POLITECNICO DI TORINO

Master degree course in Computer Engineering

Master Degree Thesis

Blockchain applications to Supply Chain: an application to last-mile delivery

Supervisors
prof. Guido Perboli
engr. Daniele Manerba

Candidate
Vittorio Capocasale

December 2019

Summary

The blockchain technology is gathering increasing interest in many fields. This is due to its many properties, which make it suitable for many different applications. The blockchain technology belongs to the wider group of the distributed ledger technologies (DLT). The common core of the DLTs is the idea of storing the sequence of modifications a database is subject to, and not only its current state, similarly to what happens in the accounting world. For this reason the database is called ledger. The ledger is distributed among many peers, and each of them keeps a full copy of it. Each peer (node) is responsible to update only its own copy of the database, which is performed each time the node receive a transaction, and the global state of the ledger is considered to be what the majority of the nodes agrees upon. To ensure that the nodes reach a majority, some cryptographic techniques and a consensus algorithm are used to solve the conflicts that may arise ("forks"), as well as the order and authenticity of the transactions. For these reasons, the blockchain technology offers an high level of data security, redundancy and distribution and guarantees at the same time its immutability and authenticity, which makes the blockchain a viable solution to solve the double spending problem without relying on third-parties. For this reason, it was initially adopted in the financial world, as proved by the Bitcoin blockchain. However, the technology application can be easily extended to other fields, and in particular to the supply chain. In fact, one of the main limits the supply chain companies have to face is the lack of a standard interface to exchange information and a way to authenticate it: data gets corrupted or simply lost in translation when exchanged between different information systems. The blockchain technology has the potential to reduce the paperwork processing costs and time, to reduce the amount of fake products introduced in the market and to offer a way to identify them, to allow to track the origin of a product from the producer to the consumer, and thus to limit the issues of a production defect. Finally, if coupled with the IoT devices, it could even create new businesses models and revolutionize the old ones. However there are also many issues which could limit the actual adoption of the blockchain technology. In particular, two are the main factors that must be taken into account. From an economic point of view, the savings related to the adoption of the blockchian must justify the risks of adopting an immature technology. From an efficiency point of view, the blockchain

must be able to guarantee the same throughput of the "legacy to be" systems. For this reason, various methods to evaluate the performance of the blockchian solution have been proposed, even if it is challenging to establish a reliable metric that takes into account all the possible aspects which may influence the measure. One of the metrics mostly used is the number of transactions that can be performed in a second (TPS). Of course, among other factors, this metric is dependant on the complexity of the transactions used in the system. To cope with this problem, the Hyperledger Sawtooth framework is used to implement a more complete supply chain and a simplified one. The former presents entities and operations that can be considered common to many real-world use-cases and mechanisms to handle shared objects and the delegation of operations. The system is used to measure the TPS value reached in three different use cases. The results are provided in figure 4.5 and figure 4.6. The results are compared with the ones of the other supply chain that can perform only very simple (almost CRUD) transactions. This could give a hint about the complexity of the complete one. The tests performed should be replicated on bigger networks and using a more sophisticated hardware in order to offer results closer to what could be a real world application.

Contents

Introduction

The blockchain technology was introduced in 2008 [1], providing a way to perform payments without the mediation of a trusted third party (e.g. a financial institution) [2]. The innovative capability of shifting the trust from an external entity to the blockchain itself in a complete transparent way has gathered increasing interest for this technology in many other fields. However, many are also the problems and the challenges that this technology presents, usually linked to security, efficiency and scalability which may hugely limit its practical applications [3].

Supply chain is one of the environments which may benefit from the blockchain application: often times each organization has its own way to organize and store the information about the products exchanged, and tracking them from the producer to the final consumer is nearly impossible. This makes easier the dispatch of counterfeit products and, at the same time, harder to identify all the items affected by a manufacturing defect. It is consequently clear how crucial is data transparency in a supply-chain environment, and the blockchain technology could satisfy this necessity[4].

In this context, it is important to understand the applicability of the blockchain solution to the supply chain by exploring its supported workload and its economic advantages and disadvantages. This work is focused on the first of such investigations. In particular, a blockchain-based supply chain has been implemented, defining actors and operations common in many use cases, with the goal to measure the quantity of data the chain is able to process. Both the system dimension and the hardware resources used in the tests are not comparable with what could be a real world wide distributed system, and for this reason also the results of transactions executed on a simpler chain are provided, in order to give a hint about the performance decay due to the business logic complexity.

The following part of this work is organized as follows:

- the chapter 1 provides an overview of the blockchain technology, introducing the main concepts and underlining the pros and cons of the blockchain solution;
- the chapter 2 introduces the supply chain and its relation to the blockchain and some current applications of the blockchain to the supply chain are discussed; then the Hyperledger Sawtooth blockchain is briefly described, pointing out

its features and the motivations of the choice of this framework for the implementation part of this work;

- the chapter 3 describes the implemented systems, their actors, operations and standard operations' flow;

- the chapter 4 provides information about the tests performed and the results obtained;

- the chapter 5, after a brief summary of the results obtained in the previous chapter, describes possible future developments of the work conducted so far.

Chapter 1

Blockchain as a distributed ledger technology

At the beginning, this chapter presents some definitions of technologies that opened the path to the blockchain evolution, then introduces the main blockchain concepts. The description is done in a general way, and specific implementations may present some differences. Reading this section, however, should provide the necessary information to have a complete understanding of the blockchain functionalities, issues and merits. The final section is dedicated to the smart contract concept and its relation with the blockchain.

1.1 Database, blockchain and shared distributed ledger

Storing information has always been a fundamental activity in human history, as testified by the revolutions linked to it: the invention of writing is commonly considered as the separator between prehistory and history, the invention of printing with movable type "accelerated the slow evolution of script and literacy" [5]. In modern times information format keep migrating from analogue to digital [6], and databases are the most common way of storing digital data.

At the beginning of the database era, the relational model was considered the proper way of organizing database data [7], [8], by guaranteeing the consistency and availability properties (CA) of the CAP theorem [8](figure 1.1). In fact, originally databases were deployed as centralized entities, and consequently they were not affected by network partition problems. Later on, due to scalability issues, the increasing quantity of web data and the Object-Relational Impedance Mismatch [8], [9], other technologies have emerged, each providing its own way of structuring data (documents, graphs, etc.). These technologies are known as "NoSQL":

> "Carlo Strozzi coined this term in 1990 to refer to databases that do not use SQL as their query language, and Eric Evans later used it to refer to nonrelational and distributed databases" [8].

Distributed databases, in particular, have the common denominator of giving priority to the partition tolerance property of the CAP theorem over the consistency or availability ones [8], [10]. Creating databases where the data is replicated in many storage devices and it is handled by different processes simultaneously over a best-effort network [11], in fact, can always suffer of network partitioning problems, to the point that it is not considered possible to have a CA system in a distributed environment [8], [10].

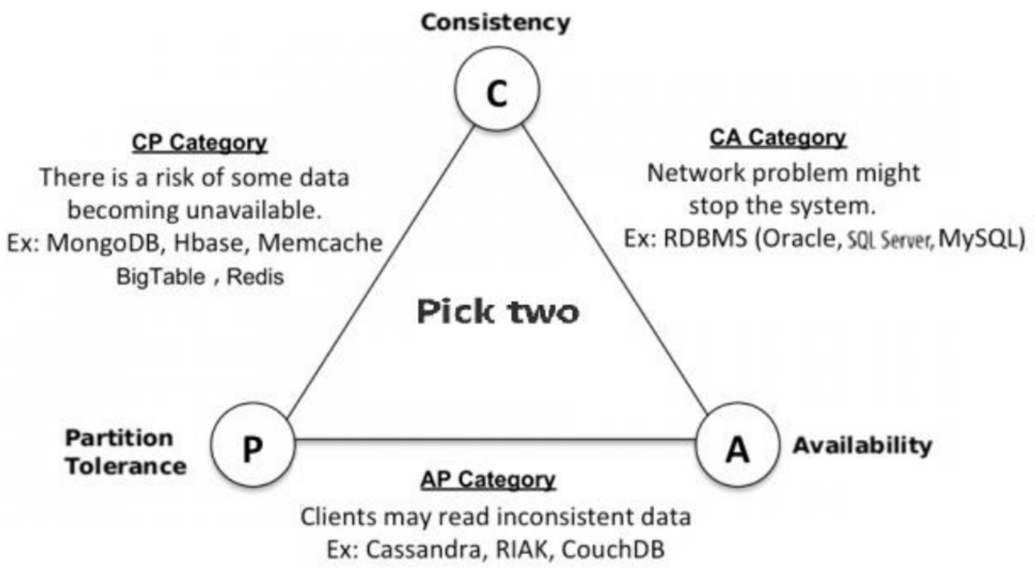
Consistency
C
CP Category
There is a risk of some data becoming unavailable.
Ex: MongoDB, Hbase, Memcache BigTable , Redis
CA Category
Network problem might stop the system.
Ex: RDBMS (Oracle, SQL Server, MySQL)
Pick two
Partition Tolerance
P
A
Availability
AP Category
Clients may read inconsistent data
Ex: Cassandra, RIAK, CouchDB

Figure 1.1. Cap theorem and databases [12].

A shared distributed ledger or distributed ledger technology (DLT) is similar to a distributed database, because data is replicated in many storage devices, but each of them is controlled by a different entity:

> "Distributed ledger technology refers to the ability for users to store and access information or records related to assets and holdings in a shared database (i.e., the ledger) capable of operating without a central validation system and based on its own standards and processes. DLTs differ from standard accounting ledgers in that they are maintained by a distributed network of participants (known as “nodes”) rather than a centralized entity. Another common feature of DLTs is the use of cryptography as a means of storing assets and validating transactions."[13]

Shared distributed ledger may seem a too much complicated name to describe a database, but it is well suited:

ledger: because of the particular way data is managed. Each copy of a DLT database is composed of records. Each record represents a modification of the database data (e.g. creation, update or deletion of an object), but does not modify the data itself. Because the current state of the data is expressed as a sequence of modifications, the sequence must be immutable and ordered [1]. For this reason, records can be added but never updated [14]. Records are added through transactions, which are the way the nodes interact with the blockchain. This particular way of recording data is derived from the accounting world [15], and for this reason the database is named ledger;

shared: because the database does not belong to only one entity, but to a group of them (nodes) and each of them does not trust the others. Each node is responsible to menage only its own copy of the database and may submit transactions to update its state. In this way, if one node misbehave, it does not affect the general database, but only its own copy. At the same time, the system works only if the majority of the nodes behaves correctly: if in the system all the copies differ from one another, is impossible to detect the only node which node is not misbehaving. For this reason, shared is not only the control of the database, but also the interest into keeping the database correctly updated;

distributed: because the database is composed by multiple copies. In particular the blockchain is a peer-to-peer distributed system, because the nodes share the system control in an even manner, and because communication "occurs directly between peers instead of through a central node. Each node stores and forwards information to all other nodes" [14]. For example, when a transaction is submitted, it should be broadcasted to all the nodes, so that each of them may update its own database. Consequently, DLTs present all the merits and all the issues of other peer-to-peer distributed systems.

Often times, nodes in a DLT may be "light", when they limit themselves to submit transactions and they do not store a copy of the database, or "full", in the case they participate to all the others DLT activities. This approach is used, for example, in IOTA [16].

One of the main reasons for the increasing interest in the DLTs and their initial employment in the financial world is their ability to solve the double spending problem without the necessity to rely on a trusted third party. As the name anticipates,

[1]Because a modification may be dependent from a previous one, their sequence must be ordered and immutable. The constraint on the order, however, could be a little relaxed by considering that transactions with different reading and writing sets could be performed in parallel.

the double spending problem consists in the attempt to use the same coin twice. Because the records in the ledger are ordered and because all (or the majority of) the nodes have the same copy of it, it is always possible to decide which transaction spent the coin and to reject the other one [2]. A blockchain is a particular DLT, and its characteristics are described in the next section.

1.2 Blockchain: an overview

1.2.1 The DLT called blockchain

According to the definition given in the previous section, in a DLT is mandatory:

- to enforce record immutability, because the ledger is a sequence of modifications;
- to enforce record order, because a different order in a sequence of modifications may produce different results;
- in case two or more nodes have a different sequence of records, to decide which one is correct;
- as a consequence of the low level of trust among the nodes, it must not be possible to impersonate other nodes or to forge transactions. While it is not required a mapping between a transaction submitter (identity) and the entity that submits the transaction [2], it is still necessary to prevent the possible malicious behavior of some transaction submitters (e.g. block any attempt to sell something belonging to someone else).

A blockchain groups transactions together, before applying them to the ledger. Groups are called blocks. The ledger is consequently a series of blocks. Each block does not contain only the transactions, but also some other data to address the immutability and order constraints of the records. In particular, each block contains the hash of the previous block [2], [18], as shown in figure 1.2. In this way, if a block is modified, its hash would not match the one stored in his subsequent block. At the same time, if two block are swapped, the hash they contain would not be the hash of their previous block. For these reasons, each block is linked to its previous, like in a chain, and this explains the name blockchain.

Blocks are created by the nodes. Each node fills a block with the transactions it is aware of, then broadcasts the block to the other nodes. In a distributed

[2]Each entity may use multiple identities to submit transactions (pseudonymity [14]) and some studies even suggest ways to achieve anonymity by decoupling and mixing senders/receivers information [1], [17].

