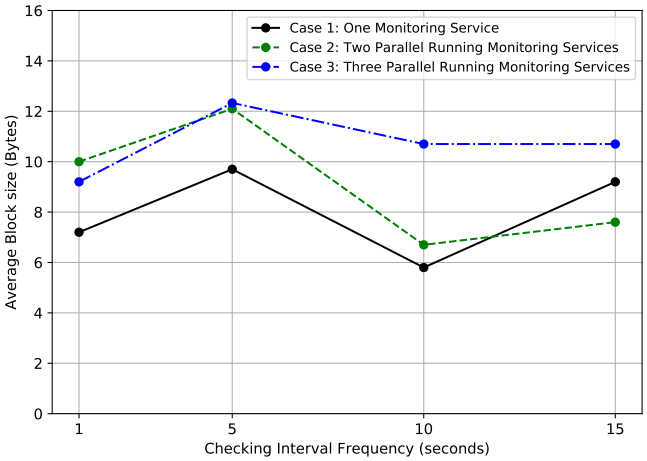


Figure 13: Transaction mining time against monitoring services

of blockchain. This rate represents the average value; that is, if a block takes more or less time than 15 seconds to be added to the blockchain, then the next block will be timed in such a way so that this average block generation rate may be restored. Combining both of these reasons, the graph may be understood at a point when case 2 and 3 when the interval is set to 10 seconds. However, the overall impact illustrates that if the system is loaded with more monitoring services keeping constant checking intervals, the average block size will be increased. Case 3 at an interval of 10 seconds shows a decline from its previous position; this is because of the delay in response by some of the endpoints when multiple (three to be exact) monitoring services in parallel had been querying endpoints. Trade-off lies between endpoints‘ responses and block generation rate to occupy memory space in the block.

As it can be observed in Figure 13, case 3 is relatively on a higher side most of the time in terms of confirming transactions to the block than case 1 due to a higher load on blockchain from case 1 to 3 through an increase in the number of monitoring services. Some exceptions may be ignored, such as for cases 1 and 2 against checking interval of 15 seconds, due to the randomness in the arrival time of transactions. For example, if a transaction arrives (through one of the monitoring services to the pool of unconfirmed transactions of the blockchain) immediately after the addition of the latest block, then the new transaction will have to wait to the addition of the next block (approximately 15 seconds in our case). Due to this reason, we have taken into account the average transaction mining time (out of nine total transactions every time while running a fixed number of monitoring service(s) and endpoints for every case such as case 1 was performed with one monitoring service while recording nine individual transactions taken at a difference of every 500 transactions for every varying number of monitored endpoints which in our scenario are 1,5,10, and 15. This makes 36 transactions at various stages of blockchain operations for each monitoring service. Hence, for all monitoring services running in parallel, that is 1, 2, and 3, a total of 36*3, which is equal to 108 transactions, have been considered. These transactions have been recorded after every 500 transactions, so overall, to get the behavior of our system, 500*108=54000 transactions were performed). Generally speaking, the transaction mining time will increase with the increase in the rate of incoming transactions to the blockchain system. This increase in time will not be due to an increase in the time it takes to put an individual transaction to the block but may also be due to the overutilization of memory space of a block once the rate of incoming transactions becomes too high to accommodate further transactions. In our case, the block size has not been completely utilized in the full operational blockchain (Figure 13), and the system throughput is also good (Figure for Transaction Mining Time), which may further be increased due to the available space in the block (Figure 13 showing the maximum consumed memory of block is well under our 8MB dedicated memory size of the block). This is why it can be said that our system is highly scalable to meet the future demands of increased throughput. Table 6, 7, 8 show the data that have been recorded for all the cases of above graph. Here MT_Avg represents the average mining time of a transaction while EP_n represents the number of observed endpoints.

