Fall 2019

Using Blockchain Technology for The Organ Procurement and Transplant Network

Utsav Jain

San Jose State University

USING BLOCKCHAIN TECHNOLOGY FOR THE ORGAN PROCUREMENT AND TRANSPLANT NETWORK

A Thesis

Presented to

The Faculty of the Department of Computer Engineering

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Utsav Jain

December 2020

The Designated Thesis Committee Approves the Thesis Titled

USING BLOCKCHAIN TECHNOLOGY FOR THE ORGAN PROCUREMENT AND TRANSPLANT NETWORK

by

Utsav Jain

APPROVED FOR THE DEPARTMENT OF COMPUTER ENGINEERING

SAN JOSÉ STATE UNIVERSITY

December 2020

Guzun Gheorghi, Ph.D.	Department of Computer Engineering
Saldamli Gokay, Ph.D.	Department of Computer Engineering
Mak Ronald, M.S.	Department of Computer Engineering

ABSTRACT

USING BLOCKCHAIN TECHNOLOGY FOR THE ORGAN PROCUREMENT AND TRANSPLANT NETWORK

by Utsav Jain

The organ donation system in the United States is centralized and difficult to audit by the general public. This centralized approach may lead to data integrity issues in the future. The Organ Procurement and Transplant Network (OPTN) was built and maintained by a non-governmental organization called the United Network for Organ Sharing (UNOS) under its proprietary UNet^{SM} umbrella platform. This platform is made up of proprietary closed source software and does not provide the general public easy access to the organ transplant data for auditing. This study investigates the feasibility, challenges, and advantages of a blockchain-based OPTN. A prototype of a blockchain-based OPTN was created using the Hyperledger Fabric framework. The policies and guidelines issued by the United States Department of Health and Human Services for UNOS and the OPTN were used as the basis of this prototype. Four factors were identified to have a direct effect on the performance of this system, viz. max batch time out, max block size, endorsement policy, and transaction rate. Additionally, two variants of the blockchain chaincode were also developed. The first variant performed the organ-candidate matching inside the blockchain (Scheme A), and the second variant performed it outside the blockchain (Scheme B). Analysis of these data showed that Scheme A outperformed Scheme B in all experiments for write-operations. However, the read operations remained unaffected by any of the experiment variables in the given environment. Based on these results, it is recommended to perform the organ-candidate matching on the blockchain with the max batch time out close to the transaction rate.

ACKNOWLEDGMENTS

I would like to thank my research advisor Professor Gheorghi Guzun for his immense reserve of patience and constant guidance. I would also like to thank the committee members, Professor Gokay Saldamli and Professor Ronald Mak of the Computer Engineering Department for agreeing to contribute to my research. Lastly, I would like to thank the countless anonymous men and women whose efforts allowed me to pursue higher education in the San José State University as an international student.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Organ donation is the process of extracting an organ from an organ donor and then surgically replacing that organ in a recipient. The donor can be living or deceased. According to government figures [1], a deceased donor can save up to eight lives. On the receiving side of the equation are the organ recipients. They come from a pool of candidates, which is made up of patients who are suffering from ailments, treatment of which requires replacement of one or more organs. Since the demand for the donated organs grossly outstrips their supply, the candidates have to be placed on a long waiting list. In fact, according to the same source, twenty people die every day waiting for an organ transplant offer, and a new candidate is added to the waiting list every ten minutes. Due to the high stakes involved and the significant disparity in the supply and demand of transplantable organs, the fairness of the organ donation system is paramount. The fairness of the system is associated with the medical information of the individual patient as well as the organ allocation process.

In 1984, the United States Congress granted United Network for Organ Sharing (UNOS) a non-governmental organization the exclusive responsibility to build and maintain an Organ Procurement and Transplant Network (OPTN). This network consists of various stakeholders in the organ transplant process, including but not limited to transplant hospitals, organ procurement organizations (OPO), histocompatibility laboratories, and the general public. This separation was done to prevent direct government interference in the process of allocation of donated organs. However, the United States Department of Health and Human Services directs the policies and bylaws which govern the operation of the OPTN. Among many other things, these bylaws and policies lay out the criteria for matching a particular donated organ with the most suitable candidate. They provide a way to generate a numeric score for a candidate and describes

an algorithm to match an organ to the candidate. This “matching algorithm” makes the process transparent.

Currently, UNOS uses its in-house proprietary digital platform called UNet^{SM}. It is a centralized service connecting all fifty-eight OPOs, two hundred fifty-four transplant hospitals, and one hundred fifty histocompatibility laboratories [2]. This platform is an umbrella for six different services providing essential services to healthcare providers to enroll patients for the transplant candidature, match them with suitable donors and log the required medical and personal information into the system over the internet. The next section provides a more detailed description of these services.

One problem with this approach is that the security and veracity of this information fall entirely on UNOS’s capabilities to keep their system secure and their ability to detect intrusions and manage the damage (if any) caused by them. Even though there have been no concrete allegations of wrongdoing against UNOS, the exactitude of the waitlist information is entirely based on people’s trust in UNOS’s abilities to keep it safe from hackers and corrupt employees. Blockchain is a technology that can eliminate this single point-of-failure and at the same time, raise the bar for a nefarious entity to manipulate any records.

A blockchain-based OPTN has the potential to solve these problems. First, a blockchain-based network for organ transplants would be a peer-to-peer network. In such a setup, all peers in the network have a copy of the entire ledger. This replication allows all the peers to conduct their audits independently and verify compliance with the codified policies and bylaws. Second, this eliminates the single point-of-failure problem. To change some data, an attacker would have to crack many different nodes in the network rather than just one, hence raising the bar for the attacker.

To design a prototype of the OPTN, the official OPTN Bylaws [3], and policies [4] were studied. They were used to design the domain description and to write the chaincode (smart contract) for the prototype blockchain network. This blockchain network was built using the Hyperledger Fabric framework, and thus its official documentation [5] was also studied.

That being said, there has been limited thorough academic research into blockchain technology. Moreover, this technology has also acquired a bad reputation due to the so-called "Blockchain Bros" who cook up unrealistic blockchain ideas, sometimes intending to defraud investors. This work aims to alleviate this problem by proposing OrganChain, a prototype of a potential design for a blockchain-based OPTN. Performance metrics were calculated from this prototype to derive the results.

This research intended to ascertain the feasibility of a blockchain-based OPTN, as well as to discover the impact certain factors have on the performance of a blockchain-based OPTN. Since such a solution has not been explored by contemporary research before, a large number of unknown variables affected the outcome of this research. For example, the creation of the blockchain network proved to be a more complex task than initially expected, forcing the author to restrict the extent of the experiments that could be run in a limited time.

Based on initial exploration, four factors were identified to have a significant effect on the proposed system. They are the "max batch time out," "max block size," "endorsement policy," and "transaction rate." The first three are the delimiting factors for cutting a block and the last one due to apparent reasons. Each of the four factors is organized as four experiments in this document. Furthermore, each experiment is further divided into sub-experiments. Additionally, the author initially hypothesized that executing the matching algorithm outside the blockchain could show performance benefits. The rationale was that since a chaincode runs in every peer, by taking this compute-heavy task

of executing the chaincode away from the peers, its load would reduce, and it could spend all its capabilities in propagating the blocks.

Consequently, to test this hypothesis, two versions of the chaincode were created. In the first one, the matching algorithm ran inside the blockchain, and in the second one, it ran outside the blockchain network. These variants are referred to as Scheme A and Scheme B in the rest of this document. Unfortunately, the data collected by the experiments strongly favor Scheme A, thus rejecting this hypothesis. Additionally, a positive correlation was found between the "transaction rate," and the "max batch time out." All experiments were run using docker containers running inside a virtual machine (VM) provisioned using the Google Cloud Platform (GCP).

The remainder of the thesis is structured as follows. Chapter 2 describes the concepts of the organ donation process, blockchains, and Hyperledger Fabric. Chapter 3 describes OrganChain, the blockchain-based prototype used to produce empirical data for this work. Chapter 4 provides the details of the experiments and the observations determined from them. Chapter 5 describes future work on this topic, and Chapter 6 concludes this thesis along with the author's closing remarks.

2 LITERATURE REVIEW

This section covers information about the process of organ transplantation, followed by fundamentals for blockchain technology. It is laid out in four parts. First, it starts with a description of the organ transplant process, which falls within the scope of this work, including information about the current system, the UNet^{SM} platform. The second part lays out the essential features of blockchain technology and the Hyperledger Fabric framework. This framework was used to create the prototype for this research. Then, the next two sections describe the advantages and limitations of using blockchain technology for the OPTN, which is based on current research available to the author. Lastly, some miscellaneous literature is mentioned.

2.1 The Organ Transplant Process

When a patient's one or more organs have been damaged beyond repair, in some instances, one or more donated organs can be used to replace them. This process of taking an organ or a tissue from a donor person and using it to replace a damaged one of the same type in a recipient is called *organ transplant*. It is also called *grafting*.

2.1.1 United Network for Organ Sharing (UNOS)

UNOS is the organization contracted by the federal government to manage the organ transplant network. UNOS maintains a patient waitlist and tracks donated organs based on well-defined policies in collaboration with various stakeholders using contemporary communication technologies. This section describes the OPTN terminology pertaining to this thesis. The complete OPTN guidelines and policies can be found at [4] and [3].

A *candidate* is any patient enrolled on the waitlist to receive one or more organs. A candidate can be an *active candidate*, an *inactive candidate*, or a *multi-organ candidate*. An active candidate is defined as a candidate who is currently eligible for an *organ transplant offer*. This implies that there exists a reasonable chance of success in case a donated organ is transplanted into this candidate.

Conversely, an inactive candidate is one who, for some reason, is currently ineligible to accept an organ offer. For example, they may have contracted an infection after being placed on the waitlist, and they must be treated for this infection before they can accept a transplant offer. Lastly, a multi-organ candidate is one who is suffering from multi-organ failure and requires multiple donated organs to make a full recovery.

Eventually, a candidate becomes a *recipient* when they receive a donated organ. However, before that, they need to appear on a list of *Potential Transplant Recipients (PTR)*. This list of candidates is the result of a *match run*. The recipient at the top of this list is the *Primary Potential Transplant Recipient*. This candidate is first made the offer for a donated organ.