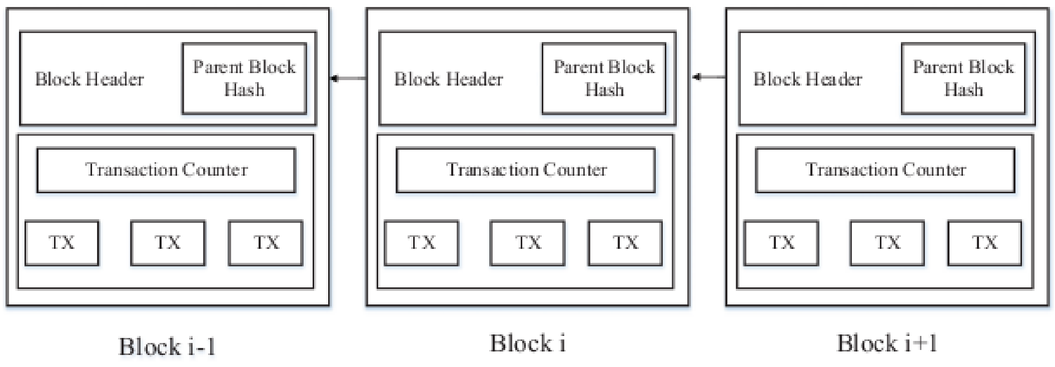

Figure 1.2. Three blocks chained [18].

system, however, it is not trivial to agree on the block sequence. A node may receive/create a block and add it to its ledger while another node does the same with another block. This creates a "fork". A graphic representation of a fork is reported in figure 1.3. On which fork-branch the system should agree is decided by a consensus algorithm. Many consensus algorithms exists, and they are listed in the next subsection. It is worth noting that, when a node creates a new block, it is not explicitly stated which transactions it will include in the block. A node may give preferences to the transactions that include a reward for the node who publish them. Finally, transactions must be digitally signed, and this prevents problems like node impersonation and transaction forgery [2], [18].

1.2.2 Consensus algorithms and their implications

> "A blockchain based system is as secure and robust as its consensus model" [19].

The consensus algorithm can be considered as the core of the blockchain technology: it must allow all the nodes to agree on the current ledger state, even if each node does not trust the information received by the others.

> "Achieving consensus in a distributed system is challenging. Consensus algorithms have to resilient to failures of nodes, partitioning of the network, message delays, messages reaching out-of-order and corrupted messages. They also have to deal with selfish and deliberately malicious nodes." [19].

In literature, this problem has been addressed as "The Byzantine Generals Problem", and it can be modelled as follows:

> "several divisions of the Byzantine army are camped out side an enemy city, each division commanded by its own general. The generals can

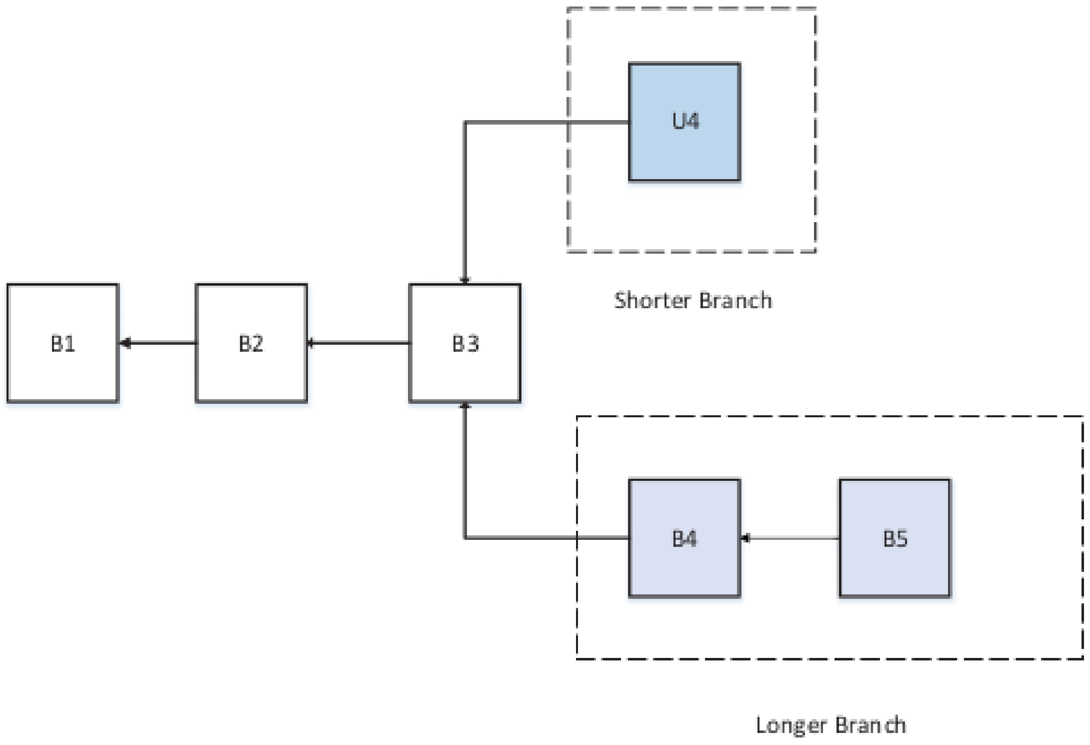

Figure 1.3. Example of a fork in a blockchain [18].

> communicate with one another only by messenger. After observing the enemy, they must decide upon a common plan of action. However, some of the generals may be traitors, trying to prevent the loyal generals from reaching agreement" [20].

The goal is to find an algorithm that guarantees that all the loyal generals take the same decision. A simplified version of the problem does not present traitors general, but only generals without messengers, so that they cannot participate in the decision-making process. Algorithms that solve this simplified problem are defined crash fault tolerant (CFT), while those that menage to solve the complete problem are called Byzantine fault tolerant (BFT). The first solutions to the complete problem were poorly suited for a real implementation. Things changed with the introduction of the Pratical BFT algorithm (PBFT):

> "...a new state-machine replication algorithm that is able to tolerate Byzantine faults and can be used in practice: it is the first to work correctly in an asynchronous system like the Internet and it improves the performance of previous algorithms by more than an order of magnitude" [21].

Nowadays blockchains use different consensus algorithms, each of them with its merits and issues. Follows a brief description of the most used.

Pratical BFT (PBFT)

PBFT is based on message exchanges among the nodes to decide the block to add to the ledger. Because the decision is taken before adding the block, forks are not possible. At the beginning of the algorithm and each time after a block is added to the ledger, a leader must be chosen, based on a round-robin algorithm. The leader collects the transactions, orders them and dispatches them to the other nodes. To order transactions and to be sure that they are committed in the same order independently from the correct behavior of the leader, three phases are required. Each node advances from a phase to the other after collecting united messages from the others nodes equal to 2/3 of the total number of nodes [18], [19], [21], [22].

This algorithm allows an high efficiency as a consequence of the lack of forks, however:

- it requires that the nodes know the number of participants in the network, so that each of them may know if the number of messages from distinct nodes is bigger than the 2/3 threshold. This implies that "every node must be known to the network" [18];
- it does not scale well, because of the number of messages exchanged [19].

Hyperledger Fabric uses (among others) a PBFT-based consensus algorithm [19].

Proof of Work (PoW)

PoW was the first consensus algorithm ever applied to the Blockchain technology. In case of a fork, nodes should prefer the longest branch, or work on either one of them until one becomes longer. The reason is that each time a block is added to the ledger, the node who creates it has to solve a mathematical challenge (mining). In particular, in the original PoW (Bitcoin Blockchain), among the non-transaction data present in a block, there is a "nonce". It is a number the node has to chose so that the block hash results smaller of a predefined threshold. Lowering the threshold is a way to increase the challenge difficulty, in order to adapt it to the varying size of the node network. Because the challenge can be resolved only with a brute-force computation by trying all the possible nonces, a longer chain means more work put into it, and it is a way to determine on what the majority of the nodes agrees, assuming a uniform distribution of the computational resources among them. While solving the challenge is a resource consuming task (time, electricity), verifying if a received block satisfy the challenge rules is instead very simple, because the nonce is known. Other PoW algorithms use a similar approach [2], [18], [19], [22].

The PoW is scalable and completely decentralized, as proven by the thousands of nodes composing the Bitcoin Blockchain, but:

- it has low efficiency, meaning that it allows few blocks to be published;
- it is highly resource consuming, in terms of electricity and time;
- it supposes a uniform distribution of the computing power. If a node(s) menages to gain more computational power then the remaining ones, it will have a bigger probability to solve the challenge before the others, thus gaining the possibility to selectively decide to stall some transactions and to perform a double spending. In case of a fork, in fact, all the nodes operating on the smaller branch will revert all the transactions until they reach the fork block, then they will apply all the transactions of the longer branch. Usually forks are resolved quickly, and a transaction can be considered committed after a few blocks are chained to the one containing the transaction itself, because, in order to revert it, it would be necessary to overcame the good nodes computational power. This is the reason this attack is known as the 51% attack [19], [22].

Proof of Stake (PoS)

> "Many blockchains adopt PoW at the beginning and transform to PoS gradually" [18].

PoS is a family of lottery based algorithms, used in blockchains with their own cryptocurrencies. The probability for a node to create the next block is proportional to the amount of coins it has and is calculated by a pseudo-random function. A 51% attack would be possible for a node who holds the 51% of the coins, but it would go to its own disadvantage, because it is damaging something he owns for the majority. In order to break the combination richness/power, many implementations of the PoS take into account other parameters in the selection of the next block creator as well. Depending on the implementation, forks are possible even if unlikely. PoS allows a good efficiency, it does not waste electricity and it is scalable. Its main drawback remains the combination richness/power that may lead to a centralized control of the system, other than some technical challenges that can be overcome with a proper design (e.g. the nothing at stake problem) [18], [19], [22].

Delegated Proof of Stake (DPoS)

DPoS tries to mitigate the combination richness/power of the PoS through an election mechanism: stakeholders can only vote for the node that will publish the next block. The vote weight is proportional to the wealth of the stakeholder and the probability for a node to publish the block is proportional to the weighted sum of the votes he has received. This mechanism is somehow similar to a representative democracy. The main advantages of this algorithm are the increased efficiency and scalability, as a consequence of the reduced number of nodes that takes part in the block creation process. However, there are concerns about the real mitigation of the combination richness/power and a more centralized control system [18], [22].

Proof of Elapsed Time (PoET)

It is a lottery based consensus algorithm that relies on a Trusted Execution Environment (TEE).

> "It guarantees the authenticity of the executed code, the integrity of the runtime states (e.g. CPU registers, memory and sensitive I/O), and the confidentiality of its code, data and runtime states stored on a persistent memory. In addition, it shall be able to provide remote attestation that proves its trustworthiness for third-parties. The content of TEE is not static; it can be securely updated. The TEE resists against all software attacks as well as the physical attacks performed on the main memory of the system. Attacks performed by exploiting backdoor security flaws are not possible" [23].

The current implementation (PoET-SGX) exploits the Intel TEE module (SGX), but it is also provided a version that runs on any other processor (PoET-CFT, only Crash Fault Tolerant, as the name states). Each node uses the TEE module to select a random amount of time to start a timer, and it has to wait until the timer expires to create a block. Because the code running in the TEE module should be impossible to temper, the node that will publish the next block is chosen in a fair way. Moreover, the TEE module provides a way for the other nodes to verify the correctness of the procedure. It may happen that two or more nodes extract a similar, low amount of time, end each of them tries to create a new block. In this case, a fork arises. The metric used for the fork resolution is the "aggregate local mean (a measure of the total amount of time spent waiting)" [24]. This algorithm has no theoretical downsides: it does not waste resources, it is decentralized and extremely scalable, and has a good efficiency [19], [25]. The only drawbacks are:

- the SGX module is produced by Intel only, making of Intel a trusted-third-party or, even worse, a necessary third-party;
- the SGX module is currently unreliable, considering that "instead of protecting users from harm, SGX currently poses a security threat, facilitating so-called super-malware with ready-to-hit exploits" [26];
- flows in the PoET implementation have been discovered [27]. However, it is hard to find software free from this type of problems.

Others

Many other algorithms have been proposed, usually named as Proof of Something. They all are somehow similar to the proof of Stake, but they differ in what determines the probability for a node to publish a block. Examples are the Proof of Capacity (based on the disk space committed), Proof of Importance (based on the number of transactions the node is involved in), Proof of Burn (based on the amount of cryptocurrency a node is willing to give up). All of these have different trade-off among waste of resources, efficiency, network control distribution and scalablity [28], [29].

1.2.3 Blockchian properties

Knowing that the blockchain technology is a DLT and that each blockchain may rely on a different consensus algorithm, comes to small wander that some properties are common to all the blockchain technologies, while other are consensus dependent:

type can be public, consortium or private. Public blockchains can be joined by anyone and it is possible to participate in the consensus and ledger maintenance. Often times they provide forms of pseudonymity/anonymity, as in the Bitcoin Blockchian. Private blockchains are handled by one organization only, that can apply its own policies and permissioning systems, although it is stated that private blockchian "is just a confusing name for a shared database" [30]. Consortium blockchains are a mix of the previous two: only some nodes are allowed participate to the consensus process, but they act as peers [18], [19], [31].

finality can be deterministic or probabilistic. It is deterministic when there is absence of forks, so that if a record is added to the ledger, it cannot be reverted. It is probabilistic if forks may arise, and consequently the network consistency is only eventual [19];

immutability is common to all the blockchain implementations, as a consequence of the ledger structure and the hash mechanism [18], [31];

autonomy refers to the possibility to submit transactions without the involvement of a trusted-third-party. [31]

decentralization is common to all the blockchain implementations, as a consequence of the peer-to-peer network. Some consensus implementations, however, may centralize the network control, transforming the system in a master-slave one [18], [31];

efficiency refers to the quantity of data flow the blockchain can handle [18], [19]. Various metrics exists, like counting the number of transactions committed per second (TPS). It is highly influenced by the consensus algorithm used, and its improvement is one of the main challenges of the blockchain technology;

currency orientation refers to the necessity of the blockchain to rely on a cryptocurrency, for example as a consequence of the consensus algorithm used (e.g. PoS), or as a reward to miners, to encourage them to perform the costly block mining [19];

scalability refers to the ability of the network to become larger without deterioration of the other properties. It highly depends on the consensus algorithm [18], [19];

anonymity/pseudonymity refers to the ability of the network to hide the real identity of a transactor (anonymity) or at least to conceal it (pseudonymity). It is a distinctive feature of some blockchains [18], [31];

transparency refers to the complete access to the data that all the nodes have [31], [32]. Of course, the use of cryptography and anonymity may limit the the level of transparency reached by a blockchain, and consortium/private blockchains may impose strong access-control over the ledger data;

trust model indicate if the nodes must be known or trusted, in order for the blockchain to work. PBFT, for example, requires nodes to be known [19];

adversary tolerance represents the maximum percentage of malicious nodes the system can tolerate without affecting the consensus. Because of the Byzantine Generals Problem, it must be lower than 1/3 of the total nodes, unless some constraints of the problem are relaxed [19], [20].