6.7. Hypothesis Testing

We designed the two sets of hypotheses to determine the statistical evidence for the obtained results for increasing the number of endpoints and monitoring services on the transaction delays. The first set of hypotheses is related to increasing the number of endpoints and delays in transaction delays.

$H_{0a} : \mu_1 = \mu_2$
$H_{1a} : \mu_1 < \mu_2$

The second set of hypothesis is related to the increasing number of monitoring services for fixed (10) number of enddpoints.

Table 6: Avg. mining time for end points at MS=1

S. No.	MT_{avg}	EP_n
1	7.2	1
2	9.7	5
3	5.8	10
4	9.2	15

Table 7: Avg. mining time for end points at MS=2

S. No.	MT_{avg}	EP_n
1	10	1
2	12.1	5
3	6.7	10
4	7.6	15

Table 8: Avg. mining time for end points at MS=3

S. No.	MT_{avg}	EP_n
1	9.2	1
2	12.3	5
3	10.7	10
4	10.7	15

$H_{0b} : \mu_3 = \mu_4$
$H_{1b} : \mu_3 < \mu_4$

1. μ_1 : The population mean for the delays in transactions pickup by miners using one monitoring service against 1 endpoint.
2. μ_2 : The population mean for the delays in transactions pickup by miners using one monitoring service against 10 endpoint.
3. μ_3 : The population mean for the delays in transactions pickup by miners using one monitoring service against 10 endpoint.
4. μ_4 : The population mean for the delays in transactions pickup by miners using three monitoring service against 10 endpoint.

The level of significance is set at $\alpha = 0.05$. We rejected the null hypothesis (H_0 or H_{0b}) when the p-value based on the paired t-test is less than 0.05 for any specific feature.

The null hypothesis H_{0a} for the first set of the hypothesis is rejected for the average delay in the transaction for miners as the p-value is less than 0.05. It means that the average delay in transactions for an increasing number of endpoints will increase.

We rejected the null hypothesis H_{0b} for the second set of the hypothesis for the feature Processed Requests as the p-value is less than 0.05 and t-value is -6.54. It means that increasing the number of endpoints increases the transaction delays. For the increasing number of monitoring services, the null hypothesis H_{0b} is also rejected as we find p-value ¡0.00001 and t-value is -30.59. It means that the increasing number of monitoring services also contributes to increasing the transaction delay.

6.8. Scalability analysis for increasing monitoring service requests on transaction delays

We carried out approximately forty-eight thousand transactions overall by assigning a different number of endpoints (one, five, ten, and fifteen) against one, two, and three parallel running monitoring services such that all of these monitoring services were run in parallel for each case of endpoints for four hundred transactions in every iteration. The observation was made to record the latency between transaction receiving and picking up by miner with the gradual increase in the number of requests. The data was evaluated against real endpoints, as shown in Table 4. Figure 14, Figure 15, and Figure 16 show the trend where latency varies from 3 to 27 seconds covering all conditions for