On the other hand, a *donor* is a patient who is the source of a donated organ. Based on certain factors, multiple organs can be extracted from an individual donor. A donor can be *living*, *deceased*, or a *domino donor*.

Healthy people can donate some organs and tissues. For example, a healthy human can live a reasonably good life with just one kidney along with proper care. Such donors fall under the category of living donors.

Deceased donors are patients who have been declared dead ("eligible death") or facing an "imminent neurological death," criteria for which are clearly defined and strictly followed. According to Penn Medicine [6], the most common myth about organ donations is that doctors will treat patients differently if they are organ donors.

Additionally, in the case of a domino donor, due to some specific medical conditions, a healthy organ needs to be separated from a living donor. This healthy organ can then be donated to another person. For example, if a person is suffering from cystic fibrosis, a lung disease, a heart-lung transplant can lead to an excellent outcome. The hearts of these recipients can then be transplanted into patients who require a heart transplant with a high chance of success [7].

According to the Centers for Medicare and Medicaid Services (CMS), [8], an *OPO* has a significant role to play in an efficient and orderly transfer of a donated organ to its intended recipient. A detailed list of responsibilities of an OPO is systematized in the OPTN Policy document published by the Health Resources and Services Administration, an agency of the Department of Health and Human Services [4].

In brief, an OPO is responsible for identifying potential organ donors in the zone in which they operate, then procure as many organs as possible from the donor and transport the organs to their selected candidate with the required care. A *donor hospital* is a hospital where the organ donor is or was undergoing treatment, and a *transplant hospital* is where the transplant surgery will be performed.

After a donor is added to the system by the host OPO, a match run is produced, and the host OPO sends an *organ offer* to the transplant hospital of the Primary PTR. The transplant hospital must respond with either of the following two responses. An *Organ Offer Acceptance* starts the process of sending a matched organ to be transplanted to the selected candidate. On the other hand, in the case of an *Organ Offer Refusal*, a new candidate is given an offer based on the PTR list.

According to the OPTN Bylaws [3], a "histocompatibility laboratory performs histocompatibility testing, including but not limited to, HLA typing, antibody screening, compatibility testing, or cross-matching, and serves at least one transplant hospital member or OPO."

A *matching system* is a computerized application that finds the most suitable candidate when an organ comes up for donation by performing a *match run*. A match run is a process that generates an ordered list of suitable candidates for each donated organ. The order of this list is based on a candidate's medical compatibility and the type of donated organ. The decision varies from organ to organ. Different organs can survive for a different number of hours outside the donor's body. The survival time ranges for each

organ are displayed Fig. 1, taken from [9]. However, some common factors need to agree, for example, blood type, body size, etc. The complete set of policies for each organ is codified in the OPTN Policy [4].

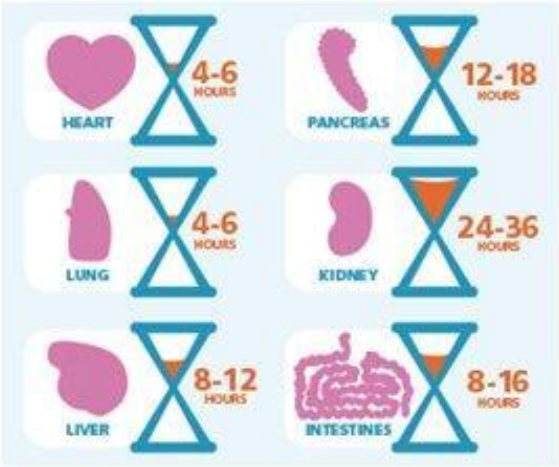


Fig. 1: Organ survival duration outside the donor's body.

2.1.2 The UNet^{SM} Platform

This platform encompasses the following six services which, work in collaboration, viz. "Waitlist^{SM}," "DonorNet$^{©}$," "DonorNet Mobile^{SM}," "UNet^{SM} APIs," "TIEDI$^{©}$," and "TransNet^{SM}" [2]. These services enable organ OPOs to offer donor organs and information electronically. Waitlist^{SM} is the service that manages the medical records of the candidates. This platform computes a numeric score for each candidate on the waiting list by applying predefined policies on the patient's medical records.

Using DonorNet^{SM}, OPOs can add donors' medical records and other pertinent information to the network by using DonotNet^{SM}. They can also perform "match runs" and generate the PTR List and produce "Organ Offers" to the candidates on the list.

To enable the bidirectional flow of relevant data to and from a certified Electronic Health Record (EHR) systems, UNet^{SM} extends an application programming interface (API). The Transplant Information Electronic Data Interchange or simply TIEDI$^{©}$

provides OPTN members access to the relevant EHRs of the patients in the Organ Transplant Network. Lastly, TransNet^{SM} is used by an OPO to create electronic package labels. This uniform labeling and marking system lead to improved logistics.

2.1.3 Current Load

According to official statistics [1], on average, one patient is added every ten minutes to the waiting list, and in total, there are currently one-hundred thirteen thousand patients on the waiting list. Also, according to Harvey and Weigel in [10], the current size of the data set holding the relevant information of the donors and candidates on the waiting list is around nine gigabytes. This information is based on the data set acquired by Harvey and Weigel in June 2014 and March 2015.

2.1.4 Criticism of the current implementation of the OPTN

The entire UNet^{SM} platform is closed source. Thus, it is difficult to vouch for its integrity. The author was unable to find a single Common Vulnerabilities and Exposure (CVE) related to the entire platform. The absence of any CVEs could mean that either no vulnerability exists in the system or no vulnerability has been found in the system. The latter is more likely.

Second, the data sets held by UNOS are not easily auditable by the general public. One can request for a copy of the data set from their online portal. However, the fee for this is one thousand US dollars. Although for academic purposes, they charge a reduced fee of two hundred fifty US dollars. Even so, UNOS ships a physical disk with historical data, not up-to-date data. Making matters even worse is the fact that there is no means for the person requesting the data to verify the veracity of the data. One has to have faith that UNOS was not compromised.

According to this press release [11], UNOS conducts regular security training, but the process is fiat of UNOS. A blockchain-based organ transplant system can provide better accountability.

2.2 Blockchains

The Jul/Aug 2019 issue of the Wired magazine published a Venn diagram [12] which portrayed blockchains as a "software platform" which is "complicated to explain" and "vaguely applicable to any problem". Nonetheless, simply put a blockchain-based system would store a set of transactions encapsulated in blocks which are then sequentially linked using advance cryptographic operations to form a chain of blocks. This chain is, in turn, replicated on all nodes that are participating in the blockchain network.

According to Sultan and et al. in [13], the most salient features of blockchain technology are *immutable-ledger*, *decentralization*, *consensus-driven validation*, and *transparency*.

Data, once written, cannot be modified or removed from an immutable ledger. However, delete transactions can be used to remove data logically. Still, the transaction history cannot be changed. Advanced cryptographic techniques are used to enforce this immutability, which makes it computationally infeasible to change any block in the blockchain.

Blockchain applications are decentralized. Not to be confused with distributed, it means that a single centralized entity does not control the decision to accept or reject a transaction. Instead, a set of instructions known as a smart contract or chaincode run individually and independently on nodes in a blockchain network to ensure that a particular transaction is acceptable. This collaborative approach eliminates any single point-of-failure.

An extension of blockchain's decentralized property is its consensus-driven validation. It implies that a subset of peers needs to agree to achieve a quorum for validating and committing a transaction to the blockchain. Finally, a blockchain's operation is transparent, and thus, any activity on it is easy to audit since all peers in the network contain a copy of the ledger.

Any blockchain application is a distributed system of considerable complexity, and each implementation has its method of dealing with this complexity. However, all blockchain frameworks have a mechanism to create a *smart contract*, a *consensus algorithm*, a *ledger*, and *peers*.

A smart contract is a piece of code that is automatically executed when a transaction is initiated. This code is mainly used to determine the validity of the transaction rather than provide any security. For example, a smart contract for a banking application would have the code to reject a money transfer from an account if the money to transfer is more than the account holder's balance.

Every blockchain-based application requires a consensus mechanism because there is no central authority that can execute some code to determine the validity of a read-write request. An agreement needs to be reached by a subset of participating nodes to approve or reject a transaction. This agreement may be achieved by a simple majority vote or a more elaborate scheme.

A blockchain ledger is the actual data store. It is like an append-only linked list where each block represents a node of the linked list. Additionally, the hash of the previous block is stored in the next block. In this way, whenever a new block is created, it has some mathematical relationship to all the previous blocks in the blockchain.

A peer is any participating node in the blockchain network. It is a machine, virtual or bare-metal, which stores a copy of the ledger. It executes the smart contract and determines consensus. However, each machine may be responsible for doing either or both of these things.

2.3 Hyperledger Fabric Framework

Hyperledger Fabric is part of the Linux Foundation's Hyperledger umbrella of blockchain technologies. It is also the framework used to create the experimental prototype for this research.

2.3.1 Hyperledger Fabric Domain Hierarchy

Since there are a large number of components involved in creating a blockchain network, Hyperledger Fabric allows for the creation of hierarchical namespaces or domains. A typical domain structure would include the *top-level domain*, the *orderer domain*, and the *organization level domain*.

The top-level domain is like a unique name given to the project. This name is similar to Domain Name Space's (DNS) top-level domain for a website. Below it is the orderer domain, and it must be separate from any of the organization's domain. All the nodes that belong to the ordering service will have addresses from this namespace. Lastly, the organization level domain is an organization's namespace. This namespace is a subdomain of the top-level domain. Each organization will allocate an address to their "peers" and "certifying authority" services from this subdomain.

In Fig. 2, taken from [14], "example.com" is the top-level domain under which is the orderer subdomain, "orderer.example.com." At the same level as the orderer is an organization's subdomain, like "org1.example.com" for Organization 1, "org2.example.com" for Organization 2, and likewise. Each organization will then provide a fully qualified domain name to their services from their subdomain, like "ca.org1.example.com" for the certifying authority service of Organization 1 or "peer1.org1.example.com" for a peer in the Organization 1.