1.3 Smart Contract

"...systems which automatically move digital assets according to arbitrary pre-specified rules." [33].

"A smart contract is an automatable and enforceable agreement. Automatable by computer, although some parts may require human input and control. Enforceable either by legal enforcement of rights and obligations or via tamper-proof execution of computer code." [34].

As introduced in the above definitions, smart contracts are a way to automatize a contract. This may seem nothing new, and actually their history precedes the one of the blockchain [35]. The idea is as simple as powerful: enforcing legal contract application through an electronic system, in an automatic way [34]. In simple terms, a smart contract is just a program running on a computer, and here resides its potentiality: as a program, it may react to external inputs, and dispatch some information in case a particular condition takes place. Its application are many and more, limited only by the complexity of the contract itself: as pointed out by Nobel Prize Oliver Hart, contracts rarely cover all the possible outcomes of a situation, mainly because future events are predictable only partially, and often times contract-related controversies are solved by dialogues and compromises among parties [36]. Of course, this is not something that can be translated into a program. However, smart contracts are powerful enough to automate simple or repetitive actions such as the ones linked to bureaucratic paperwork. The scarce application they have experienced in the past is mainly a consequence of the lack of a platform to support them. Things changed with the introduction of the blockchain. In fact, smart contract code could run on the blockchain itself: since the blockchain environment is trusted and records are immutable, parties could agree on a contract without the involvement of a notary, for example. The contracts could automatically react to transaction submission that changes the status of contract-related data [33], [35]. Other advantages include the increased efficiency due to the automation and the reduced legal costs as a consequence of the absence of a trusted third party [37]. The versatility of the smart contracts is expressed by the number of fields in which they find application: voting systems [38], financial applications [33], automation of Internet of Things workflows [39] and identity management [40] are a few examples.

Chapter 2

Blockchain and supply chain

This chapter is focused on the relation between supply chain and blockchain. The first section gives a definition of supply chain and introduces some important related concepts, focusing on the importance of the information flows in the supply chain context. The second section discusses some of the supply chain current limitations and shows how the blockchain application can mitigate them. In the third section the dissertation is contextualized in the current status of integration between supply chain and blockchain. Finally it is introduced the Hyperledger Sawtooth blockchain, a project that well fits the supply chain needs.

2.1 Supply chain: an introduction

In order to discuss about the relation between blockchain and supply chain, it is first required to introduce the concepts of competitive advantage, supply chain and the related ones.

> "Competitive advantage grows out of value a firm is able to create for its buyers that exceeds the firm's cost of creating it. Value is what buyers are willing to pay, and superior value stems from offering lower prices than competitors for equivalent benefits or providing unique benefits that more than offset a higher price. There are two basic types of competitive advantage: cost leadership and differentiation."[41].

For the purposes of this work, the following definition of supply chain is considered well suited:

> "a set of three or more entities (organizations or individuals) directly involved in the upstream and downstream flows of products, services, finances, and/or information from a source to a customer" [42].

It is worth noting that this definition explicitly underlines the importance of the flow of information as a component of the supply chain, but it is also quite generic, in order to cope with the multiple transformations the supply chain undergoes. The process leading these changes is the supply chain management (SCM). The literature proposes many definitions of SCM, to the point that it "is actually trying to define two concepts with one term" [42]. For the purposes of this work, however, it is not particular relevant to enter into the details of the matter, and even one of the earliest definitions can be considered sufficient:

> "The objective of managing the supply chain is to synchronize the requirements of the customer with the flow of materials from suppliers in order to effect a balance between what are often seen as conflicting goals of high customer service, low inventory management, and low unit cost" [43].

Already in those early stages, it was underlined how one of the main goals of SCM was to improve supply chain integration (SCI) [43], [44], to the point of considering SCI as the core of SCM [45]:

> "supply chain integration is the alignment, linkage and co-ordination of people, processes, information, knowledge, and strategies across the supply chain between all points of contact and influence to facilitate the efficient and effective flows of material, money, information, and knowledge in response to customer needs" [45].

The relation between SCM and competitive advantage is well expressed by the following quote:

> "supply chain management (SCM) has become a potentially valuable way of securing competitive advantage and improving organizational performance since competition is no longer between organizations, but among supply chains" [46].

In more recent times, the development of many new technologies in the IT sector supported SCM toward a better integration of the information flow and a bigger amount of data shared in the supply chain system, providing many advantages, including a reduction of the inventory size, an increased delivery efficiency, a better capital utilization [47] and, more generally, a reduction of costs [48]. The impact that those technologies had on the supply chain was so decisive that the expression "digital supply chain" (DSC) was introduced in literature [49]. The adoption of the blockchain technology may lead to an ulterior revolution in the supply chain context.

2.2 Blockchain for supply chain

The real value the blockchain technology has to offer to the supply chain is the standardization of immutable and transparent information and the automation of its exchanges, all in a trusted environment, that may "transform the current trust-based theories in supply chain management" [50]. Despite the progresses in the supply chain information flow integration as a consequence of the IT technologies development, issues are still present, and their relevance increases with the number of entities involved in the system (actors) and their dispersion across the world [4], [51]. The reason is mainly linked to the heavy bureaucracy needed for international shipments and to the different ways the information is organized by each actor. This last problem cannot be overcome by the definition of a common interface to exchange data because of the low level of trust present among the actors [52]. Moreover,

> "the data sharing must be secured, distributed (e.g., for optimizing the subsystems locally) and with some automated actions related to the different regulations and negotiations" [53].

The supply chain limitations expressed above are particularly relevant in the following four use cases [4], and the introduction of the blockchain technology may be the way to overcome them:

bureaucracy/paperwork processing: the information standardization and the information exchange automation with the use of smart contracts may reduce paperwork processing time and cost. Moreover, it would limit the human interaction and thus its related errors [4], [49];

counterfeit products identification: the transparency and immutability properties of the blockchain technology may reduce the number of counterfeit products by either making their recognition easier or even by preventing their introduction in the materials and products flows [4];

tracking of products: the transparency and immutability properties of the blockchain technology may also increase the detail level of information of the products, which would make easier to track down contamination sources and the defects affecting them [4]. It could also create an additional differentiation competitive advantage for some actors, because costumers may easily discover important qualities of the products (like bio, green, Km0 etc.). For the same reason, it could discourage unethical or unsustainable actors' behavior [50];

> "Sustainability has been defined by the triple-bottom-line concept that includes a balance of environmental, social, and business dimensions when managing the supply chain" [50].

IoT integration: the blockchain technology in cooperation with the smart contract one could be used to automatize in a secure and reliable way the interaction of IoT devices [4], [39]. This could be very powerful due to the fact that IoT devices are often times the interface between the digital and the physical worlds [39].

While some of the possible benefits of the application of the blockchain technology to the supply chain have been described so far, there are also many barriers that may limit their entanglement, which is important to identify. To this purpose, the article "Blockchain technology and its relationships to sustainable supply chain management" is taken in great consideration because

> "is the first papers to clearly identify and categorise blockchain barriers in general, and those specific to the adoption of the technology for supply chain purposes" [50].

Following the distinction proposed in such an article, four main types of barriers are identified:

intra-organizational: these barriers are originated from challenges internals to each organization. They take into consideration the costs of the migration toward the blockchain solution, the lack of awareness and long-term commitment at management level, the lack of knowledge and expertise, as well as the proper policies for the usage of this new technology. They also take into account the reluctance in replacing working "legacy" systems, the hesitation due to the possible changes in the organizational hierarchy and the general lack of indicators, tools, and methods to evaluate the sustainability of the blockchain solution application. [50];

inter-organizational: these barriers are linked to the relationship among supply chain actors. They take into account the concerns about the information disclosure to non-trusting parties, the problems linked to different priorities, goals and culture that may reduce the level of communication and collaboration among the actors. Moreover, there are challenges linked to the supply chain transformation toward sustainability, which, as described above, in indirectly led by the enhanced traceability of the products [50];

system-related: these challenges are linked to the technology itself. They take into account its immaturity, its security flows, its immutability, which would make impossible to hide eventual errors and the difficulties of developing/purchasing new IT tools to fully benefit from the blockchain solution adoption. They also take into account the association between the blockchain and the cryptocurrencies used for illegal practices, and the bad perception this gives to the general public [50];

external: these barrier are linked to the interaction with entities not directly involved in the supply chain. They include the lack of governmental regulations, the lack of external investments and incentives toward sustainability, the scarce interest of companies in ethical practices, the market competition and uncertainty, which may not reward the investment in the blockchain technology [50].

A scheme representing the main barriers the blockchain technology has to overcome is shown in figure 2.1.

Summarizing, the adoption of the blockchain technology could radically change the supply chain. Some of these changes may benefit all the actors in the system, some only a few of them, some may even damage them. As a consequence, the interest of the companies may go against the investment in this new technology. The interest in a more sustainable economy, however, should push the governments to encourage the adoption of the blockchain.

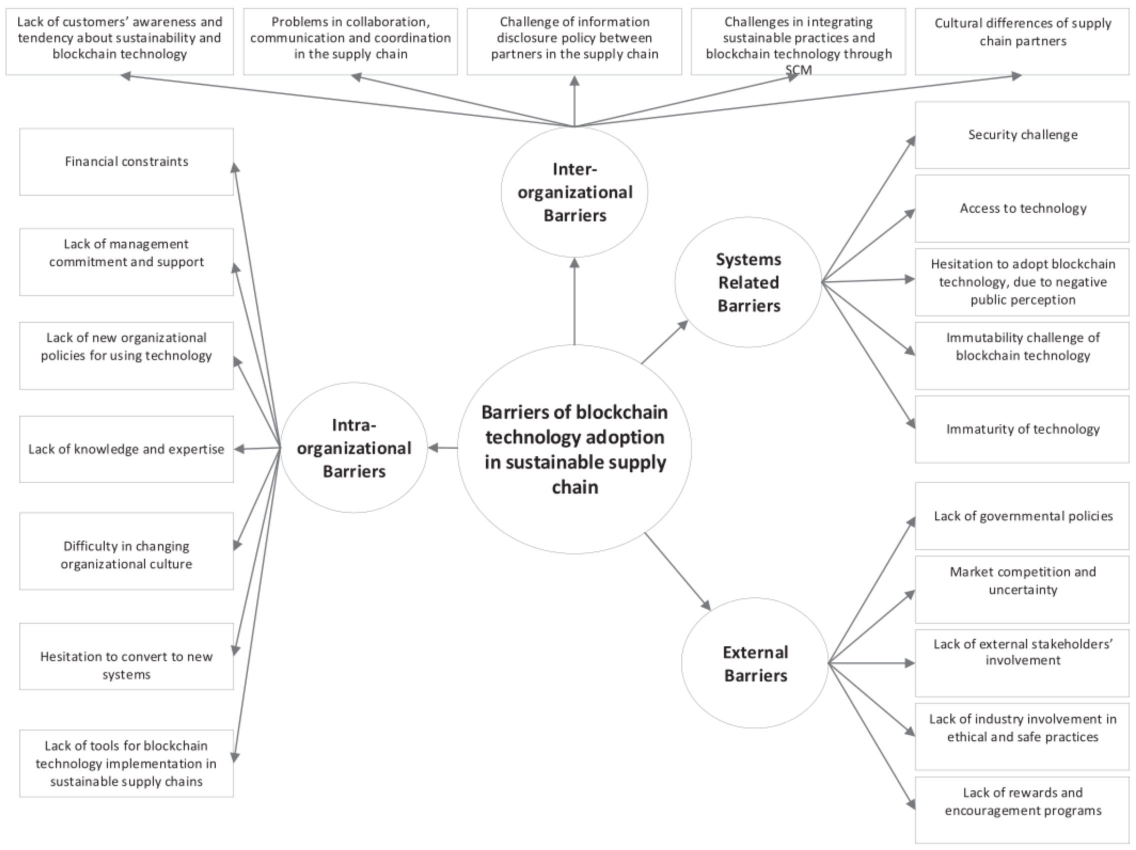
Lack of customers' awareness and tendency about sustainability and blockchain technology
Problems in collaboration, communication and coordination in the supply chain
Challenge of information disclosure policy between partners in the supply chain
Challenges in integrating sustainable practices and blockchain technology through SCM
Cultural differences of supply chain partners
Inter-organizational Barriers
Financial constraints
Lack of management commitment and support
Lack of new organizational policies for using technology
Lack of knowledge and expertise
Difficulty in changing organizational culture
Hesitation to convert to new systems
Lack of tools for blockchain technology implementation in sustainable supply chains
Intra-organizational Barriers
Barriers of blockchain technology adoption in sustainable supply chain
Systems Related Barriers
Security challenge
Access to technology
Hesitation to adopt blockchain technology, due to negative public perception
Immutability challenge of blockchain technology
Immaturity of technology
External Barriers
Lack of governmental policies
Market competition and uncertainty
Lack of external stakeholders' involvement
Lack of industry involvement in ethical and safe practices
Lack of rewards and encouragement programs

Figure 2.1. Barriers of blockchain adoption in the supply chain environment (with a focus on sustainability) [50].