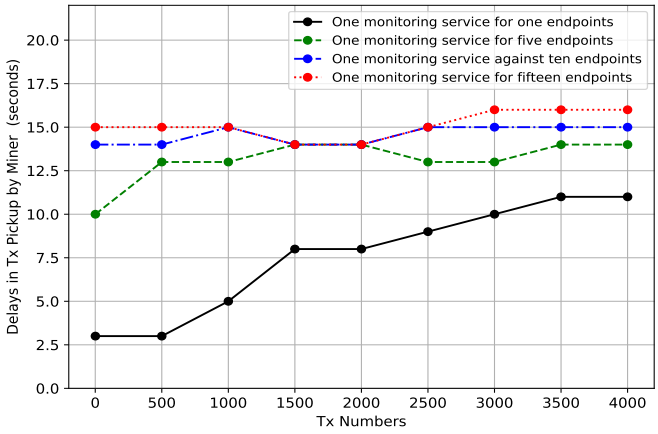


Figure 14: One MS with increasing load

number of monitoring services and endpoints. We can see here in Figure 14 that increasing the number of endpoints pushes more transactions in a block. This situation helps better utilization of blocks but may sometimes result in unexpected output. For instance, when a miner pickups a transaction after "n" seconds and another transaction after "n+2" seconds and both the transactions get their place in the same block. In this situation, the first transaction waits for the block to confirm the chain, although it came earlier than the rest of the incoming transactions for the same block. In general, we observe a slight increase in the delay when more transactions against requests are pushed into the blockchain by monitoring services. However, the maximum amount of delay after 50 thousand transactions was almost stable at 27 seconds. This does not pose any scalability threat to our system as the transactions were flooded at an equal interval of only five seconds with a maximum load of fifteen endpoints by three concurrent monitoring services, which is a higher rate even in a private blockchain.

6.9. Formal scalability analysis

We carried out formal analysis of our system by considering the total number of monitoring requests confirmed into the consensus blockchain by a single monitoring service for a specific length of time. This will help determine how scalable our system is with initially one monitoring service as a functional unit of operation. This scheme may be helpful to assess the scalability strength of the entire system when put into operation as a whole. The entire

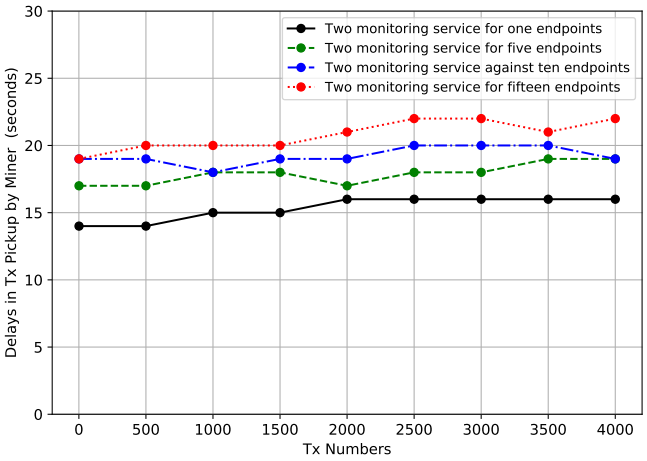


Figure 15: Two MS with increasing load

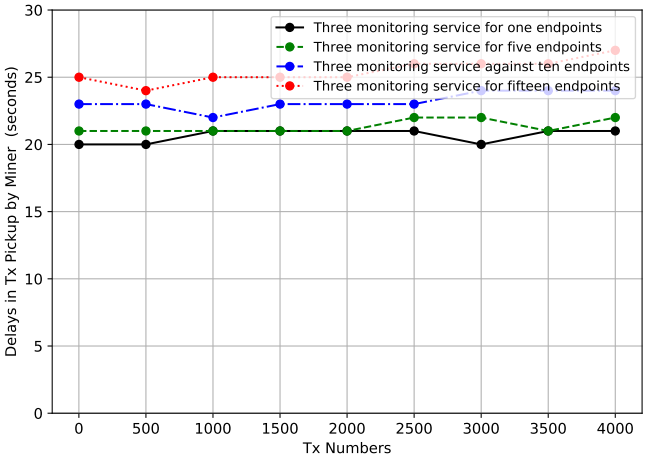


Figure 16: Three MS with increasing load

process of observing and recording data for cloud-hosted services into blockchain by a monitoring service may be modelled by using a queuing theory principle. Suppose there are 'k' number of cloud-hosted services that are being monitored by a monitoring service M_i per unit time 't'. In this scenario, this system follows a queue-like centralized service provider scheme, and therefore, its throughput may be monitored using an approach as proposed by Almeida et al. [45].