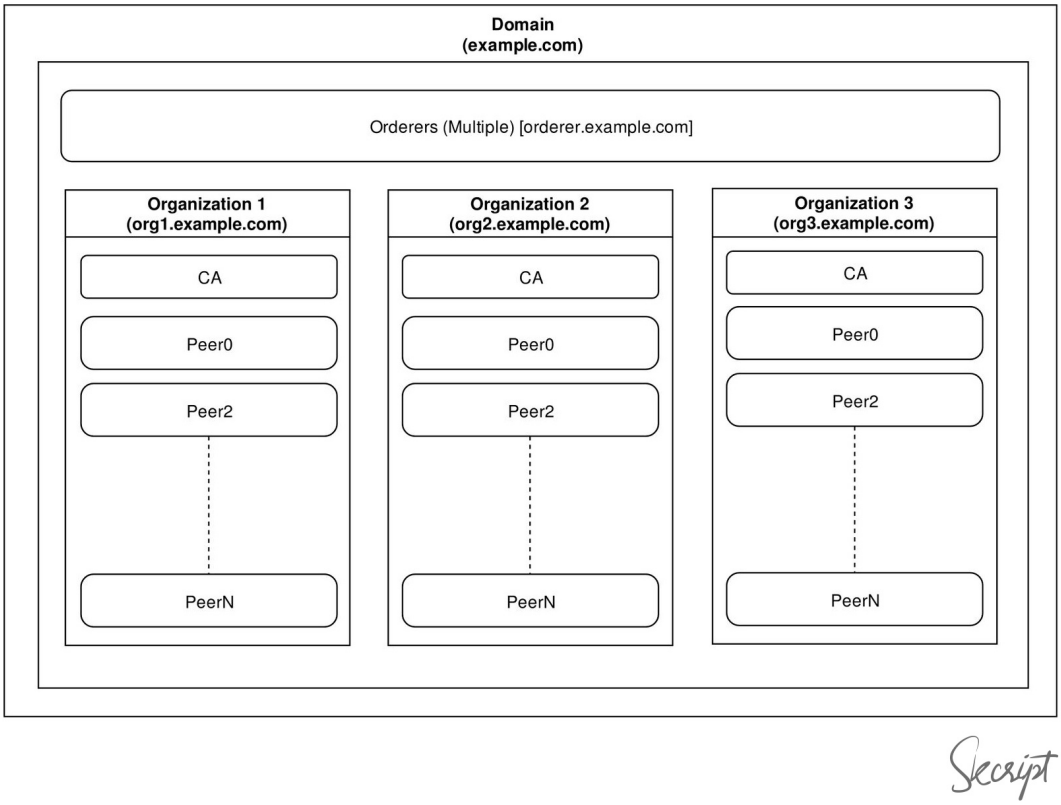


Fig. 2: Domains in Hyperledger Fabric. (Signature by author)

2.3.2 Architectural Components

The smart contract is known as the *chaincode* in Hyperledger Fabric. It contains the transaction logic that represents some business logic. A *Channel* is the link joining the members of the blockchain network before they can execute transactions among themselves. Peer nodes join a channel, a chaincode is then installed on that channel, and an endorsement policy is defined before transactions can be executed. Each member of the channel is authenticated before joining the channel.

The *Membership Service Providers (MSP)* component of Hyperledger Fabric is the mechanism responsible for user authentication. They act as Certificate Authorities who

generate and validate certificates. There can be more than one MSP in a network, and each organization can have its MSP to validate the identity of its peers.

Hyperledeger Fabric sacrifices some decentralization in exchange for high performance by having an *ordering service* that determines the order in which blocks are added to the blockchain. It can be made up of a single (solo) or multiple nodes (kafka or raft based).

Configuration Blocks are the initial blocks generated for a blockchain. They only consist of metadata for the blockchain like the genesis block, channel configuration, consensus requirements. An *Endorsement Policy* is specified while instantiating a chaincode on a channel. It specifies the consensus requirements for a valid transaction.

The *Gossip Protocol* specifies the protocol for broadcasting a block. An elected leader peer is defined for each organization that gets blocks directly from the orderer and is responsible for updating all peers under its jurisdiction. The leader maintains a health-checked connection with the ordering service.

2.3.3 Transaction flow in Hyperledger Fabric

According to the official documentation [5], Hyperledger Fabric has an "execute-order-validate" model for transactions. This essentially means that the transaction process can be divided into three distinct phases.

Typically, a transaction in Hyperledger Fabric can be divided into the following three phases as illustrated by Fig. 3 taken from [15]:

- Proposal phase: all transactions start with a client application creating a transaction proposal and sending it to the endorsing peers. Endorsing peers are a subset of all peers in a network that executes the chaincode with the parameters mentioned in the proposal and sign those proposals with their private key and sends it back to the client.

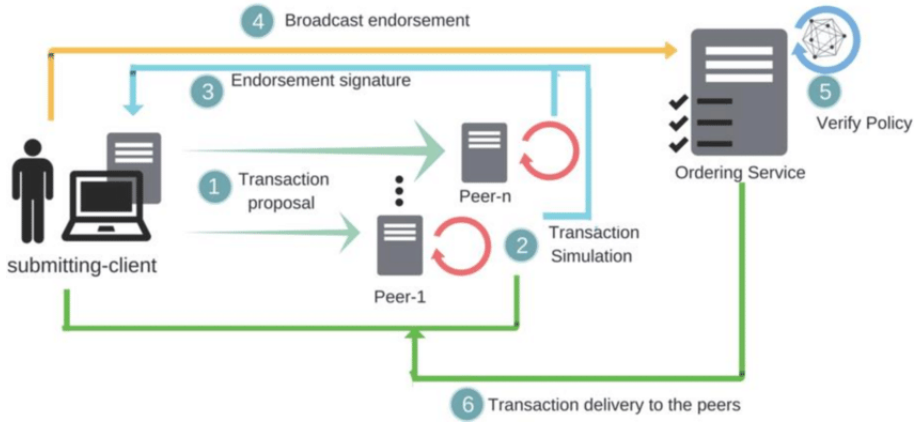


Fig. 3: Outline of transaction flow in Hyperledger Fabric.

- Endorsement phase: in this phase, the client collects the endorsed proposals. Once the client has received the required number of endorsements, based on the endorsement policy on the channel, the client sends the endorsed transaction to be grouped into a block to the ordering service. Once the ordering service receives the endorsed transactions, it verifies the signatures on the endorsements and then creates the next block to be added to the blockchain. The decision of when to cut a block is based on three factors. First is the *batch time out*. It is the maximum amount of time a transaction can be held before cutting a block. Second is the *batch size*. It is the actual size of a block. Hyperledger Fabric allows the setting of preferred size and absolute maximum size for a block. If the size of a transaction is greater than the absolute max batch size, that transaction will not be added to the blockchain. Last is the *maximum number of transactions* that can be grouped in a block.
- Commit phase: once the orderer creates the block, it signs it and sends it to the leader peers connected to it. The leader peers then broadcast the block to all the peers under their jurisdiction. The peers then commit that block to their local copy of the ledger.

Fig. 4 presents a typical three-phase transaction flow as described previously. This figure was taken from the official documentation [5].

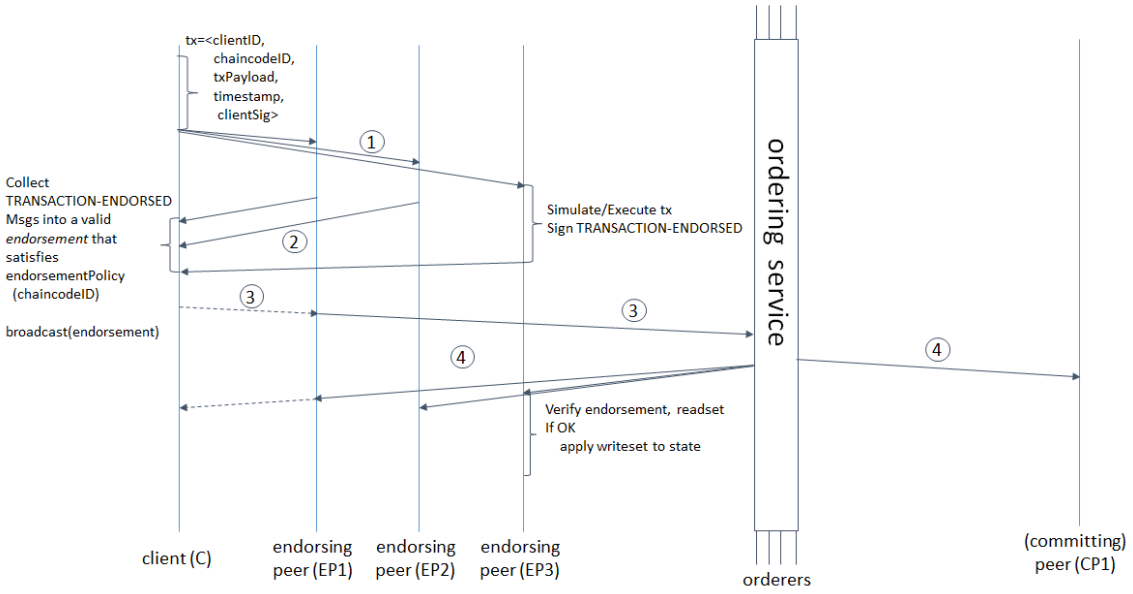


Fig. 4: Typical transaction flow in Hyperledger Fabric.

2.4 Advantages of a blockchain-based OPTN

The author finds three significant advantages of a blockchain-based OPTN. First, it is easier to audit. Antipova, in [16], states, "The government audit is designed to vouch for the reliability of the financial statements, not the soundness of the finances they portray." Blockchains can help government auditors have access to immutable audit logs, which can be easy to access. Using a blockchain-based OPTN would automatically generate reliable logs in the form of timestamped and cryptographically linked blocks. This system can increase the public's trust in the organ transplant system and might increase the public's willingness to become organ donors.