2.3 Current applications

In literature, there are many papers describing the blockchain, its possible application to the supply chain and the related advantages and disadvantages, considering that, at the time of writing, it has been introduced by only a decade. However, the applications they describe are often times only hypothetical or cannot be fully disclosed due to confidentiality reasons; consequently there is a general lack of information regarding the applicability of the blockchain solution in terms of return on investment and supported workload. Consequently, more studies should investigate it in real cases scenarios, or at least in a simulation of those.

In the article "Blockchain in Logistics and Supply Chain: A Lean Approach for Designing Real-World Use Cases", the authors provide an interesting description of an application of the blockchain to a supply chain for the fresh food delivery. In particular they show the cost sustainability of the blockchain solution, with savings mainly related to the identification of unsafe storage conditions and a better usage of information about the expiration date of the products. Moreover, they underline how it is critical to identify the right processes to migrate toward the blockchain technology, as a consequence of its current efficiency limitations [53].

In the article "Blockchain-based Traceability in Agri-Food Supply Chain Management: A Practical Implementation" another blockchain-based food supply chain is implemented and then used to compare two blockchain platforms: Hyperledger Sawtooth and Ethereum. The results give a performance advantage to the first, but the authors warn about the higher maturity level of the second [54].

Walmart and IBM announced in 2016 a collaboration with the objective to proof that the blockchain could be used to authenticate and trace food deliveries from supplier to consumer. With the adoption of the blockchain in the mangoes supply chain, they reduced the time to find the origin of a product from almost a week to a few seconds, by solving the problem that "stakeholders maintain records for one step up and one step down. This means that each stakeholder in the mango supply chain had to work with the next node in the chain to identify the provenance of my mangoes" [55]. The figure 2.2 shows how the introduction of the blochchain technology transforms the information flow in a supply chain from linear to circular. Walmart and IBM also used the blockchain technology to track and authenticate pork meat, reducing the number of paper documents and increasing the information detail level associated with each pork. Given the promising results of these first implementations, they extended the blockchain adoption to the production environment [55].

The project SmartLog, funded by the EU and with the purpose to address the lack of information sharing in the supply chain context, is another interesting application of the blockchain solution. The project stores information about cargo transportation in order to improve its efficiency [56], [57], but the official website does not provide any insight about its achievements.

In the article "Blockchain based Wine Supply Chain Traceability System", the authors describe a traceability system in the context of a wine supply chain. The system assigns to each wine bottle a unique identification number, which can be used by a customer to retrieve all the transactions involving it [58]. The authors' research, however, is more focused on the transparency and security of their solution than on its costs and performance evaluation. Other articles with a similar approach describe examples of the blockchain application to other types of supply chains, like manufacturing [59], drugs [60] or wood [61].

Given the lack of papers discussing the performance evaluation of the blockchain solution in the supply chain context, this work objective is to provide some results in such a direction. The first step is to identify a blockchain platform well suited for this goal.

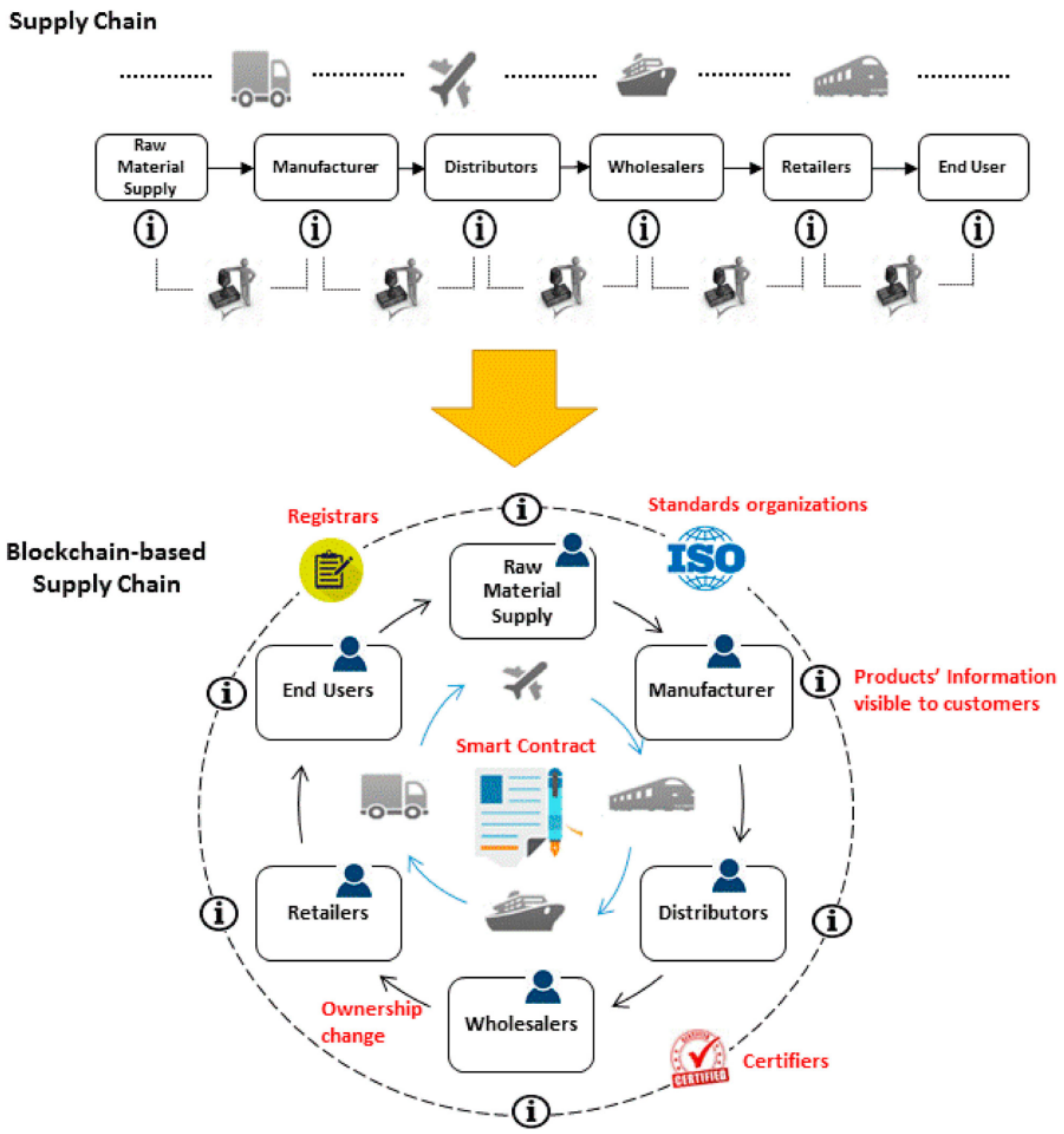

Figure 2.2. Supply chain transformation [50].

2.4 Hyperledger Sawtooth, the chosen platform

Hyperledger Sawtooth is the platform used for the supply chain implementation. It is a framework for the development of blockchains for business purposes. Its main strengths are its scalability and its modularity, which make of Hyperledger Sawtooth a good framework for a great variety of use cases. It offers a dynamically pluggable consensus algorithm, the possibility to implement a custom business logic through the use of smart contracts and many of the most common programming languages, like Go, Python, Javascript, Java and C++. Moreover, it requires from an application layer programmer a minimal knowledge of the underlying framework implementation details, freeing him from the problem to address the security concerns. In the framework, nodes are called "validators". Each block they publish contains a group of batches, with each batch being a group of transactions. The transactions in a batch are applied in order, with an all or nothing approach [62]. The framework architecture includes five components [63]:

the peer to peer network, that allows the nodes to exchange messages and the transactions to be propagated through a gossip algorithm. Nodes use 0MQ messages over a TCP channel. Messages are serialized using Google's Protocol Buffers. At this layer is also implemented the access control system, that allows to decide which nodes can join the network, which can participate at the consensus and which can submit transactions;

the distributed log, used to record transactions. It is the actual blockchain ledger, and transactions are strictly ordered. Even so, at transaction processing time, the framework offers a parallel scheduler;

the smart contract logic layer, used to process the application business logic. The framework introduces the concept of transaction family, which is a way to indicate a set of custom operations (transactions) the system recognizes. The smart contract handling such operations is called transaction processor. An interesting transaction family provided by the framework is the settings one. It allows to set some parameters regulating the blockchain itself, like the consensus algorithm to be used, in a dynamic way. Other transaction families and transaction processors can be defined at will;

the distributed state, which can be considered as a key-value store where the transactions processors can read and store data. In this way it is not necessary to go trough all the distributed log in order to retrieve the current value of some data. It thus represents the current state of the blockchain. The system is consistent when, after any transaction, all the nodes reach the same state (the data stored is the same). Wrong smart contract code may lead to state inconsistencies. For example, in the Go programming language, maps are iterated in random order, which may cause inconsistencies if the serializer used to store them in the distributed state does not enforce an external order;

the consensus algorithm, which is pluggable and can be dynamically changed. The following consensus algorithms are currently implemented: Devmode (used for testing of networks with at most one validator); PoET-CFT, PoET-BFT, PBFT, Raft (non-forking CFT algorithm).

The choice of this framework and the PoET consensus algorithm is based on the following considerations:

- the adoption of the blockchain technology is based on a common goal of the actors of the supply chain. For this reason, the block mining process should be driven by such a goal and not from a block mining reward. This allows the cost for submitting a transaction to be zero. Consequently, non currency-oriented platforms are preferable. Hyperledger Sawtooth is not currency oriented, but a currency system can be implemented over it;
- costumers should not handle the consensus algorithm, but they should be able to interact with the blockchain (e.g. by placing orders). Often times customers represent the majority of the participants in a supply chain and do not necessarily share the same goal of the rest of them. Moreover, competitors may join the network as customers, with the objective to take over the control of it. For these reasons, and because currency oriented blockchains are often times also public, the desired platform is a consortium blockchain, like Hyperledger Sawtooth;
- efficiency is a major concern. The blockchain solution should be able to handle the same number of transactions of a centralized system. This imposes a big limitation on the viable consensus algorithms to be used. The possibility to dynamically change consensus algorithm as a consequence of the evolving efficiency requirements is thus considered useful. Hyperladger Sawtooth has a pluggable consensus algorithm. The PoET-BFT has a good efficiency, and its CFT version can be used with a non-TEE processor for testing purposes;
- scalability is not a major concern. Because consensus nodes in the supply chain should represent organizations and not single users, the blockchain network dimension can be considered limited to hundreds of nodes in the worst case scenario. The PoET good scalability is thus an extra;
- deterministic finality has to be preferred over the probabilistic one, but the benefit is not crucial. For some financial applications waiting a few seconds for a transaction to be committed can make a huge economic difference. This is usually never the case in the supply chain environment, where a delay of even a few hours can be considered acceptable. The PoET guarantees only probabilistic finality;

- smart contracts usage and custom application logic definition should be possible, in order to cope with the different necessities of the supply chain actors. Moreover, the application logic layer should be separated from the blockchain one (to avoid the creation of security flows) and it should be possible to define permissioning rules at both levels. Hyperledger Sawtooth provides such functionalities.

Chapter 3

Implementation

This chapter describes the entities and the operations used to test the efficiency of the supply chain with blockchain support. The first section is dedicated to a simple system that implements the minimal functionalities of a supply chain. The second and third sections are dedicated to a more complex system that includes some of the most common entities and operations present in many supply chain systems. In particular, the second section is dedicated to the entities, while the third one to the operations. The last section shows a standard flow of operations in the complex system, to better clarify how entities and operations are related.

3.1 Simple supply chain

The simple supply chain implements the basic concept of property exchange in the blockchain context. Because of its simplicity, it can be used to estimate an upper bound to the blockchain performance in a given network configuration. It defines only three entities and four possible operations that can be performed:

actor entity: it is the entity that interacts with the supply chain by performing the related operations. It has only to generate a pair of valid keys to digitally sign the transactions it submits. Actors are not explicitly handled by the system, and their identity is deduced from the key used to sign the transactions;

asset entity: it models the concept of exchangeable property. It is composed of a string treated as an unique identifier, an integer representing the state of the asset and by a string indicating the public key of the actor who owns it;

proposal entity: it models the attempt from an actor to change the ownership of an asset. It keeps track of both the actor and the asset.

create operation: is used to create a new asset. The asset owner is considered to be the actor who submitted the create transaction. Such an actor must also provide a unique id to identify the asset, otherwise the transaction is not committed;

update operation: it is used to update the state of an asset. The system guarantees that only the owner of an asset can update its state;

exchange operation: it is used to start the exchange of an asset, by creating a proposal. Any actor can start an exchange, but it does not take place unless both the original owner and the new owner agree on it;

vote operation: it concludes a previously started exchange, by voting an existing proposal. After this operation, the property of an asset is assigned to the new owner.

The system described has many limitations, in particular it does not model the concepts of custody, of shared property and of operation delegation, where an actor could allow another one to perform some operations on his behalf. The system, however, allows the exchange of assets in a secure and transparent way, without the involvement of any third-party.

3.2 Full supply chain: entities

The full supply chain extends the simple one by introducing new entities and new operations, common in many supply chain use cases. Moreover, it introduces a more sophisticated access control system that allows to model the concepts of custody, operation delegation and shared property.

3.2.1 Actor

It is the entity that interacts with the supply chain by performing the related operations. It has only to generate a pair of valid keys to digitally sign the transactions it submits. Actors are not explicitly handled by the system, and their identity is deduced from the key used to sign the transactions. An actor can be a customer, an employee of a company, an IoT device or even a smart contract;

3.2.2 Policy

It is the entity used to perform access control. It is composed of:

ID: a string unique in the system;

permissions: a map between a string (the name of an operation) and the list of allowed public keys. For each entry, the map also stores a counter, which represents the quorum that must be reached in order to perform the operation.