We assume that there is no restriction for monitoring any number of cloud-hosted services. These services will be monitored depending upon the monitoring service's current status, which is receiving the request to log the response time of cloud-hosted services into the blockchain. The rate of evaluating cloud-hosted services (number of services monitored per unit time) does not depend upon the overall number of services (size of the queue that forms the entire list of services) and the time consumed by the service under process. Suppose the process of monitoring and confirming the number of hosted services to blockchain per second is represented by M_t. We are interested in investigating the amount of time T_k when the system is dealing with the 'k' number of hosted services by a monitoring agent. The duration of time, taken by a single cloud-hosted service may be determined as a little duration when the monitoring agent is engaged.

$$T_k = 1 - \frac{M_i}{M_t}(\frac{M_i}{M_t})^k \tag{5}$$

Now, we may take into account the total amount of time, $\propto$, when the whole system (under consideration) is involved in monitoring the hosted services. $\propto$ may be computed as;

$$\propto = \sum_{k>0}^{k} T_k = 1 - T_0 = \frac{M_i}{M_t} \tag{6}$$

Simplifying the above equation for T_k; using $\propto = \frac{M_i}{M_t} in Equ.(5)$

$$T_k = (1- \propto)(\propto)^k \tag{7}$$

if $\overline{M}$ represents an average number of hosted services which are currently present in the system under consideration and being monitored by a monitoring agent, it may be calculated as;

$$\overline{M} = \sum_{k>0}^{k} k.T_k$$

$$\implies \overline{M} = \textstyle\sum_{k>0}^{k} k.(1- \propto)(\propto)^k$$

$$\implies \overline{M} = (1- \propto) \textstyle\sum_{k>0}^{k} k.(\propto)^k = (\frac{(1-\propto)\propto}{(1-\propto)^k})$$

$$\implies \overline{M} = (\frac{\propto}{1-\propto})$$

This formal analysis of scalability implements the concepts behind queuing and derived results of Little's formula from the book entitled "Distributed Systems" (2017, third edition, chapter No. 1) by Maarteen van Steen and

Andrew S. Tanenbaum, where they have shown the modelling of service scalability for a single servicing agent. We used the same principle to initially investigate the scalability for one monitoring service to project the impact of using multiple monitoring services. For a detailed investigation regarding scalability for the given scenario, we also need to factor into the equation the waiting time which is experienced by a hosted service(in the queue) in order to get its turn for evaluation along with the time consumed by the monitoring agent to record its (hosted service) performance attribute into the blockchain. The results of such analysis will help assess the total response time R_t of the unit of the system, we are modeling. The status of a monitoring agent service may be marked as busy when it is engaged in facilitating a hosted cloud service. This implies that our proposed system will be conducting evaluation against SLA with a throughput of M_t services per unit time. Similarly, the system is not idle for this specific length of activity (in terms of time) out of the whole time.

System's throughput β may be calculated as;

$$\beta = \propto .M_t + (1- \propto).0 = \frac{M_i}{M_t}.M_t = M_i \tag{8}$$

Using Little's formula, R_t may be determined as;

$$R_t = \frac{K_a vg}{\propto} \tag{9}$$

Solving the above equation using relations for $K_a vg$, $\propto and \mathrm{M}_t$, we get;

$$\frac{R_t}{Ser_t} = \frac{\propto}{(1- \propto)} \tag{10}$$

Where Ser_t represents service time allocated to a hosted service while R_t is the response time which is provided by the monitoring service agent to facilitate the hosted service. The value of $\propto$ is very critical here to maintain the ratio between response and service time. If the value of is very low, the system would be regarded as very scalable, and there will not be much waiting time in the queue for the hosted service to get its turn. The situation may change dramatically if the system is fully utilized and $\propto$ reaches 1. At this stage, the monitoring service may become unresponsive for some cloud-hosted endpoints.