Second, since a blockchain is a peer-to-peer network, there is no dependency on a centralized source of truth or authority. All nodes in the network have their copy of the ledger and have to reach a consensus to add new blocks to the blockchain. Even though

there are some centralized components in some blockchain frameworks, like the ordering service in Hyperledger Fabric, they perform mechanical tasks. Any tampering in those tasks would be easily detected, thus reducing any impact of an attack on those components. This decentralized nature of blockchain eliminates the single point-of-failure problem.

Last, since a blockchain-based organ procurement system would not require a trusted governmental organization for its legitimacy, it does not need to be restricted by national borders. An international blockchain-based OPTN can have hospital members from various countries and the logic to match organs.

2.5 Limitations of a blockchain-based OPTN

Three significant limitations were identified, viz. quantum computing, negative social perception, and lack of a medical data interoperability standard.

First is quantum computing, which may seem like something from science fiction, but it poses a severe threat to any blockchain. As suggested by Yang and et al. in [17], bitcoins are vulnerable to quantum computing. One of the strengths of blockchain technology comes from encryption such that it is computationally infeasible to decrypt the blocks. When and if quantum computing becomes commercially viable, it can be used to defeat the encryption on the blockchain, thus losing the confidentiality of data. However, this can be overcome by the accurate selection of the encryption algorithm. As noted in the same source, cryptosystems like “McEliece” and “New Hope” are not vulnerable to quantum computers.

The second limitation is the social perception of blockchain. Medical and governmental organizations are known to be slow in adopting new technologies. On top of that, with all the negative publicity blockchains have been getting, due to its use by bad actors for illicit purposes, have only compounded this problem. John Oliver in his multiple Emmy winning show, “Last Week Tonight” while covering blockchain

technology mentioned "... I'm not saying that every blockchain company is [expletive] what I am saying is in a speculative mania it can be incredibly hard to tell which companies are for real ...". Social perception will dissuade policymakers from using blockchain for this purpose. Only strong scientific evidence can convince policymakers to consider this project seriously.

Lastly, the lack of a widely used standard for keeping medical records makes the automatic donor-candidate matching difficult. Mertz, in [18], throws light on the issue of moving medical data from one clinic to another. Challenges like digital data formats and protocols make it difficult to share medical data between different medical institutions. However, in a blockchain-based organ procurement system, if a hospital wants to participate in the network, it must have the ability to conform to the data format acceptable to the blockchain's chaincode. The existence of such a standard for the medical data could have benefits for patients admitted for reasons unrelated to organ transplants as well. It can help all patients by letting them move their health records from one healthcare provider to another.

Zang, Kaiwn, and Hans, in [19], provided a proposal template comprising of a series of questions and was used to design the prototype in this project. On the same theme, Tyler, in [20] provides a taxonomy of utility concepts to determine whether a particular application will benefit from a blockchain implementation. Some lesser-known blockchain consensus protocols, like the Stellar consensus protocol, are introduced by Shankar, Sindhu, and Sethumadhavan in [21]. Mertz, in [18], proposes a three-tier approach for designing a blockchain application. These tiers are the presentation layer, the middle layer, and the final layer, which represents the user frontend, off-chain computation-backend, and the blockchain-based backend, respectively. In [22], Zhang, and et al. warn about the potential challenges of a blockchain-based medical system's compliance with the Health Insurance Portability and Accountability Act (HIPAA). A

fascinating medical record-keeping system that uses bitcoins as payment for moving health records is introduced in [23]. Lewis in [24] provides a curated list of thirty exciting innovations that use blockchain technology, including one about buying beer.

3 METHOD

OrganChain is the prototype blockchain-based OPTN created for this study. This prototype was built using Hyperledger Fabric Framework. The prototype has four organizations, each corresponding to a UNOS permanent member type. These organizations are “Hospitals,” “Histocompatibility Labs,” “Organ Procurement Organizations,” and “General Public.” This section describes the domain hierarchy, the network topology, the chaincode, and the environment setup steps of the OrganChain prototype. The author’s first attempt at creating a prototype for a blockchain-based OPTN was [25]. However, the approach taken in this work is entirely independent of the previous work.

3.1 OrganChain Domain Hierarchy

The OrganChain prototype has “organ.com” as the top-level domain under which the “orderer.organ.com” subdomain is used for the ordering service. “histocompatibility.organ.com,” “gp.organ.com,” “opo.organ.com,” and “hospital.organ.com” are the subdomains used for the organizations. In turn, each organization contains subdomains for their peers. For example, “peer0” and “peer1” in the “gp” organization are identified by “peer0.gp.organ.com” and “peer1.gp.organ.com,” respectively. This hierarchy is illustrated in Fig. 5.

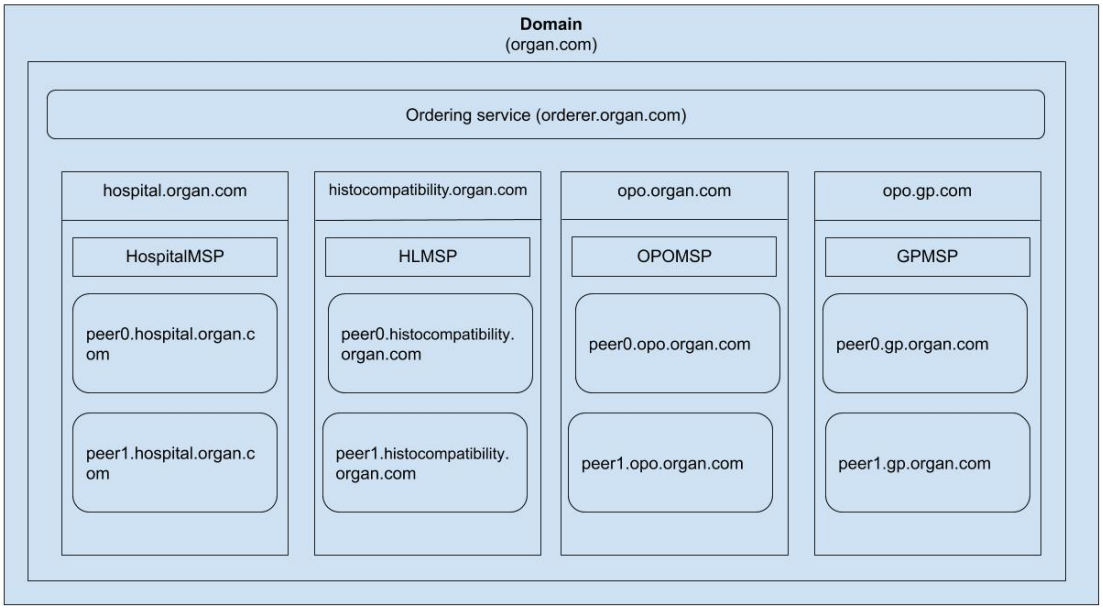


Fig. 5: OrganChain domain hierarchy.

3.2 OrganChain Network

The network topology of OrganChain is represented by Fig. 6. All the peer nodes are docker containers running in a docker network called `organ_chain_network`. Each organization has two peer nodes called "peer0" and "peer1," and thus, there are eight peers in the network. There is an additional orderer peer which runs the ordering service. All the peers are connected to the same channel called the `organ_channel`. The "peer0" of each organization is the designated anchor peer for its organization. The configurations and steps required to bring up this network topology are explained in section 3.4.

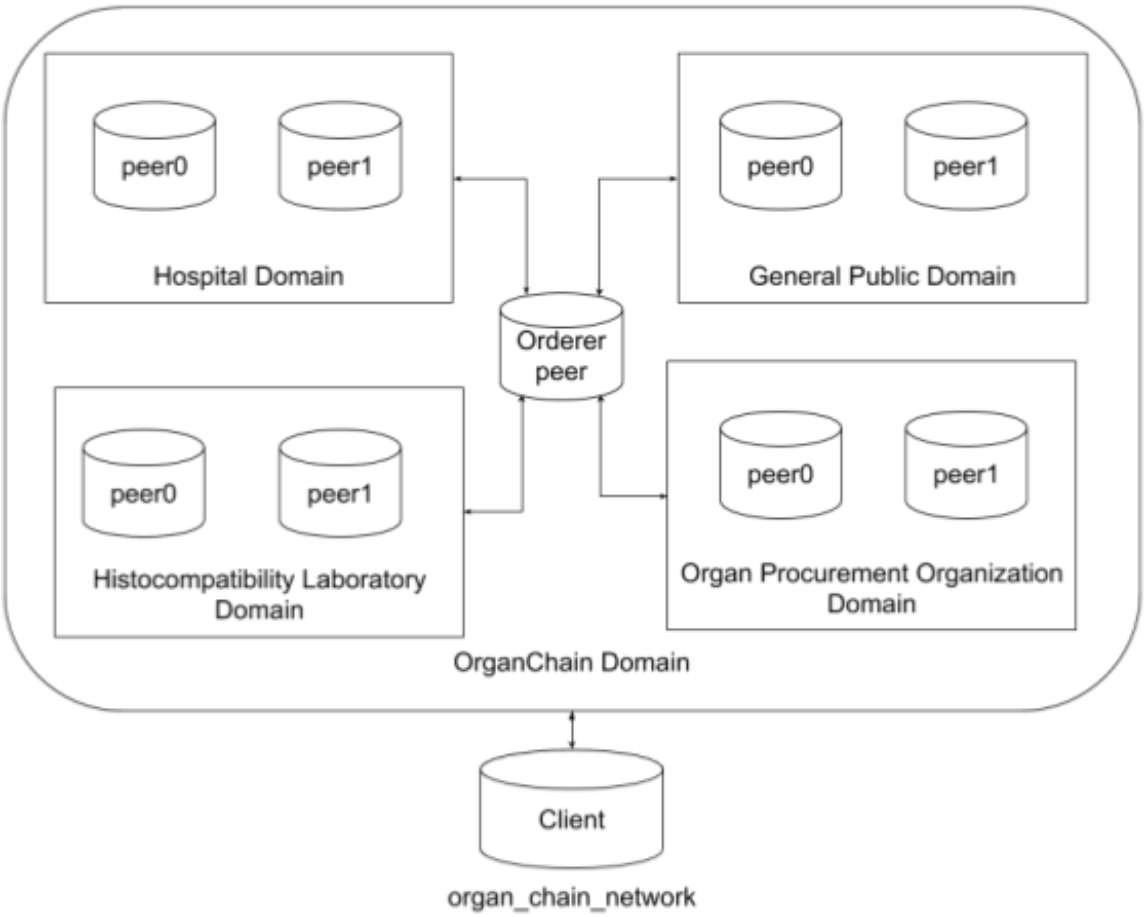


Fig. 6: OrganChain network.