The operations are of two types: the ones affecting another entity and the ones affecting the policy itself;

admin: a string representing an actor's public key. Such an actor is considered the default administrator of the policy. In case an operation on the policy itself is not defined among the permissions, only the default administrator is allowed to perform it (so it is treated as a fallback).

3.2.3 Proposal

In case an actor tries to modify an entity whose policy has a permission with a quorum bigger than one, a proposal must be created. A proposal stores information about the entities involved, the operations to be performed and the number of actors who already accepted them (voters). In particular a proposal is composed of:

ID: a string unique in the system. Currently it is assigned automatically by the system, by concatenating the id of the entities to be modified and the public key of the actor proposing the modifications. As a consequence, an actor cannot propose more than one modification on the same set of entities;

promoter: a string representing the public key of the actor proposing the modification;

modifications: a list containing information on the proposed operations to be performed. Each element of the list stores the data to retrieve the entity to be modified from the distributed state, the name of the operation to be performed on the entity, the list of actors who already agreed on the operation and the set of actors whose vote is required. Once the number of voters is bigger than the quorum and all the required voters are included in the list of voters, the operation can be performed. If all the operations in the list of modifications can be performed, the changes are applied.

3.2.4 Asset

It is the entity that represents objects exchanged in the supply chain. It has some properties common in many supply chains, but specific ones could be added to better represent information useful in some particular scenarios. The properties already defined are:

ID: a string unique in the system;

owner: a string representing the public key of the actor who owns the asset.

keeper: a string representing the public key of the actor who is currently in custody of the asset. In the current implementation the keeper must be explicitly allowed by the policy to update the asset, but a more flexible mechanism could be used in the future to reduce the dependencies between owner and keeper;

volume: an integer representing the volume of the asset, in volume units. This is a simplified version of what could be a real case scenario, where also the shape of the asset could be important for storing purposes;

temperature: an integer representing the temperature of the asset;

position: a pair of floats representing the latitude and the longitude where the asset is located;

policy: a string representing the name of the policy regulating the asset. In case a policy is not specified, the owner is the only actor who can modify the properties of the asset.

3.2.5 Delivery

It is the entity used to represent the movement of assets in the system. It is described by the following properties:

ID: a string unique in the system;

keeper: a string representing the public key of the actor who is currently handling the delivery;

receiver: a string representing the public key of the receiver of the delivery. In case the receiver's policy is set, this field is ignored;

status: the current delivery status. In the current implementation a delivery can be "in preparation", if the sender has not started the delivery process yet, "in transit", if the sender has started the delivery process, but the assets didn't reach the receiver yet or "received" if the delivery process is completed;

assets: the list of assets that belong to the delivery;

estimated time: the time at which the delivery is supposed to reach its destination;

real time: the time at which the delivery reached its destination;

delay cause: a string containing the reason the delivery arrived late, in case the real time is bigger than the estimated time;

policy: a string representing the name of the policy regulating the delivery. In case a policy is not specified, the keeper is the only actor who can modify the properties of the delivery.

receiver's policy: a string representing a policy. This policy should define which actors can accept the delivery. In case a policy is not specified, the receiver is the only actor who can accept the delivery.

3.2.6 Carrier

A carrier is the entity in charge of moving assets in the system. It is described by the following properties:

ID: a string unique in the system;

owner: a string representing the public key of the actor who owns the carrier.

fuel consumption rate: a float representing fuel consumption rate of the carrier. In reality, this parameter should vary based on various factors, like the carrier speed and the traffic condition, but for simplicity it is considered to be a constant;

emission rate: a float representing the CO_2 emission rate of the carrier. As for the fuel consumption rate, this value is considered a constant also if in reality its value is affected by various conditions;

fuel consumption: a float representing the quantity of fuel consumed by the carrier;

emission: a float representing the quantity of CO_2 produced by the carrier;

position: a pair of floats representing the position of the carrier;

current deliveries: a list representing the deliveries the carrier is currently performing;

completed deliveries: a list representing the deliveries the carrier has completed. While historic data can be always retrieved by analyzing the distributed log, it is more efficient to store it in the distributed state. This is an example of historic data saved in the distributed state;

policy: a string representing the name of the policy regulating the carrier. In case a policy is not specified, the owner is the only actor who can modify the properties of the carrier.

3.2.7 Warehouse

It is the entity used to represent a storing facility for the assets. It is described by the following properties:

ID: a string unique in the system;

owner: a string representing the public key of the actor who owns the warehouse.

free space: an integer representing the quantity of space still available in the warehouse for storing purposes. It is expressed in volume units;

free docks: an integer representing the number of carriers that can be accepted in the warehouse;

free trucks: an integer representing the number of available carriers ready to be loaded;

temperature: an integer representing the temperature of the warehouse;

policy: a string representing the name of the policy regulating the warehouse. In case a policy is not specified, the owner is the only actor who can modify the properties of the warehouse.

3.3 Full supply chain: operations

3.3.1 Actor

Because the actor entity is not directly handled by the system, no operation can be performed on it.

3.3.2 Policy

On a policy the following actions can be performed:

create: creates a new policy. The admin field is initialized with the public key of the actor who submitted the create transaction;

update admin: updates the admin field with the value provided as parameter;

append permission: append a new permission to the policy. After this transaction is committed, the policy allows a new operation to be performed on all the entities regulated by the policy;

remove permission: removes from the policy a permission;

create proposal: creates a proposal to perform some of the previous operations. It is the only option to perform such operations in case the number of voters needed to perform such operations is bigger than one.

3.3.3 Proposal

On a proposal the following operations can be performed:

vote: adds the current transactor to the voters list. The proposal is committed if all the required voters are included in the voters list and if the quorum is reached;

unvote: undo a previous vote operation. If after this operation the voters list is empty, the proposal is removed from the system.

3.3.4 Asset

On an asset the following actions can be performed:

create: creates a new asset;

delete: deletes an asset, based on its ID;

update owner: updates the owner of the asset. It is possible to specify a new policy too;

update keeper: updates the keeper of the asset;

update position: updates the position of the asset;

update volume: updates the volume of the asset;

update temperature: updates the temperature of the asset;

update policy: updates the policy of the asset. The current policy must allow to be replaced for this transaction to be successful;

create proposal: creates a proposal to perform some of the previous operations.

3.3.5 Delivery

On a delivery the following actions can be performed:

create: creates a new delivery;

delete: deletes a delivery;

update keeper: updates the information about the actor who is performing the delivery;

add asset: add an asset to the ones belonging to the delivery;

remove asset: removes an asset from the ones belonging to the delivery;

update estimated time: updates the estimated delivery time;

update delay cause: updates the information about what caused the delivery to be late;

update receiver: updates the receiver of the delivery;

update receiver's policy: updates the policy regulating who can accept the delivery;

update policy: updates the policy regulating the delivery itself;

create proposal: creates a proposal to perform some of the previous operations;

begin delivery: starts a delivery from a non-warehouse entity. It updates the delivery status and performs the operation "add delivery" of the carrier specified as parameter. This action can only be performed as a proposal;

finish delivery: ends a delivery to a non-warehouse entity. It updates the status and the real time properties of the delivery, and performs the operation "end delivery" of the carrier handling it. This action can be only performed as a proposal;

start delivery: as for begin delivery, but from a warehouse entity. Additionally, it performs the "send delivery" operation of the warehouse entity. This action can be only performed as a proposal;

end delivery: as for finish delivery, but to a warehouse entity. Additionally, it performs the "receive delivery" operation of the warehouse entity. This action can be only performed as a proposal;

stop delivery: as for end delivery, but it does not change the status or the real time properties of the delivery. A delivery can be stopped because it reached an intermediate warehouse where some operation needs to be done on the assets. This action can be only performed as a proposal;

continue delivery: continues a previously stopped delivery. It is similar to "start delivery", but it does not change the delivery status. This action can be only performed as a proposal.

3.3.6 Carrier

On a carrier the following actions can be performed:

create: creates a new carrier;

delete: deletes a carrier, based on its ID;

update owner: updates the owner of the carrier. It is possible to specify a new policy too;

update fuel consumption rate: updates the fuel consumption rate of the carrier;

update emission rate: updates the CO_2 emission rate of the carrier;

update fuel consumption: updates the total fuel consumption of the carrier;

update emission: updates the total CO_2 emission of the carrier;

update position: updates the position of the carrier;

update policy: updates the policy regulating the carrier;

create proposal: creates a proposal to perform some of the previous operations;

add delivery: adds a delivery to the active ones. This operation cannot be executed directly by submitting a transaction, but it is executed as a consequence of the operations called on a delivery entity;

end delivery: removes a delivery from the active ones and moves it in the completed ones. This operation cannot be executed directly by submitting a transaction, but it is executed as a consequence of the operations called on a delivery entity.

3.3.7 Warehouse

On a warehouse the following actions can be performed:

create: creates a new warehouse;

delete: deletes a warehouse;

update owner: updates the owner of the warehouse. It is possible to specify a new policy too;

update free space: updates the free space in the warehouse;

update free docks: updates the number of carriers that can enter in the warehouse;

update free trucks: updates the number of carriers ready to start a delivery;

update temperature: updates the temperature of the warehouse;

update policy: updates the policy of the warehouse with the one whose ID is provided as parameter;

create proposal: creates a proposal to perform some of the previous operations;

send delivery: updates the warehouse internal state as a consequence of a delivery leaving the warehouse. It increases its free space of an amount equals to the total amount of the volume of the assets belonging to the delivery, increases the number of free docks and reduces the number of free trucks. This operation cannot be executed directly by submitting a transaction, but it is executed as a consequence of the operations called on a delivery entity;

receive delivery: updates the warehouse internal state as a consequence of a delivery reaching the warehouse. It reduces its free space of an amount equals to the total amount of the volume of the assets belonging to the delivery, decreases the number of free docks and increases the number of free trucks. This operation cannot be executed directly by submitting a transaction, but it is executed as a consequence of the operations called on a delivery entity.

3.4 Full supply chain: example flow

In this section it is provided an example to better understand the relation between the entities and the operations of the full supply chain. For simplicity, the example shows the delivery of one asset from a producer to a consumer, with the support of one retailer owing two warehouses. The actors in the systems are the following:

- a temperature sensor (TS). For simplicity, TS is the only sensor used;
- a producer (P) who creates the asset (A) that will be exchanged in the system. P places TS on A;
- a retail company (RC), represented by its administrator. RC owns two warehouses (W1 and W2) and it does not want to handle directly all the exchanges of assets in which it is involved. RC buys A from P. For simplicity, RC trusts TS and it does not feel the need to place its own sensor on the received asset. RC also has a website (S) where the assets belonging to RC are advertised;
- a customer (C) who buys A from RC;
- four warehouse managers (WM1, WM2, WM3 and WM4) hired by RC. WM1, WM2 and WM3 are in charge of W1 while WM4 is in charge of W2. RC wants to enforce that any operation in W1 takes place only if at least two among WM1, WM2 and WM3 give their consent. A is delivered to W1 from P, then from W1 to W2, which is the nearest warehouse to C;
- a carrier company (CC) represented by its administrator. For simplicity, the company has only one carrier and the administrator is also the carrier driver.

Initialization In the initialization phase, all the entities interacting in the system are created. Their creation could also be performed at any other time, but creating them all together before starting the exchange of the asset makes the example clearer.

First of all, each actor needs to generate a pair of keys to interact with the system. Each actor does it autonomously and no transaction is submitted to the system.

P creates a policy and then uses the "append permission" policy operation to allow himself to perform any operation and to allow TS to perform the "update temperature" operation on an asset.

RC creates two policies (WP1 and WP2), one for each warehouse. RC uses the "append permission" policy operation of WP1 to allow WM1, WM2 and WM3 to perform any operation, and sets the quorum to 2 for each operation. RC uses the "append permission" policy operation of WP2 to allow WM4 to perform any operation, and sets the quorum to 1 for each operation. RC will then create two warehouses and assign to each of them the proper policy using the "update policy" warehouse operation. The policies created can also be assigned to other objects, like the assets belonging to RC. For this reason, the policies allow TS to perform the "update temperature" asset operation and CC to perform the "update position" asset operation.

CC will create a carrier end will not assign any policy to it. As a consequence only CC will be able to perform any operation on the carrier because he is its owner.

From P to W1 P creates A and uses the "update policy" asset operation to assign the policy he created to A. He also uses the others update operations of the asset to set its position, volume, temperature and any other relevant information. At this point, WM1 creates a proposal to update the owner of A using policy WP1 (WM1 tries to buy A). P and WM2 vote the proposal, committing it. P creates a delivery and adds A to it. P sets the receiver's policy of the delivery to be the policy used to buy A (WP1). P performs the "delivery begin" operation to start the delivery using carrier CC. CC votes the proposal, becoming the keeper of the delivery and of A. During the travel, TS periodically updates the temperature and CC the position of A. CC updates also the position, the fuel consumption and the CO_2 emission of his carrier according to the distance traveled. In proximity of W1, CC creates a "delivery end" proposal. WM1 and WM2 vote the proposal, committing it (WP1 allows them to vote in stead of RC). At this point the owner and keeper of A are both equals to RC.

From W1 to W2 C visits S, adds A to his cart and tires to purchase it. WM1 creates a proposal to update the owner of A with the public key of C. WM2 votes the proposal, as C does (by pressing a confirmation button). The proposal is committed. A different policy definition could allow directly S to create such proposal.