6.10. Discussion

The experimental data detailed in the above sections assume a constant and frequent rate of incoming transactions from blockchain clients (monitoring services) through the entire experimentation work. Merely increasing block generation rate and other seemingly good looking attributes cannot alone guarantee for improved performance unless their dependencies upon other quantities are not evaluated properly. For example, in Figure 11, an increase of 5 seconds is causing 231 transactions to wait for the next block in a window of 1 to 20 seconds (average block generation time for $T_{Tp} = 692.04$ case), although at the same time it probably would have caused better utilization of block memory (in rare cases where the rate of incoming transactions may fall low). Fig. 10 and Fig 11 conclude that the operating attributes of blockchain should be kept realistic in accordance with the demand and nature of the application. Any miscalculation may result in the wastage of resources in terms of block memory (adding a block to blockchain containing a reasonable amount of empty spaces), Computational strength by over-utilizing resources (making block generation rate very high and keep nodes on toes), and under-utilizing space for mining pool of unconfirmed/waiting transactions. The paper shows an exhaustive evaluation of sending bulk transactions (approximately fifty thousand after every five seconds upto a maximum of 3 concurrent running monitoring services) in order to monitor its impact on the system in terms of the latency and responses of endpoint services. A detailed formal analysis of proposed model has also been conducted in context of its scalability. The analysis reveals the factors which are critical in enhancing the performance of the proposed system upon scaling by focusing on the capacity of a single monitoring service in execution for facilitating multiple services (as a unit of scalability) to project the behaviour of entire system in this context. This determines the overall scalability strength of the system.

7. Conclusion and Future Work

Service Level Agreements (SLAs), typically represented in the form of SLOs and financial penalties, are a mechanism to communicate and enforce service guarantees within cloud computing. An automatic method to periodically verify SLA terms in a transparent and trustworthy method is fundamental to effective SLA monitoring. Such a method will help mitigate inaccuracies and delays due to manual processes and can lead to widespread acceptance and credibility among cloud service users. This paper has presented a blockchain-based SLA monitoring infrastructure that leverages fundamental blockchain properties to achieve immutable and trustworthy SLA monitoring within cloud services. We have implemented the proposed system using Multichain and have evaluated it in different scenarios for a varying number of monitoring services and service endpoints. In the future, we wish to strengthen the capability of the network by introducing smart nodes, which are expected to make smart decisions on their own by learning historical network data to respond as per situation and building consensus over it for maximizing the potential of the network to address the challenges of scalability.

References

[1] Damián Serrano, Sara Bouchenak, Yousri Kouki, Frederico Alvares de Oliveira Jr, Thomas Ledoux, Jonathan Lejeune, Julien Sopena, Luciana Arantes, and Pierre Sens. Sla guarantees for cloud services. *Future Generation Computer Systems*, 54:233–246, 2016.

[2] Benedikt Pittl, Werner Mach, and Erich Schikuta. Bazaarblockchain: A blockchain for bazaar-based cloud markets. In *2018 IEEE International Conference on Services Computing (SCC)*, pages 89–96. IEEE, 2018.

[3] Marcel Müller, Sandro Rodriguez Garzon, Martin Westerkamp, and Zoltan Andras Lux. Hidals: A hybrid iot-based decentralized application for logistics and supply chain management. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0802–0808. IEEE, 2019.

[4] H. Zhou, X. Ouyang, Z. Ren, J. Su, C. de Laat, and Z. Zhao. A blockchain based witness model for trustworthy cloud service level agreement enforcement. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, April 2019.

[5] Eder John Scheid and Burkhard Stiller. Automatic sla compensation based on smart contracts. Technical report, Technical Report IFI-2018.02 https://files. ifi. uzh. ch/CSG/staff/scheid . . . , 2018.

[6] Eder J Scheid, Bruno B Rodrigues, Lisandro Z Granville, and Burkhard Stiller. Enabling dynamic sla compensation using blockchain-based smart contracts. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 53–61. IEEE, 2019.

[7] Amir Teshome Wonjiga, Sean Peisert, Louis Rilling, and Christine Morin. Blockchain as a trusted component in cloud sla verification. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, UCC '19 Companion, page 93–100, New York, NY, USA, 2019. Association for Computing Machinery.