3.3 OrganChain Chaincode

The chaincode comprises JavaScript methods, which represent logic behind a transaction. The “fabric-shim” library was used to write the chaincode. The transactions are used to create objects for donated organs and candidates. In the chaincode, there are helper methods that are used to update and read the states of the organs and candidate objects. The organ and candidate objects have three attributes. The first is a unique identifier, the second is the type of the organ, and the last is the medical data. The medical data are stores in a JavaScript Object Notation (JSON) object created using the Self-Defining Text Archive and Retrieval (STAR) file provided by UNOS, available in the

author's project repository [26]. For this thesis, simulated medical data were created for only the intestine organ type and used for experiments. Two variants of the chaincode were created.

3.3.1 Scheme A: Matching Inside the Blockchain

The *initOrgan* and *initCandidate* transactions were used for the matching process. The initOrgan transaction was used to create a new organ. When invoked, it creates a list of all available candidates and tries to match the new organ using this list of candidates. When the medical data of the candidate and the donated organ have the same value sixty-five percent of the time for the list of attributes in their medical data field, it is considered a match. The candidate then becomes a recipient. The initCandidate transaction is similar to initOrgan transaction but for a candidate. When invoked, it creates a list of all available organs and then tries to find a match with the same criteria.

3.3.2 Scheme B: Matching Organs Outside the Blockchain

The assumption for this approach was that, since the integrity of the data is guaranteed cryptographically, the compute-intensive matching could be taken out of the chaincode. Since chaincode is executed at least the same number of times as required by the consensus algorithm, moving the compute-intensive matching process out of the chaincode could give a significant performance boost in the propagation time of a block.

In this scheme, the initOrgan and the initCandidate methods simply created the candidate and organ object. A Python script was run to find a matching candidate-organ pair. Then, a *transferOrgan* transaction was invoked to match a candidate-organ pair. This transaction would change the state of these two objects and finish the matching process.

3.4 OrganChain Environment

This section describes the steps that were followed to run an instance of the experiment OrganChain network. For each experiment, the parameters were changed

accordingly in the steps described in this section. All relevant code is available in the author's GitHub repository [26].

Initially, the experiments were run on the author's personal computer inside a VM. The VM was created using the Virtual Box hypervisor. However, the data acquired from that setup had a large number of anomalies, and sound conclusions could not be drawn from them. Therefore, the experiments were run on a VM provisioned on the Google Cloud Platform. The VM specifications are in Table 1.

Table 1: Specification of the Experiment VM

Resource	**Value**
CPU	8 Core vCPU
Memory	30 GB
Storage	25 GB SSD
Operating System	Ubuntu 18.04

Before running the commands to bring up a network, the following three kinds of configuration files were created in this order.

1) Configuration files for creating cryptographic material: This includes creating certificates and key files for the organizations, i.e., Hospital, General Public, Histocompatibility Laboratories, OPOs as well as the orderer organization. This configuration file is written in the YAML format. A command-line interface (CLI) tool called "cryptogen" was used to create these, and this YAML formatted configuration file is given to it as an input. The output of this is the cryptographic material in the format expected by Hyperledger Fabric.
2) Configuration files to create channel artifacts: This includes the creation of the genesis block, a channel creation transaction block, and an anchor peer creation block for each of the organizations. Another CLI tool called "configtxgen" was used to create these artifacts. These are the configuration blocks that only store metadata, as explained in the previous section.

3) Docker compose files for infrastructure: These files are "infrastructure as code" and are written in YAML format. They describe the containers that run to create the network, what docker images these containers instantiate, the environment variables these containers contain, the ports and endpoints these containers expose, the attached volumes to these containers, and other infrastructure-related information. The docker-compose command takes this file as input along with some flags and to bring up the infrastructure.

To bring up an instance of the OrganChain network, a series of commands need to execute. Each step with its associated commands are described below:

1) Generate cryptographic material for all peers.

```
rm -rf crypto-config
cryptogen generate --config=./crypto-config.yaml
```

2) Create the genesis block for OrganChain.

```
export FABRIC_CFG_PATH=$PWD && mkdir channel-artifacts
configtxgen -profile OrganChainOrdererGenesis \
    -channelID organ-sys-channel \
    -outputBlock ./channel-artifacts/genesis.block \
```

3) Create channel transaction artifacts for `organ-channel`.

```
export CHANNEL_NAME=organ-channel  && configtxgen \
    -profile OrganChainChannel  \
    -outputCreateChannelTx \
    ./channel-artifacts/channel.tx  \
    -channelID $CHANNEL_NAME
```

4) Create configurations for anchor peer of each organization. Repeat this for all organizations.

```
configtxgen -profile OrganChainChannel \
    -outputAnchorPeersUpdate \
    ./channel-artifacts/HospitalMSPanchors.tx \
    -channelID organ-channel -asOrg HospitalMSP
```

5) Run docker-compose to create network components.

```
export COMPOSE_PROJECT_NAME=''
docker-compose -f  docker-compose-cli.yaml up -d
```

6) Create `organ-channel` from the client container.

```
docker exec -it cli bash
export CORE_PEER_MSPCONFIGPATH=/../Admin@gp.organ.com/msp
export CORE_PEER_ADDRESS=peer0.gp.organ.com:7051
export CORE_PEER_LOCALMSPID="GPMSP"
export CORE_PEER_TLS_ROOTCERT_FILE=/../peer0.gp.organ.com/tls/ca.crt
export CHANNEL_NAME=organ-channel
peer channel create -o orderer.organ.com:7050 \
    -c $CHANNEL_NAME -f ./channel-artifacts/channel.tx --tls \
    --cafile /../msp/tlscacerts/tlsca.organ.com-cert.pem
```

7) Join all peers to `organ-channel`. Execute this from all peer containers.

```
peer channel join -b organ-channel.block
```

8) Update the anchor peers of all organizations.

```
peer channel update -o orderer.organ.com:7050 \
    -c $CHANNEL_NAME -f ./channel-artifacts/HospitalMSPanchors.tx \
    --tls --cafile /../tlscacerts/tlsca.organ.com-cert.pem
```

9) Install the `organcc` chaincode on the `organ-channel`.

```
peer chaincode install -n organcc -v 1.0 \
    -l node -p /opt/gopath/src/github.com/chaincode/
```

10) Instantiate the `organcc` chaincode with an endorsement policy.

```
peer chaincode instantiate -o orderer.organ.com:7050 --tls \
    --cafile /../tlscacerts/tlsca.organ.com-cert.pem \
    -C organ-channel -n organcc -l node -v 1.0 \
    -c '{"Args":["initOrgan","123", "heart", "This is donor info"]}' \
    -P "OR('HospitalMSP.peer','OPOMSP.peer','HLMSP.peer','GPMSP.peer')"
```

3.5 OrganChain Analysis

A Jupyter Notebook was used to invoke the chaincode after the network was entirely created, as mentioned in the preceding paragraph. This notebook contains code to invoke the read and write transactions using the `peer chaincode invoke` command. The matching transactions were explicitly called only for Scheme B. After invoking a fixed number of transactions, the docker logs of all the peer and orderer containers were stored in a file using the same notebook. Then the entire network was destroyed.

Custom Python scripts were written to analyze the logs created by the docker container representing nodes in the blockchain network. Each line in the log is represented as a JSON object. Each of these JSON objects has `log` and `time` key, which stores the log message, and its timestamp was done, respectively.

During the log analysis, the code looks for the log messages with `Received Block [`, `Validated Block [` and `Committed Block [` strings. The timestamps on these log messages are of importance to the research. These times are used to determine the propagation time for each block and to plot graphs. To determine the propagation time for a particular block (X), a list containing the times at which each node received that block is created, RT_X. A similar list is created for commit times of that block, CT_X. The difference between the smallest value in the receive time list and the largest value in the commit time list is the propagation time for that block.

$$PropogationTime(X) = min(RT_X) \sim max(CT_X)$$

4 EXPERIMENTS AND EVALUATIONS

The network was created thirty-two times with sixteen different combinations of the four experiment parameters, as mentioned below. To investigate the effect of each experiment parameter, four different values of each parameter were considered to generate a combination. Then each combination was run with both the schemes of the chaincode. Each run is referred to as a sub-experiment in this document. This section contains a detailed evaluation of the results from these experiments.

4.1 Experiment Parameters

Transactions are grouped in blocks that are broadcasted throughout the network. The number of times a block is created has a direct impact on the performance of any blockchain-based application. In Hyperledger Fabric, the first three of the following are responsible for the decision of when to cut a block. Hence, experiments with different combinations of these values were run to determine their impact. The transaction rate experiment was conducted for apparent reasons.

- Batch Time Out: It is the maximum amount of time (in seconds) the ordering service waits to create a block. The values of two seconds, five seconds, ten seconds, and fifteen seconds were used to run experiments.
- Preferred Max Block Size: Measured in kilobytes, this is the size at which the ordering service creates a new block regardless of other factors. The experiments were run with the preferred max block sizes of eight kilobytes, sixteen kilobytes, and thirty-two kilobytes.
- Endorsement Policy: This configuration of consensus policy determines the number and type of endorsements a client needs to submit to the ordering service to get a particular transaction committed to the ledger. The “ALL AND,” “ALL OR,” “2 OutOf ALL,” and “3 OutOf ALL” endorsement policies were used. The “ALL AND” endorsement policy requires at least one peer from each organization to endorse a

transaction. Conversely, the “ALL OR” requires an endorsement of only one peer from any of the organizations. “2 OutOf ALL” and “3 OutOf ALL” require one peer from two and three out of all the organizations, respectively.

- Transaction Rate: It is the number of seconds between two consecutive transactions. The transaction rates of one transaction per one second, two seconds, five seconds, and fifteen seconds were used.