WM1 creates a delivery using WP1 as policy, adds A to the delivery and sets the receiver of the delivery to be C. WM1 executes the "delivery start" operation, which creates a proposal. CC votes the proposal and becomes the keeper of the delivery and of A. WM1 and WM2 update the policy of A to be WP2. During the travel, TS periodically updates the temperature and CC the position of A. CC updates also the position, the fuel consumption and the CO_2 emission of its carrier according to the distance traveled. In proximity of W2, CC creates a "delivery stop" proposal using policy WP2, because some operation needs to be performed on A in W2. WM4 votes the proposal, committing it (WP2 allows him to vote in stead of RC). At this point the owner of A is C and its keeper is RC.

From W2 to C The next day, the delivery can be resumed. For this reason, WM4 executes the "delivery continue" operation, which creates a proposal. CC votes the proposal and becomes the keeper of A once again. TS periodically updates the temperature and CC updates the position of A. CC updates also the position, the fuel consumption and the CO_2 emission of its carrier according to the distance traveled. Arrived at C's house, CC executes a "delivery finish" operation which creates a proposal. C votes the proposal once he has the asset in his hands. C is now the owner and the keeper of A.

Chapter 4

Testing

This chapter describes the tests performed on the two implemented chains. According to the "Hyperledger Blockchain Performance Metrics" paper [64], first the general system description is given, including hardware, software and network configuration used to perform the tests. Then the tests performed on the simple chain are described, and finally the ones performed on the full chain using three different use cases. The use cases differ in the number of producers and customers in the system, but overall the flow of transactions is similar to the one described in section 3.4. The results of the tests performed are reported in Appendix A.

4.1 Test Harness and methodology

The tests are performed using a single computer and the Docker platform to virtualize the blockchain network. The transaction processors are implemented in Go, while the clients in Typesript. The configuration of the properties of the blockchain and of the validators are based on the one proposed by Mattew Rubino [65].

4.1.1 Hardware configuration

MODEL: ASUS N56JK-CN051H;

CPU: Intel Core i7-4710HQ, 2.50GHz, octacore;

RAM: 7,7 GiB DDR3, 1600 MT/s;

DISK: 343,0 GB, 5400 RPM.

4.1.2 Software configuration

Ubuntu: 18.04.3 LTS;

Docker: version 19.03.3, build a872fc2f86;

Sawtooth: 1.0.5;

Go: version go1.11.2 linux/amd64;

Node: v11.0.0;

Angular: 8.3.1;

Google Chrome: 70.0.3538.77, launched with the "–disable-web-security" command line option to avoid problems related to the CORS policy.

4.1.3 Network configuration

Consensus protocol: PoET-CFT;

Geographic distribution: co-located nodes;

Network model: 5-node complete graph (fig. 4.1);

Number of nodes involved in the test transaction: 1 to 5, specified in each test;

Software component dependencies: none, other than the default ones.

4.1.4 Blockchain's properties configuration

sawtooth.poet.target_wait_time: 5;

sawtooth.poet.initial_wait_time: 25;

sawtooth.publisher.max_batches_per_block: 1000;

sawtooth.validator.max_transactions_per_block: 1000;

sawtooth.poet.ztest_minimum_win_count: 999999999.

4.1.5 Validator's properties configuration

peering: dynamic;

scheduler: parallel;

network: trust;

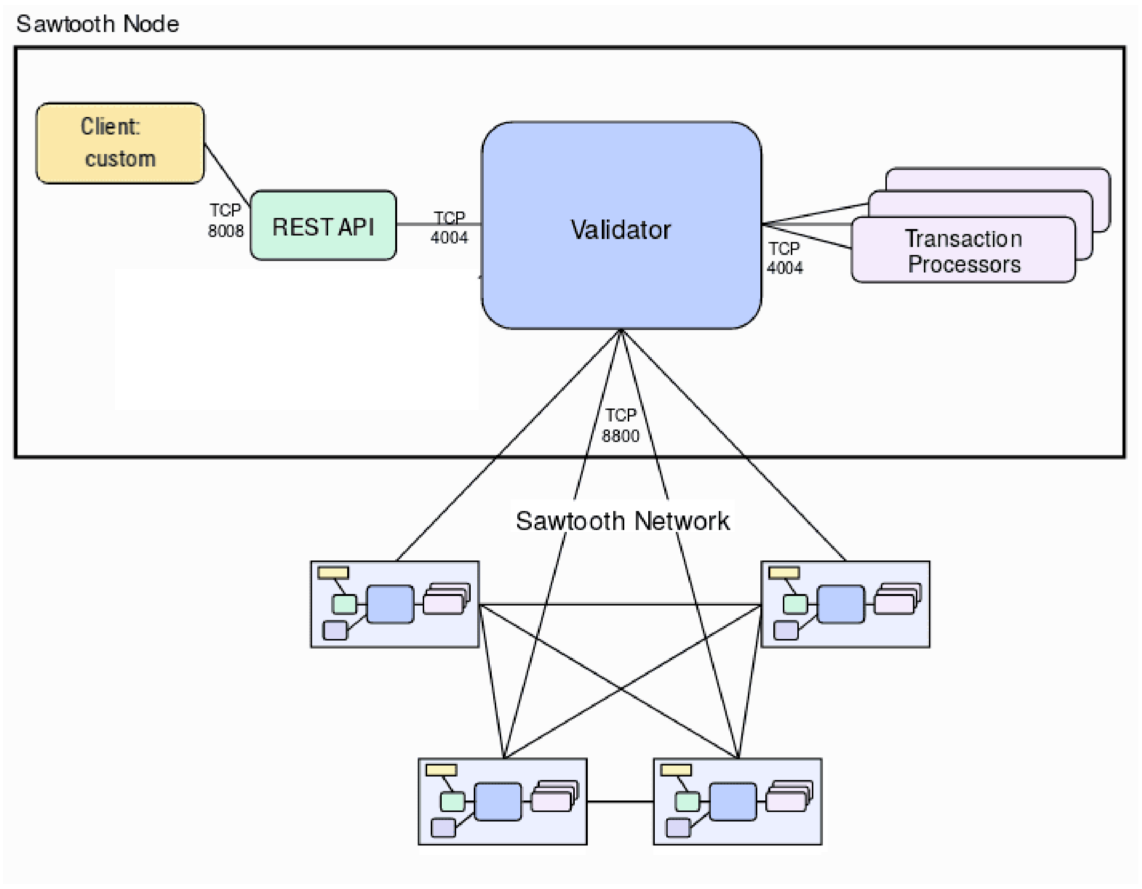

Figure 4.1. Network of validators [66].

4.1.6 Methodology

Test tools and framework. The tests are performed in a local environment, thus the client is hosted on the same machine of the network of validators. Network load is generated and captured using the Angular framework and the Google Chrome web browser. The node used by the client to submit the transactions changes in each test. This information is thus provided with the description of each of the tests performed.

Workload. For the simple supply chain the workload is a combination of the only four possible operations. For the full supply chain a sample workload is described in section 3.4.

Finality threshold. The finality threshold is 100% of the nodes: all the validators must consider a transaction committed before it is considered as such by the client.

Measure type. The focus of this work is on the transaction throughput measure (TPS), defined as: *total committed transactions / total time in seconds* [64].

Observation points. The blockchain performance is measured from the perspective of a client. The total time to calculate the TPS measure is thus defined as: *time the client reads the transaction as committed on all the nodes* − *time the client submits the transaction to one node.*

Testing strategy. The tests are performed using a polling strategy. This allows the client to know the status of a batch with a little delay in the worst case, while the events broadcasted by the framework may be grouped together to reduce the number of messages sent [67]. Moreover, the reads performed to poll the system help the simulation, because in a real-case scenario both transactions and reads should be submitted by the clients. During the tests.The network traffic quantity is kept almost constant, by allowing only up no N batches to be pending. Various values of N are used in the tests. All the batches used in each test are prepared in advance, so that they can be submitted immediately without spending computational time on the client. Due to the memory constraints of the computer used, this may limit the total number of batches used.

Transactions characteristics. The transactions used for testing purposes can all be considered small and simple. Some of them are more complex than others, to the point that some transactions represent groups of simpler ones often submitted together. However, even transactions that are linear in the number of entities defined in the system can be considered simple as a consequence of the limited amount of such entities. The dependencies and data access patterns of the transactions follow the ones of a simple production use.

4.2 Simple supply chain

The tests performed on the simple supply chain are various and involve a limited number of repetitions (ten each). The goal is to find out differences in the TPS values as a consequence of different batches sizes and submission methods.

4.2.1 Single batch tests

In this type of tests only one batch is submitted in the system (N=1), and thus all the transactions are submitted to the same validator. The objective of these tests is to find the number of transactions in a batch that maximizes the TPS value. The results are reported in figure 4.2, and the maximum TPS value is reached with batches containing among 35 and 350 transactions.

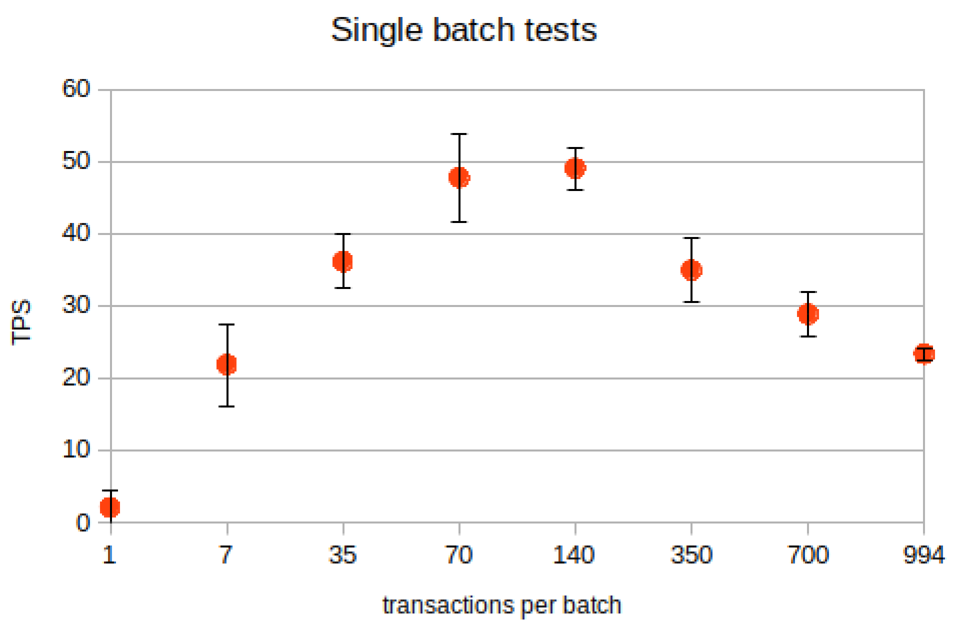

Figure 4.2. TPS for the single batch tests.

4.2.2 Batch dependency tests

In this type of tests the client submits each batch to a different validator, in a round robin fashion. A total of twenty batches are submitted, each containing a different number of transactions in each test. The client allows up to three batches to be not yet committed (N=3). Each batch is dependent from the previous one, so a total order on the batches is enforced. The objective of these tests is to clarify the impact the dependencies among transactions and batches have on the TPS value. The results are shown in figure 4.3. It is important to remark that while the numbers of transactions per batch used in these tests are a subset of the one used in the previous one, the total number of transaction and the way they are submitted differs. This must be taken into account when comparing the results of the two type of tests.

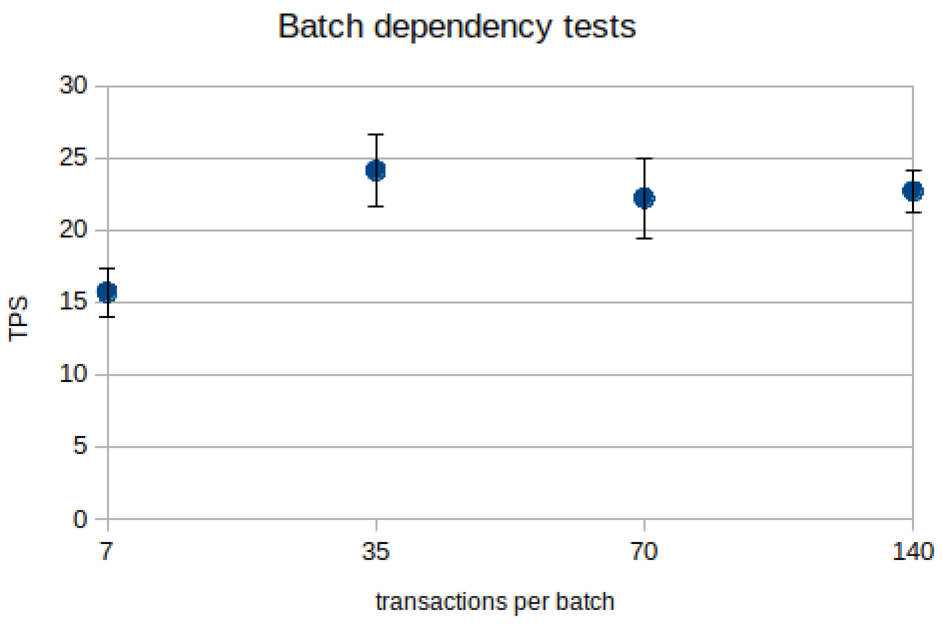

Figure 4.3. TPS for the batch dependency tests.