[8] Amir Teshome, Louis Rilling, and Christine Morin. Verification for security monitoring slas in iaas clouds: The example of a network ids. *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7, 2018.

[9] E. Marilly, O. Martinot, S. Betge-Brezetz, and G. Delegue. Requirements for service level agreement management. In *IEEE Workshop on IP Operations and Management*, pages 57–62, 2002.

[10] Carlos Schweizer. Slamer: a blockchain-based sla management system. 2019.

[11] Adil Maarouf, Marzouk Abderrahim, and Abdelkrim Haqiq. Practical modeling of the sla life cycle in cloud computing. pages 52–58, 12 2015.

[12] Kashif Mehboob Khan, Junaid Arshad, and Muhammad Mubashir Khan. Investigating performance constraints for blockchain based secure e-voting system. *Future Generation Computer Systems*, 105:13 – 26, 2020.

[13] Robert Karaszewski, Paweł Modrzyński, and Joanna Modrzyńska. The use of blockchain technology in public sector entities management: An example of security and energy efficiency in cloud computing data processing. *Energies*, 14(7):1873, 2021.

[14] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[15] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[16] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE international congress on big data (BigData congress)*, pages 557–564. IEEE, 2017.

[17] Suporn Pongnumkul, Chaiyaphum Siripanpornchana, and Suttipong Thajchayapong. Performance analysis of private blockchain platforms in varying workloads. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–6. IEEE, 2017.

[18] Keke Gai, Jinnan Guo, Liehuang Zhu, and Shui Yu. Blockchain meets cloud computing: A survey. *IEEE Communications Surveys & Tutorials*, 2020.

[19] Zhen Zhang, Zicong Hong, Wuhui Chen, Zibin Zheng, and Xu Chen. Joint computation offloading and coin loaning for blockchain-empowered mobile-edge computing. *IEEE Internet of Things Journal*, 6(6):9934–9950, 2019.

[20] Mubariz Rehman, Nadeem Javaid, Muhammad Awais, Muhammad Imran, and Nidal Naseer. Cloud based secure service providing for iots using blockchain. In *IEEE Global Communications Conference (GLOBCOM 2019)*, 2019.

[21] Sheng Cao, Gexiang Zhang, Pengfei Liu, Xiaosong Zhang, and Ferrante Neri. Cloud-assisted secure ehealth systems for tamper-proofing ehr via blockchain. *Information Sciences*, 485:427–440, 2019.

[22] Marco Savi, Daniele Santoro, Katarzyna Teresa Di Meo, Daniele Pizzolli, Pincheira Miguel, Raffaele Giaffreda, Silvio Cretti, Kum Seung-woo, and Domenico Siracusa. A blockchain-based brokerage platform for fog computing resource federation. In *Conference on Innovation in Clouds, Internet and Networks*, 2020.

[23] Nir Kshetri and Jeffrey Voas. Blockchain-enabled e-voting. *IEEE Software*, 35(4):95–99, 2018.

[24] K. Gai, J. Guo, L. Zhu, and S. Yu. Blockchain meets cloud computing: A survey. *IEEE Communications Surveys Tutorials*, 22(3):2009–2030, 2020.

[25] Huan Zhou, Xue Ouyang, Zhijie Ren, Jinshu Su, Cees de Laat, and Zhiming Zhao. A blockchain based witness model for trustworthy cloud service level agreement enforcement. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1567–1575. IEEE, 2019.

[26] Nils Neidhardt, Carsten Köhler, and Markus Nüttgens. Cloud service billing and service level agreement monitoring based on blockchain. In *EMISA*, pages 65–69, 2018.

[27] Ahmed Taha, Ahmed Zakaria, Dongseong Kim, and Neeraj Suri. Decentralized runtime monitoring approach relying on the ethereum blockchain infrastructure. In *2020 IEEE International Conference on Cloud Engineering (IC2E)*, pages 134–143. IEEE, 2020.