4.2 Test Load

For each of the thirty-two sub-experiment, fifty candidate objects and fifty organ objects were created using the `initCandidate` and `initOrgan` transactions. Then those fifty candidates and fifty organs were read using the `readCandidate` and `readOrgan` transactions. Essentially, in every sub-experiment, one hundred write-operations and one hundred read-operations were performed. For sub-experiments running with chaincode in Scheme A, this translated to two hundred transactions in total since the matching occurs by the chaincode. However, for Scheme B, each write-operation of an organ or candidate requires a read of all the available candidates or organs, respectively. These additional read-operations significantly increase the number of transactions and consequently the number of blocks as well.

4.3 Observations

This section contains graphs plotted from the data collected from the logs of docker containers that represented the nodes in the experiment setup. Then the observations derived from these graphs are described. For each experiment, a table containing the values of all the parameters’ combinations is presented, followed by the graphs plotted using the data.

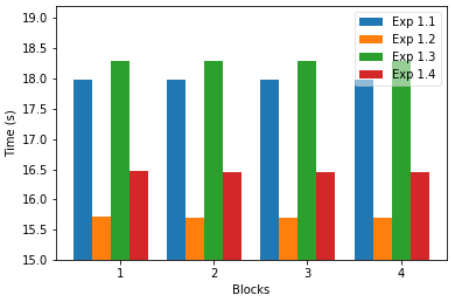
Experiment 1 (max batch time out): The batch time out values of two seconds, five seconds, ten seconds, and fifteen seconds were taken while keeping the preferred max block size, endorsement policy, and sleep between transactions fixed to eight kilobytes,

"All AND" and ten seconds respectively. All four combinations of values were run using both the chaincode schemes, as describes above.

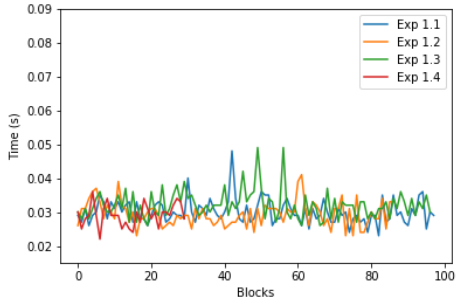
Table 2: Experiment 1 (Max Batch Time Out) Experiment Parameters

	Batch Time Out	Pref. Max Block Size	Endorsement Policy	Sleep Between Transactions
Exp 1.1	2 sec	8 KB	ALL AND	10 sec
Exp 1.2	5 sec	8 KB	ALL AND	10 sec
Exp 1.3	10 sec	8 KB	ALL AND	10 sec
Exp 1.4	15 sec	8 KB	ALL AND	10 sec

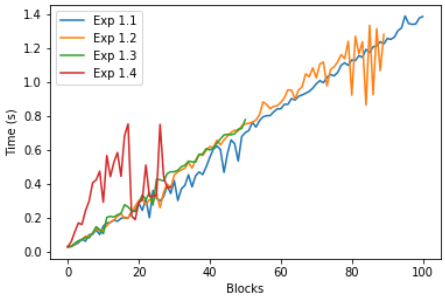
Experiment 1 (max batch time out) observations: The max batch time out has a direct impact on the number of blocks generated. In both schemes, the number of blocks decreases with an increase in batch time out, which does not have a significant impact on the propagation time of the blocks containing the transaction from only the read-operations. Propagation times of these blocks varied only within the margin of error. On the other hand, the write-operations showed a trend of linear increase for all the sub-experiments in Scheme A. For Scheme B, the propagation time for blocks with write-operations is higher than that of Scheme A since a write-operation in Scheme B may include a large number of read-operations. Fig. 7 contains graphs plotted from data generated from this experiment. Also, the differences in the propagation times for the configuration blocks for this set of sub-experiments were insignificant.



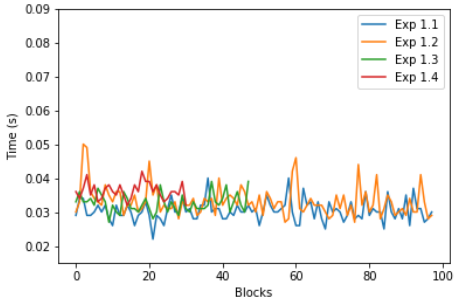
(a) Propagation time of configuration blocks.



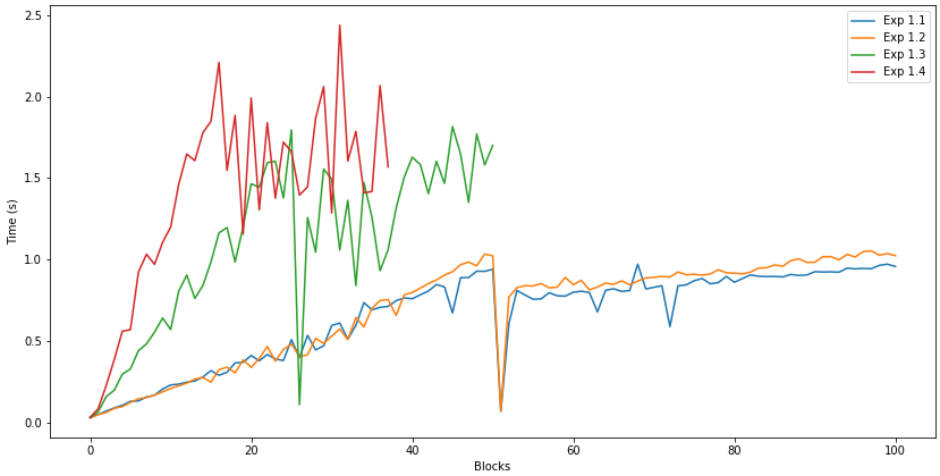
(b) Propagation time of read-operation blocks (Scheme A).



(c) Propagation time of write-operation blocks (Scheme A).



(d) Propagation time of read-operation blocks (Scheme B).



(e) Propagation time of write-operation blocks (Scheme B).

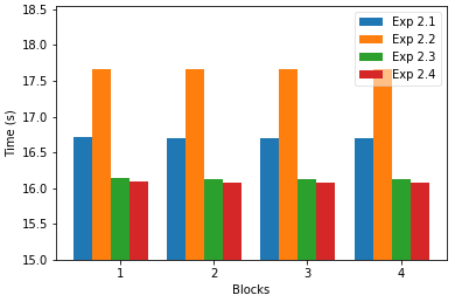
Fig. 7: Results of experiment 1 (max batch time out).

Experiment 2 (max block size): The preferred max block sizes of eight kilobytes, sixteen kilobytes, and thirty-two kilobytes were used with the batch time out, endorsement policy, and transaction rate of ten seconds, “All AND,” and one per two seconds respectively for the first three sub-experiments. The fourth sub-experiment was done with thirty-two kilobytes of the preferred max block size with “ALL AND” endorsement policy, but with a transaction rate of five per second. This information is tabulated in Table 3.

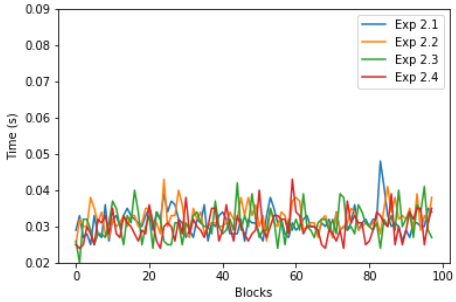
Table 3: Experiment 2 (Max Block Size) Experiment Parameters

	Batch Time Out	Pref. Max Block Size	Endorsement Policy	Sleep between transactions
Exp 2.1	10	8 KB	ALL AND	2 sec
Exp 2.2	10	16 KB	ALL AND	2 sec
Exp 2.3	10	32 KB	ALL AND	2 sec
Exp 2.4	10	32 KB	ALL AND	5 sec

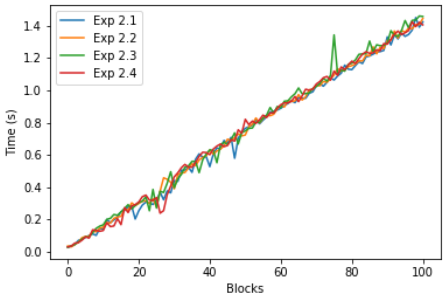
Experiment 2 (max block size) observations: First, the deviation between the propagation times of configuration blocks for the sub-experiments was minuscule. Second, the propagation times of the blocks containing transactions for the read-operations are identical. This regularity suggests that within the bounds of these experiments, the preferred max block size does not affect the read operations. Finally, the write-operation in Scheme B took three and a half times more blocks to create the fifty organ objects, and fifty candidate objects. This increase is because each write-operation in Scheme B triggers a read of all available organs or candidates. This behavior also explains the wavy pattern in the graph of the propagation times for the write-operation. The peaks represent the blocks containing write transactions and the troughs representing the propagation times of the blocks with only the read transactions. Fig. 8 contains graphs plotted from data generated from this experiment.



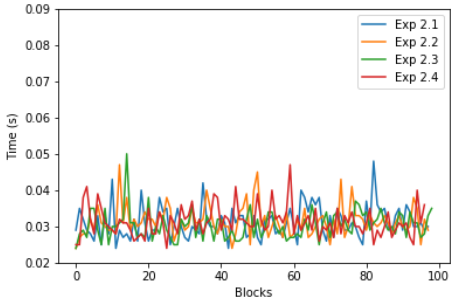
(a) Propagation time of configuration blocks.