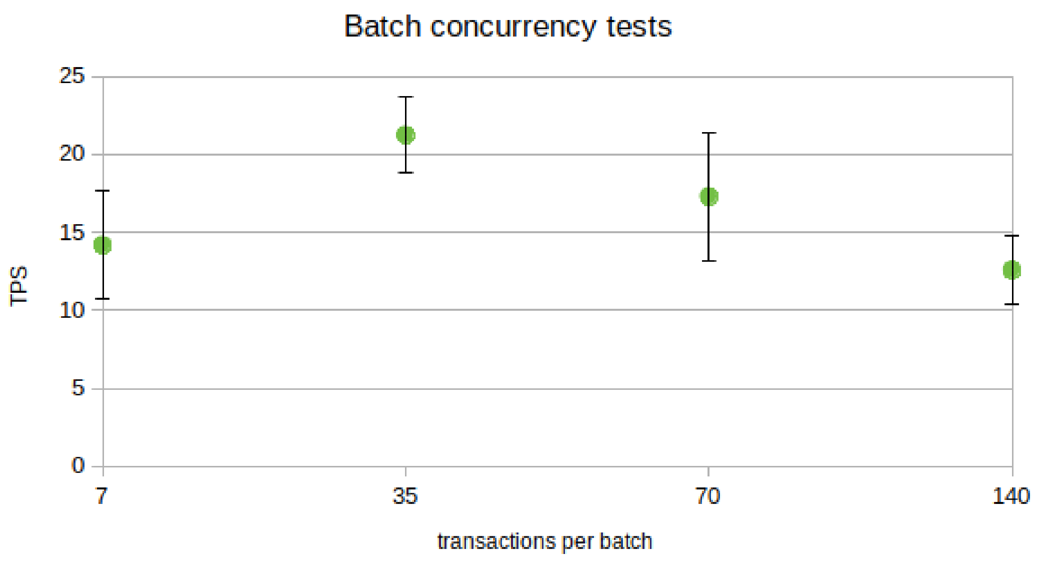

Figure 4.4. TPS for the batch concurrency tests.

4.2.3 Batch concurrency tests

In this type of tests the client submits each batch to a different validator, in a round robin fashion. A total of twenty batches are submitted, each containing a different number of transactions in each test. The client allows up to ten batches to be not yet committed (N=10). All batches are independent from one another. The objective of these tests is to clarify if the simultaneous presence of many transactions in the system improves the TPS value found in the previous section as a consequence of the fact that each block will contain more transactions (and thus a minor number of blocks is required to be mined), or if this approach only favors the arise of forks. The results are shown in figure 4.4, and they highlight a general performance decay with respect to the results provided in section 4.2.2.

4.3 Full supply chain

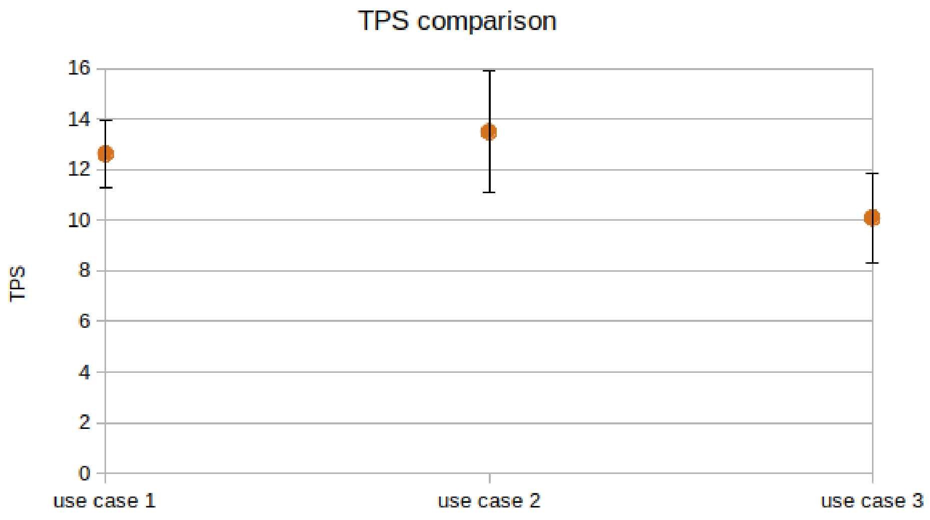

Figure 4.5. Transaction per second (TPS) for the three use cases.

The tests performed on the full supply chain take into account the results obtained in the previous section in order to optimize the TPS value. A total of twenty batches are simultaneously submitted in each test, and each batch contains thirty-five transactions. Differently from the previous section, the client also counts the total number of reads performed. Each test includes a total of fifty repetitions. The results of the tests are reported in figures 4.5 and 4.6. Figure 4.5 shows the TPS value reached in each use case, while figure 4.6 shows the number of reads performed in a second (RPS) for each use case. This value is used to give an idea

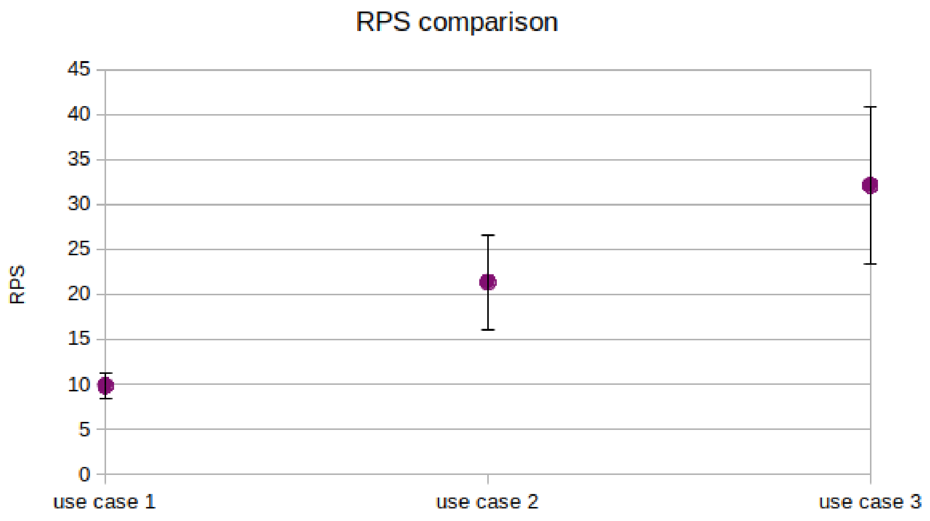

Figure 4.6. Read per second (RPS) for the three use cases.

of the workload the system was subject to during the tests, and it does not represent the maximum number of reads the system is able to support. The graphs show a good consistency among the results of the three use cases and they confirm some observations already done in the previous section: the presence of too many transactions in the system and their submission to different validators negatively affects the TPS value. Although this tendency is in line with the behaviour of other networks, like the IP one, it can be probably mitigated with a variation of the "sawtooth.poet.target_wait_time" setting value.

4.3.1 First use case

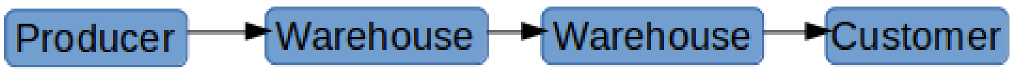

Figure 4.7. Supply chain configuration for the first use case.

This use case presents a single producer and a single customer. A graphical representation is reported in figure 4.7. This use case is characterized by an high number of dependent transactions, because the assets are moved between the same points, which enforces a total order among the batches. At the same time, all the batches are submitted to the same validator.

4.3.2 Second use case

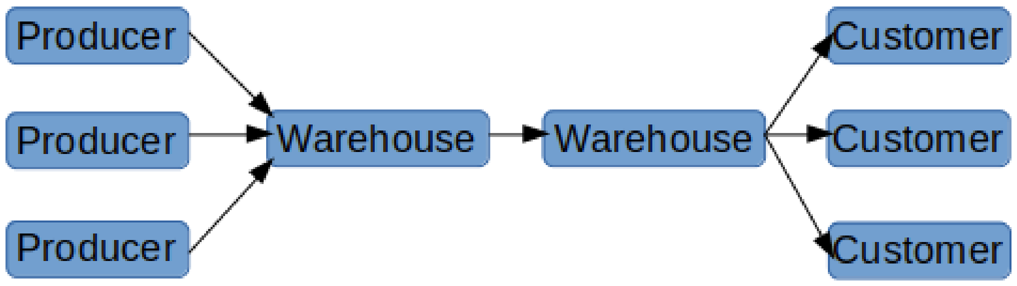

Figure 4.8. Supply chain configuration for the second use case.

This use case presents three producers and three customers. A graphical representation is reported in figure 4.8. This use case is characterized by a greater number of parallel deliveries with respect to the previous one, because the assets can be produced by different sources and can reach different destinations. The main transaction flow, however, is similar to the one described in section 3.4, with the difference that there are three concurrent deliveries from the various producers to the warehouses and three more from the warehouses to the customers. The batches are submitted to three different validators (batches related to the same delivery flow are submitted to the same validator).

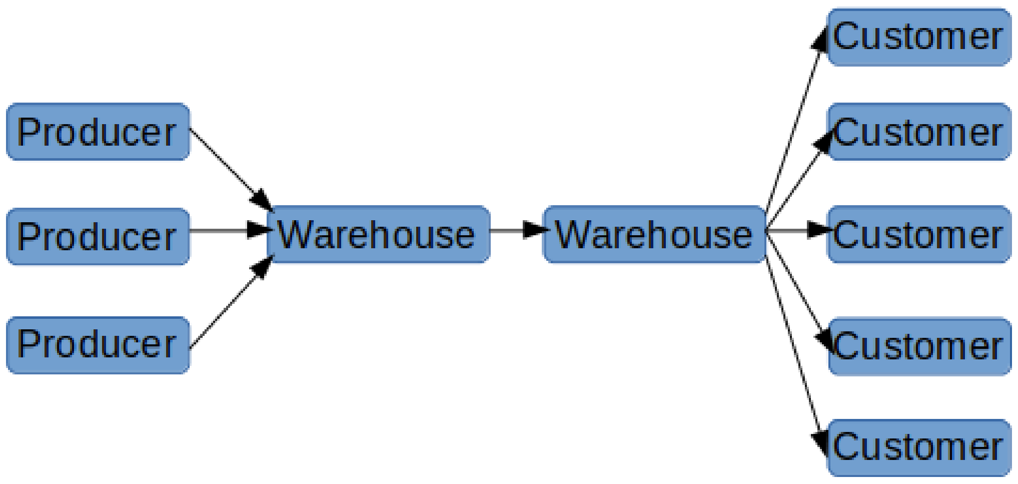

Figure 4.9. Supply chain configuration for the third use case.

4.3.3 Third use case

This use case presents three producers and five customers. A graphical representation is reported in figure 4.9. As a consequence of the bigger number of customers with respect to the previous use case, the number of concurrent deliveries is also increased. In fact, there are three deliveries from the various producers to the warehouses, and five from the warehouses to the customers. The batches are submitted to all the validators (batches related to the same delivery flow are submitted to the same validator).

Chapter 5

Conclusion

The blockchain technology has the potential to revolutionize the supply chain environment by offering a standard way to share data among different entities in an immutable and transparent way. Its application could reduce the paperwork processing costs and time, reduce the amount of fake products introduced in the market and offer a way to identify them, allow to track the origin of a product from the producer to the consumer, and thus to limit the issues related to a production defect. Finally, if coupled with the IoT devices, it could even create new businesses models and revolutionize the old ones. One of the main issues affecting the technology is its low efficiency, if compared to the one of many distributed databases. In literature there is a general lack of information regarding the quantity of data that the blockchian technology is able to process because it is hard to define a reliable metric that takes into account all the variables that can affect the blockchain performance. One of the metrics mostly adopted is the TPS value, used in this work to evaluate the blockchain performance. In particular, a network of five nodes hosted on the same computer was built using the Hyperledger Sawtooth framework and the the Docker technology. The choice of the Sawtooth framework was mainly due to its modularity and its scalability. Hyperledger Fabric could have been evenly if not better suited, but it has been discarded because it was not fully decentralized [68] when the choice was made, nor completely BFT [69]. Quorum (Ethereum based) was another viable solution, but it offered only two consensus algorithms: Raft (CFT) and IBFT which is based on PBFT and inherits its scalability limitations [70]. Finally, Corda was mostly renowned in the financial sector [71], while Sawtooth already furnished some supply chain implementation examples.

Two types of supply chain have been implemented: a simpler supply chain characterized by the definition of really simple transactions, whose complexity is comparable with the one of a CRUD operation, and a more complex one, that defines entities and operations common to many supply chains and gives the possibility to manage shared assets and to delegate operations. Various tests have been performed that highlight a decay of the performance of the blockchain system if too many transactions are submitted simultaneously or if they are submitted to different nodes. This is probably a consequence of the creation of forks related to the consensus algorithm used in the blockchain, the PoET-CFT, and is thus probably linked to the "sawtooth.poet.target_wait_time" setting value. Moreover, the reduced hardware resources used during the tests limits the exploitation of the results in a real case scenario. For these reasons, possible extensions of the work conducted so far could involve the usage of a wider blockchain network and of more advanced hardware resources or the usage of different consensus algorithms and blockchain settings. Of the two implemented supply chain, the more complex one has been tested in three different use cases, but many more could be produced to find out which are the factors that mainly affect the blockchain performance or as a reference of supported workload for real case usages. Moreover, some investigations could be performed to determine if there is a strong correlation between the complex and the simple supply chain TPS values, independently from the network configuration, which could lead to the definition of a transaction unit of standard complexity. Finally, the system itself could be improved. In particular the following observations are considered interesting:

- the system currently does not distinguish the concepts of delivery, intended as a point to point (e.g. warehouse to warehouse) movement of assets and of order, intended as an actor to actor movement of assets. This distinction could produce a more realistic implementation of the system;

- the policy mechanism is quite powerful, but at the same time not much flexible. The introduction of a role-based approach could reduce this restriction. In particular, the implementation of the concept of custody could benefit from a more flexible solution;

- the proposal mechanism is currently limited to sets of predefined modifications. The introduction of a way to add a custom modification among the ones defined in the system could allow users to define custom sets of operations to be performed. The drawback of this approach is that voting such proposals must be done carefully because the voter must be fully aware of how each modification may affect the system;

- each modification in the proposal mechanism can be a CRUD, an append or a remove operation. The mechanism could be enhanced to invoke any function defined in a registry in the system;

- the system could define the Actor entity and associate a policy to it. This could allow the removal of the owner, keeper and receiver fields from all the entities and replace them with the respective policies, in order to simplify the transaction logic.