[28] T. V. Nguyen, L. S. Lê, B. Dao, and K. Nguyen-An. Leveraging blockchain in monitoring sla-oriented tourism service provisioning. In *2019 International Conference on Advanced Computing and Applications (ACOMP)*, pages 42–50, 2019.

[29] Hamda Al-Breiki, Muhammad Habib Ur Rehman, Khaled Salah, and Davor Svetinovic. Trustworthy blockchain oracles: Review, comparison, and open research challenges. *IEEE Access*, 8:85675–85685, 2020.

[30] Mona Taghavi, Jamal Bentahar, Hadi Otrok, and Kaveh Bakhtiyari. A blockchain-based model for cloud service quality monitoring. *IEEE Transactions on Services Computing*, 2019.

[31] Rafael Brundo Uriarte, Rocco De Nicola, and Kyriakos Kritikos. Towards distributed SLA management with smart contracts and blockchain. In *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 266–271. IEEE, 2018.

[32] Limao Ma, Kosuke Kaneko, Subodh Sharma, and Kouichi Sakurai. Reliable decentralized oracle with mechanisms for verification and disputation. In *2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW)*, pages 346–352. IEEE, 2019.

[33] Shafaq Naheed Khan, Faiza Loukil, Chirine Ghedira-Guegan, Elhadj Benkhelifa, and Anoud Bani-Hani. Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-peer Networking and Applications*, pages 1–25, 2021.

[34] F. Daniel and L. Guida. A service-oriented perspective on blockchain smart contracts. *IEEE Internet Computing*, 23(1):46–53, 2019.

[35] Amir Teshome Wonjiga, Louis Rilling, and Christine Morin. Defining Security Monitoring SLAs in IaaS Clouds: the Example of a Network IDS. Research Report RR-9263, Inria Rennes Bretagne Atlantique, March 2019.

[36] A. Alzubaidi, E. Solaiman, P. Patel, and K. Mitra. Blockchain-based sla management in the context of iot. *IT Professional*, 21(4):33–40, 2019.

[37] A. Alzubaidi, K. Mitra, P. Patel, and E. Solaiman. A blockchain-based approach for assessing compliance with sla-guaranteed iot services. In *2020 IEEE International Confer-*

ence on Smart Internet of Things (SmartIoT), pages 213–220, 2020.

[38] Lei Hang and Do-Hyeun Kim. Sla-based sharing economy service with smart contract for resource integrity in the internet of things. *Applied Sciences*, 9(17), 2019.

[39] Mostafa Ghobaei-Arani, Sam Jabbehdari, and Mohammad Ali Pourmina. An autonomic approach for resource provisioning of cloud services. *Cluster Computing*, 19(3):1017–1036, 2016.

[40] Mostafa Ghobaei-Arani, Reihaneh Khorsand, and Mohammadreza Ramezanpour. An autonomous resource provisioning framework for massively multiplayer online games in cloud environment. *Journal of Network and Computer Applications*, 142:76–97, 2019.

[41] Mostafa Ghobaei-Arani and Alireza Souri. Lp-wsc: a linear programming approach for web service composition in geographically distributed cloud environments. *The Journal of Supercomputing*, 75(5):2603–2628, 2019.

[42] Xing Liu. Towards blockchain-based resource allocation models for cloud-edge computing in iot applications. *Wireless Personal Communications*, pages 1–19, 2021.

[43] A. Alzubaidi, K. Mitra, P. Patel, and E. Solaiman. A blockchain-based approach for assessing compliance with sla-guaranteed iot services. *2020 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pages 213–220, 2020.

[44] Multichain. Open platform for blockchain applications.

[45] Virgilio AF Almeida and Daniel A Menasce. Capacity planning an essential tool for managing web services. *IT professional*, 4(4):33–38, 2002.