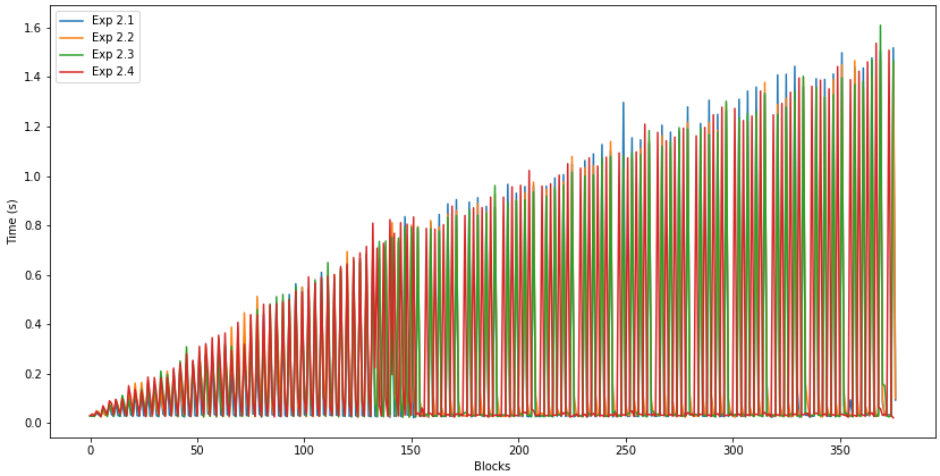
(b) Propagation time of read operation blocks (Scheme A).



(c) Propagation time of write operation blocks (Scheme A).



(d) Propagation time of read operation blocks (Scheme B).



(e) Propagation time of write operation blocks (Scheme B).

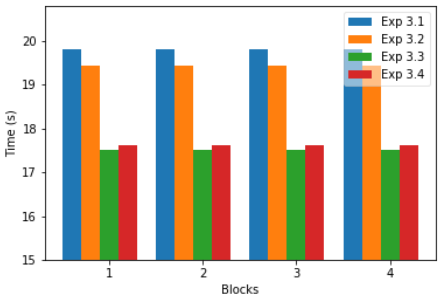
Fig. 8: Results of experiment 2 (max batch time out).

Experiment 3 (endorsement policy): The four endorsement policies that were studied by this experiment in the order of being most restrictive to least restrictive were “ALL AND,” “ALL OR,” “2 OutOf ALL,” and “3 OutOf ALL”. For all the eight sub-experiments, the batch time out, preferred max block size, and a transaction rate of ten seconds, sixteen kilobytes and one transaction per five seconds were taken respectively. This information is tabulated in Table 4.

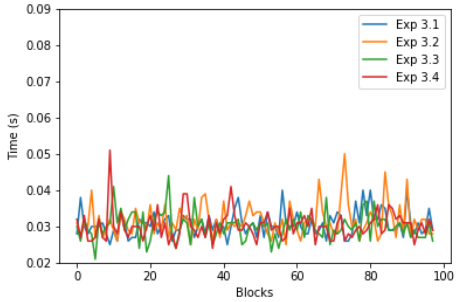
Table 4: Experiment 3 (Endorsement Policy) Experiment Parameters

	Batch Time Out	Pref. Max Block Size	Endorsement Policy	Sleep between transactions
Exp 3.1	10	16 KB	ALL OR	5 sec
Exp 3.2	10	16 KB	ALL AND	5 sec
Exp 3.3	10	16 KB	2 OutOf ALL	5 sec
Exp 3.4	10	16 KB	3 OutOf ALL	5 sec

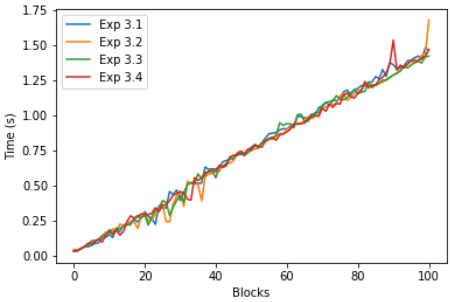
Experiment 3 (endorsement policy) observations: First, the propagation times of the configuration blocks were similar. It implies that the Endorsement policy does not have a significant effect on them. Second, the blocks with the transactions of read-operations were also within the margin of error for both the schemes. Last, the propagation times for the write-operation blocks showed a similar increase in all four endorsement policies for both the schemes. Also, again, Scheme B took three and a half times the number of blocks than Scheme A. This observation implies that, under the current setup, the endorsement policy does not have a significant impact on the propagation time of the blocks containing transactions for the read and write operations. Fig. 9 contains graphs plotted from data generated from this experiment.



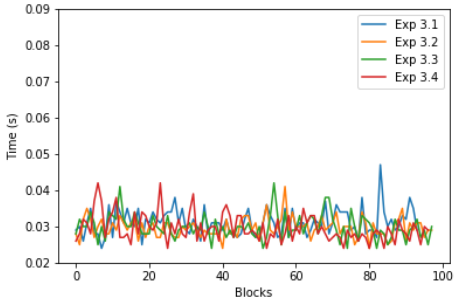
(a) Propagation time of configuration blocks.



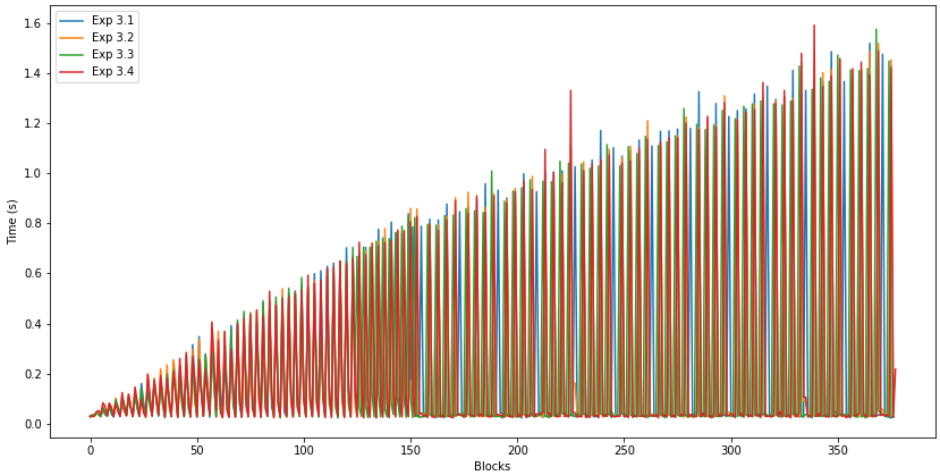
(b) Propagation time of read operation blocks (Scheme A).



(c) Propagation time of write operation blocks (Scheme A).



(d) Propagation time of read operation blocks (Scheme B).



(e) Propagation time of write operation blocks (Scheme B).

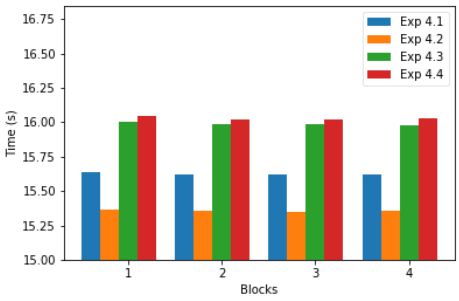
Fig. 9: Results of experiment 3 (endorsement policy).

Experiment 4 (transaction rate): In this experiment, the transactions were issued at the rate of one transaction per one second, two seconds, five seconds, and ten seconds. A batch time out, preferred max block size and endorsement policy of ten seconds, thirty-six kilobytes, and "ALL OR" was taken respectively. This information is tabulated in Table 5.

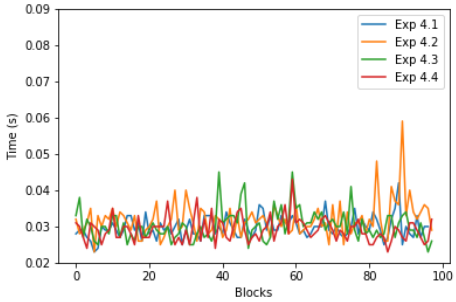
Table 5: Experiment 4 (Transaction Rate) Experiment Parameters

	Batch Time Out	Pref. Max Block Size	Endorsement Policy	Sleep between transactions
Exp 4.1	10	36 KB	ALL OR	1 sec
Exp 4.2	10	36 KB	ALL OR	2 sec
Exp 4.3	10	36 KB	ALL OR	5 sec
Exp 4.4	10	36 KB	ALL OR	10 sec

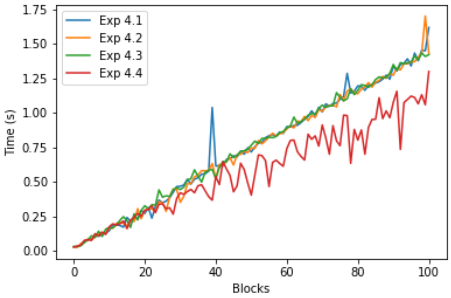
Experiment 4 (transaction rate) observations: First, the propagation times of the configuration blocks did not show a significant variance. Second, the variance in the propagation times of the read-operation blocks was within the margin of error for all eight sub-experiments. This view suggests that the current experiment setup can handle all the transaction rates equally. Last, as for the propagation time of the write-operation blocks, the sub-experiment 4.4 for Scheme A performed better than the rest of the sub-experiments. This result may occur because the transaction rate was equal to the "batch time out". This setup resulted in one transaction per block, which in turn lead to a smaller payload than that of other sub-experiments in this experiment. Fig. 10 contains graphs plotted from data generated from this experiment.



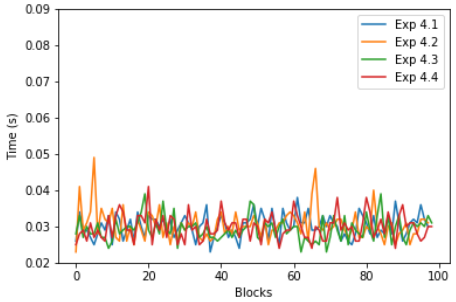
(a) Propagation time of configuration blocks.



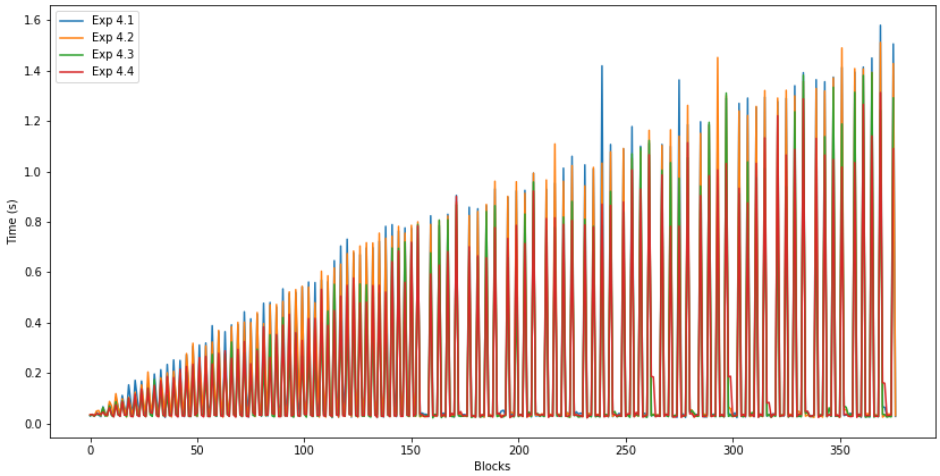
(b) Propagation time of read operation blocks (Scheme A).



(c) Propagation time of write operation blocks (Scheme A).



(d) Propagation time of read operation blocks (Scheme B).



(e) Propagation time of write operation blocks (Scheme B).

Fig. 10: Results of experiment 4 (transaction rate).

5 FUTURE WORK

Throughout this research, the author realized the great need for software process automation firsthand. The author recommends the use of the following tools to automate various repeatable tasks, which are better done programmatically than by hand, Terraform, Ansible, ad-hoc Python scripts, and managed blockchains.

First, using Terraform for configuration management has various benefits such as documentation of infrastructure, quickly reproducible infrastructure, and automatic deployment of all the bits of infrastructure automatically ranging from an "auto scaling group launch configuration" to a "single firewall rule." Using such a tool would allow for easy and quick deployment and tear down of various configurations.

Second, Ansible can be used as a task runner and would enable quick and automatic installation of all binaries, such as Hyperledger Fabric CLI, that are required to run the applications. Various Ansible playbooks can be created containing several plays representing a set of logically related tasks like bringing up a network, archiving logs of servers, tearing down a network, and so on. These plays, in turn, can have more fine tasks such as generating cryptographic material, creating channels, and creating anchor peers. These tasks can either be performed using command-line instructions or ad-hoc Python scripts.

Third, ad hoc Python scripts can be used to perform a repeatable set of tasks. In particular, Python scripts can be used to divide the command reference file in [26] to smoothen the network creation process. In this case, Python is preferred over Shell scripts because much of the code is required to execute on remote servers, and Python libraries like Paramiko would easily facilitate it.

Last, using a managed blockchain service can significantly reduce the amount of work needed to bring up the network. Amazon Web Services (AWS), in its arsenal of services, have now started to provide blockchain as a service as well under the banner of "Amazon

Managed Blockchain." With this service, creating a blockchain network becomes just as easy as clicking a few buttons, like any other product provided by AWS. When combined with AWS's other services like "Elastic Cloud Compute" and "AWS Key Management Service," a large number of combinations of experiment setups can be easily created. Moreover, each time an entirely new environment is created, keeping each experiment isolated. When the author started this study, AWS did not offer this service.

Apart from process automation, it is necessary to run more experiments on a more extensive network. The current prototype uses a virtual network created using docker networks. However, the real-world scenario could have nodes of different kinds connected by networks of variable throughput and latency. Given the constraints of this research, it was reasonable to use a virtual network on a single host. Nonetheless, in the real world, the network has a significant impact on the overall performance of the application. Hence, experiments must be run on different network configurations to study its effects on application performance. The use of automation tools, as mentioned in the previous point, would be prudent.

Also, the chaincode needs to be improved to reflect the actual application. Organ matching is a very complicated process. It is affected by numerous factors such as a candidate's waiting time as well as their medical compatibility. On top of that, each organ has its matching criteria. The chaincode of the current prototype provides a good starting point to create a complete matching algorithm. A prototype with a more elaborate chaincode than the one created for this study will be helpful to convince policymakers in favor of using blockchains for the organ transplant system.

6 CONCLUSIONS

The current Organ Procurement and Transplant Network is a centralized system that maintains a ledger of organ transplant candidates along with their medical information and then matches them with a donated organ based on a set of well-codified algorithms and practices. This study finds that blockchain technology can decentralize this process. On top of that, it can also bring potential advantages like increased transparency, global organ procurement, and easier auditing.

While conducting this research, the author perceived that the most complex, error-prone, and time-consuming aspect of this research was designing and bringing up of the nodes of the blockchain network for the OrganChain prototype. Accordingly, the author proposes the use of various automation tools for future work. In contrast, the implementation of the business logic, represented by chaincode, was much simplified by the abstractions provided by libraries in Hyperledger Fabric.

The observations of all the thirty-two sub-experiments conducted in this research leads to the conclusion that for the given set of experiment environments, Scheme A of executing the chaincode is superior because it generated fewer blocks for a write-operation as compared to a write-operation in Scheme B. Therefore, this study endorses leaving the "match run" process in the chaincode. Additionally, the read-operations seem unaffected by any of the parameters under the scope of this study. However, the write-operations performed best when the "batch time out" period was close to the transaction rate.

It is safe to assume that medical institutions, especially government-backed, would be extremely circumspect in using any new technology. This empirical study concludes in favor of continued research in using blockchains for this purpose. Nevertheless, there are significant unanswered questions on issues such as data privacy and performance, requiring more investigation.

Literature Cited

[1] "Organ donation statistics," https://www.organdonor.gov/statistics-stories/statistics.html (accessed Oct. 31, 2019).

[2] "UNet," https://unos.org/technology/unet/ (accessed Oct. 31, 2019).

[3] Organ procurement and transplantation network bylaws. Accessed: Oct. 31, 2019. [Online]. Available: https://optn.transplant.hrsa.gov/governance/bylaws/

[4] Organ procurement and transplantation network policies. Accessed: Oct. 31, 2019. [Online]. Available: https://optn.transplant.hrsa.gov/governance/policies/

[5] "Hyperledger fabric, a blockchain platform for the enterprise," https://hyperledger-fabric.readthedocs.io/en/release-1.4/ (accessed Oct. 31, 2019).

[6] "Organ donation myths, debunked penn medicine," https://www.pennmedicine.org/updates/blogs/transplant-update/2019/march/myths-about-organ-donation (accessed Oct. 31, 2019).

[7] M. Yacoub, N. Banner, A. Khaghani, M. Fitzgerald, B. Madden, V. Tsang, R. Radley-Smith, and M. Hodson, "Heart-lung transplantation for cystic fibrosis and subsequent domino heart transplantation." *The Journal of Heart Transplantation*, vol. 9, no. 5, pp. 459–66, 1990.

[8] Centers for Medicare and Medicaid Services (CMS), "Medicare and medicaid programs; conditions for coverage for organ procurement organizations (opos). final rule." *Federal Register*, vol. 71, no. 104, p. 30981, 2006.

[9] "Matching donors and recipients," https://www.organdonor.gov/about/process/matching.html (accessed Oct. 31, 2019).

[10] C. Harvey and R. Weigel, "Transplant2mongo: Python scripts that insert organ procurement and transplantation network (optn) data in mongodb," *Journal of Open Research Software*, vol. 7, no. 1, 2019.

[11] "Your unet security administrator helps keep unos systems safe," https://unos.org/news/your-unet-security-administrator-helps-keep-unos-systems-safe/ (accessed Oct. 31, 2019).

[12] J. J. Eilenberg, "Chartgeist," *Wired*, no. Jul/Aug 2019, p. 16, Jul 2019.

[13] K. Sultan, U. Ruhi, and R. Lakhani, "Conceptualizing blockchains: Characteristics & applications," *arXiv preprint arXiv:1806.03693*, 2018.

[14] V. Raj, "Hyperledger fabric architecture: Explained in detail," https://www.skcript.com/svr/understanding-hyperledger-fabric-s-architecture/ (accessed Oct. 31, 2019).

[15] O. Choudhury, H. Sarker, N. Rudolph, M. Foreman, N. Fay, M. Dhuliawala, I. Sylla, N. Fairoza, and A. Das, "Enforcing human subject regulations using blockchain and smart contracts," *Blockchain in Healthcare Today*, 03 2018.

[16] T. Antipova, "Using blockchain technology for government auditing," in *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, June 2018, pp. 1–6.

[17] D. Yang, J. Gavigan, and Z. W.-O. Hearn. "Survey of confidentiality and privacy preserving technologies for blockchains". Accessed: Oct. 31 2019. [Online]. Available: https://www.r3.com/reports/survey-of-confidentiality-and-privacy-preserving-technologies-for-blockchains/

[18] L. Mertz, "(block) chain reaction: A blockchain revolution sweeps into health care, offering the possibility for a much-needed data solution," *IEEE Pulse*, vol. 9, no. 3, pp. 4–7, May 2018.

[19] Z. Kaiwen and J. Hans-arno, "Towards dependable, scalable, and pervasive distributed ledgers with blockchains," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 07 2018, pp. 1337–1346.

[20] T. D. Smith, "The blockchain litmus test," in *2017 IEEE International Conference on Big Data (Big Data)*, Dec 2017, pp. 2299–2308.

[21] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of consensus protocols on blockchain applications," in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Jan 2017, pp. 1–5.

[22] P. Zhang, M. A. Walker, J. White, D. C. Schmidt, and G. Lenz, "Metrics for assessing blockchain-based healthcare decentralized apps," in *2017 IEEE 19th*

International Conference on e-Health Networking, Applications and Services (Healthcom), Oct 2017, pp. 1–4.

[23] P. Tasatanattakool and C. Techapanupreeda, "Blockchain: Challenges and applications," in *2018 International Conference on Information Networking (ICOIN)*, Jan 2018, pp. 473–475.

[24] R. Lewis, "30 things you can do with a blockchain," https://medium.com/yope-chain/30-things-you-can-do-with-a-blockchain-b23b2ab39664 (accessed Oct. 31, 2019).

[25] C. Arora, U. Jain, M. Bhatt, and Y. Liang, "Organdonation-chain," https://github.com/SJSU272LabF17/OrganDonation-Chain (accessed Oct. 31, 2019).

[26] U. Jain, "Organchain," https://zenodo.org/badge/latestdoi/188463205 (accessed Oct. 31, 2019).