Appendix A

Tests results

This appendix contains the results of the tests. For the simple supply chain, from each set of tests the maximum and minimum values are removed before performing any computation on the data.

Simple supply chain: single batch tests, 1 transaction			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
1	1	1184	284 ± 50
1	1	1184	339 ± 50
1	1	1184	262 ± 50
1	1	1184	287 ± 50
1	1	1184	7281 ± 50
1	1	1184	318 ± 50
1	1	1184	298 ± 50
1	1	1184	1815 ± 50
1	1	1184	270 ± 50
1	1	1184	278 ± 50

Simple supply chain: single batch tests, 7 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
1	7	11844	281 ± 50
1	7	11844	278 ± 50
1	7	11844	299 ± 50
1	7	11844	387 ± 50
1	7	11844	275 ± 50
1	7	11844	271 ± 50
1	7	11844	8748 ± 50
1	7	11844	267 ± 50
1	7	11844	502 ± 50
1	7	11844	247 ± 50

Simple supply chain: single batch tests, 35 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
1	35	58401	825 ± 50
1	35	58401	691 ± 50
1	35	58401	1132 ± 50
1	35	58401	3251 ± 50
1	35	58401	1018 ± 50
1	35	58401	874 ± 50
1	35	58401	1061 ± 50
1	35	58401	908 ± 50
1	35	58401	948 ± 50
1	35	58401	992 ± 50

Simple supply chain: single batch tests, 70 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
1	70	116596	1577 ± 100
1	70	116596	3122 ± 100
1	70	116596	1677 ± 100
1	70	116596	1258 ± 100
1	70	116596	1253 ± 100
1	70	116596	1452 ± 100
1	70	116596	1729 ± 100
1	70	116596	1295 ± 100
1	70	116596	1472 ± 100
1	70	116596	1265 ± 100

Simple supply chain: single batch tests, 140 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
1	140	232987	2943 ± 100
1	140	232987	2826 ± 100
1	140	232987	3808 ± 100
1	140	232987	2852 ± 100
1	140	232987	3221 ± 100
1	140	232987	2786 ± 100
1	140	232987	2658 ± 100
1	140	232987	2793 ± 100
1	140	232987	2646 ± 100
1	140	232987	2761 ± 100

Simple supply chain: single batch tests, 350 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
1	350	582157	10264 ± 250
1	350	582157	12374 ± 250
1	350	582157	10778 ± 250
1	350	582157	9062 ± 250
1	350	582157	15910 ± 250
1	350	582157	8767 ± 250
1	350	582157	8357 ± 250
1	350	582157	10307 ± 250
1	350	582157	10262 ± 250
1	350	582157	7848 ± 250

Simple supply chain: single batch tests, 700 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
1	700	1164107	27056 ± 250
1	700	1164107	25508 ± 250
1	700	1164107	28374 ± 250
1	700	1164107	27249 ± 250
1	700	1164107	20162 ± 250
1	700	1164107	25291 ± 250
1	700	1164107	22839 ± 250
1	700	1164107	19916 ± 250
1	700	1164107	24168 ± 250
1	700	1164107	21872 ± 250

Simple supply chain: single batch tests, 994 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
1	994	1652945	38397 ± 250
1	994	1652945	39832 ± 250
1	994	1652945	42287 ± 250
1	994	1652945	42553 ± 250
1	994	1652945	42622 ± 250
1	994	1652945	43755 ± 250
1	994	1652945	42396 ± 250
1	994	1652945	45318 ± 250
1	994	1652945	42456 ± 250
1	994	1652945	44686 ± 250

Simple supply chain: batch dependency tests, 7 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
20	7	12761	9373 ± 250
20	7	12761	8279 ± 250
20	7	12761	8768 ± 250
20	7	12761	9944 ± 250
20	7	12761	9609 ± 250
20	7	12761	7799 ± 250
20	7	12761	10054 ± 250
20	7	12761	7597 ± 250
20	7	12761	7127 ± 250
20	7	12761	11311 ± 250

Simple supply chain: batch dependency tests, 35 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
20	35	62986	26950 ± 500
20	35	62986	23606 ± 500
20	35	62986	25628 ± 500
20	35	62986	30135 ± 500
20	35	62986	31304 ± 500
20	35	62986	29086 ± 500
20	35	62986	24807 ± 500
20	35	62986	33403 ± 500
20	35	62986	30856 ± 500
20	35	62986	34988 ± 500

Simple supply chain: batch dependency tests, 70 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
20	70	125766	61599 ± 1500
20	70	125766	61664 ± 1500
20	70	125766	53255 ± 1500
20	70	125766	74967 ± 1500
20	70	125766	58010 ± 1500
20	70	125766	45959 ± 1500
20	70	125766	69280 ± 1500
20	70	125766	77287 ± 1500
20	70	125766	71152 ± 1500
20	70	125766	54310 ± 1500

Simple supply chain: batch dependency tests, 140 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
20	140	251327	106031 ± 2500
20	140	251327	133364 ± 2500
20	140	251327	155395 ± 2500
20	140	251327	123921 ± 2500
20	140	251327	115735 ± 2500
20	140	251327	130033 ± 2500
20	140	251327	113196 ± 2500
20	140	251327	131385 ± 2500
20	140	251327	122542 ± 2500
20	140	251327	117376 ± 2500

Simple supply chain: batch concurrency tests, 7 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
20	7	11844	8949 ± 250
20	7	11844	6336 ± 250
20	7	11844	9067 ± 250
20	7	11844	12991 ± 250
20	7	11844	15467 ± 250
20	7	11844	7077 ± 250
20	7	11844	8277 ± 250
20	7	11844	10758 ± 250
20	7	11844	13861 ± 250
20	7	11844	8071 ± 250

Simple supply chain: batch concurrency tests, 35 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
20	35	58401	30099 ± 500
20	35	58401	34425 ± 500
20	35	58401	41535 ± 500
20	35	58401	30206 ± 500
20	35	58401	33105 ± 500
20	35	58401	30509 ± 500
20	35	58401	29543 ± 500
20	35	58401	32865 ± 500
20	35	58401	31026 ± 500
20	35	58401	46687 ± 500

Simple supply chain: batch concurrency tests, 70 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
20	70	116596	56985 ± 1500
20	70	116596	54803 ± 1500
20	70	116596	78956 ± 1500
20	70	116596	84849 ± 1500
20	70	116596	86417 ± 1500
20	70	116596	64722 ± 1500
20	70	116596	92318 ± 1500
20	70	116596	66564 ± 1500
20	70	116596	121532 ± 1500
20	70	116596	117321 ± 1500

Simple supply chain: batch concurrency tests, 140 transactions			
Batches	Transactions per batch	Batch length (bytes)	Execution time (ms)
20	140	232987	152076 ± 2500
20	140	232987	249963 ± 2500
20	140	232987	225297 ± 2500
20	140	232987	313814 ± 2500
20	140	232987	221052 ± 2500
20	140	232987	187504 ± 2500
20	140	232987	163286 ± 2500
20	140	232987	216584 ± 2500
20	140	232987	227064 ± 2500
20	140	232987	290825 ± 2500

Full supply chain: First use case			
Batches (transactions per batch)	Total batches length (kilobytes)	Number of reads	Execution time (ms)
20 (35)	1157	635	66753 ± 250
20 (35)	1157	535	54174 ± 250
20 (35)	1157	455	45365 ± 250
20 (35)	1157	435	42688 ± 250
20 (35)	1157	495	49553 ± 250
20 (35)	1157	515	51698 ± 250
20 (35)	1157	575	58748 ± 250
20 (35)	1157	530	53347 ± 250
20 (35)	1157	560	56938 ± 250
20 (35)	1157	565	57842 ± 250
20 (35)	1157	470	47649 ± 250
20 (35)	1157	655	68340 ± 250
20 (35)	1157	550	55731 ± 250
20 (35)	1157	675	71210 ± 250
20 (35)	1157	540	55816 ± 250
20 (35)	1157	545	56543 ± 250
20 (35)	1157	535	55355 ± 250
20 (35)	1157	560	58451 ± 250
20 (35)	1157	500	51313 ± 250
20 (35)	1157	485	50851 ± 250
20 (35)	1157	560	57225 ± 250
20 (35)	1157	475	48053 ± 250
20 (35)	1157	585	61461 ± 250
20 (35)	1157	490	49297 ± 250
20 (35)	1157	505	51086 ± 250
20 (35)	1157	525	53412 ± 250
20 (35)	1157	555	56424 ± 250
20 (35)	1157	530	53682 ± 250
20 (35)	1157	615	63695 ± 250
20 (35)	1157	520	52353 ± 250
20 (35)	1157	570	58069 ± 250
20 (35)	1157	580	59533 ± 250
20 (35)	1157	525	53176 ± 250
20 (35)	1157	510	51962 ± 250
20 (35)	1157	595	61320 ± 250
20 (35)	1157	595	60992 ± 250
20 (35)	1157	580	59273 ± 250

20 (35)	1157	540	55053 ± 250
20 (35)	1157	625	64704 ± 250
20 (35)	1157	565	57447 ± 250
20 (35)	1157	580	59279 ± 250
20 (35)	1157	585	59870 ± 250
20 (35)	1157	510	49948 ± 250
20 (35)	1157	520	51204 ± 250
20 (35)	1157	560	56439 ± 250
20 (35)	1157	450	43786 ± 250
20 (35)	1157	525	51959 ± 250
20 (35)	1157	580	55603 ± 250
20 (35)	1157	545	54848 ± 250
20 (35)	1157	550	55109 ± 250

Full supply chain: second use case			
Batches (transactions per batch)	Total batches length (kilobytes)	Number of reads	Execution time (ms)
20 (35)	1157	1175	54462 ± 250
20 (35)	1157	790	38632 ± 250
20 (35)	1157	1285	58977 ± 250
20 (35)	1157	1275	57620 ± 250
20 (35)	1157	1450	64969 ± 250
20 (35)	1157	1105	53329 ± 250
20 (35)	1157	1060	45077 ± 250
20 (35)	1157	840	38566 ± 250
20 (35)	1157	1125	51063 ± 250
20 (35)	1157	1200	67251 ± 250
20 (35)	1157	1450	67958 ± 250
20 (35)	1157	940	42766 ± 250
20 (35)	1157	1300	55694 ± 250
20 (35)	1157	870	42084 ± 250
20 (35)	1157	1095	51413 ± 250
20 (35)	1157	1275	64047 ± 250
20 (35)	1157	1275	57776 ± 250
20 (35)	1157	1045	44221 ± 250
20 (35)	1157	1310	73980 ± 250
20 (35)	1157	970	44356 ± 250
20 (35)	1157	975	47146 ± 250
20 (35)	1157	900	42051 ± 250
20 (35)	1157	1115	49759 ± 250
20 (35)	1157	1230	55593 ± 250

20 (35)	1157	1095	48824 ± 250
20 (35)	1157	915	43324 ± 250
20 (35)	1157	1060	47783 ± 250
20 (35)	1157	1325	67694 ± 250
20 (35)	1157	1430	71639 ± 250
20 (35)	1157	1220	53457 ± 250
20 (35)	1157	930	42107 ± 250
20 (35)	1157	1245	62922 ± 250
20 (35)	1157	840	44590 ± 250
20 (35)	1157	1140	49357 ± 250
20 (35)	1157	960	44085 ± 250
20 (35)	1157	1430	69148 ± 250
20 (35)	1157	1220	56132 ± 250
20 (35)	1157	1150	50667 ± 250
20 (35)	1157	1025	46560 ± 250
20 (35)	1157	935	40457 ± 250
20 (35)	1157	965	45991 ± 250
20 (35)	1157	825	39927 ± 250
20 (35)	1157	1260	58422 ± 250
20 (35)	1157	810	44231 ± 250
20 (35)	1157	925	42558 ± 250
20 (35)	1157	1015	47451 ± 250
20 (35)	1157	1070	52785 ± 250
20 (35)	1157	1310	57299 ± 250
20 (35)	1157	1045	49358 ± 250
20 (35)	1157	1005	48339 ± 250

Full supply chain: third use case			
Batches (transactions per batch)	Total batches length (kilobytes)	Number of reads	Execution time (ms)
20 (35)	1157	1640	54238 ± 250
20 (35)	1157	1735	60592 ± 250
20 (35)	1157	2070	61689 ± 250
20 (35)	1157	1930	57299 ± 250
20 (35)	1157	2695	77529 ± 250
20 (35)	1157	1630	53575 ± 250
20 (35)	1157	2180	64768 ± 250
20 (35)	1157	1745	58826 ± 250
20 (35)	1157	2260	70708 ± 250
20 (35)	1157	1370	48563 ± 250
20 (35)	1157	3195	91598 ± 250

20 (35)	1157	2050	69319 ± 250
20 (35)	1157	3325	98733 ± 250
20 (35)	1157	2495	83813 ± 250
20 (35)	1157	2090	69074 ± 250
20 (35)	1157	1940	58541 ± 250
20 (35)	1157	1855	56813 ± 250
20 (35)	1157	3100	88520 ± 250
20 (35)	1157	2005	64469 ± 250
20 (35)	1157	1955	72516 ± 250
20 (35)	1157	2210	69597 ± 250
20 (35)	1157	2145	64615 ± 250
20 (35)	1157	2020	64817 ± 250
20 (35)	1157	2380	75818 ± 250
20 (35)	1157	2415	70255 ± 250
20 (35)	1157	2090	70936 ± 250
20 (35)	1157	3005	102231 ± 250
20 (35)	1157	2120	59438 ± 250
20 (35)	1157	2675	83054 ± 250
20 (35)	1157	1985	60832 ± 250
20 (35)	1157	2525	75038 ± 250
20 (35)	1157	2470	83747 ± 250
20 (35)	1157	2725	77515 ± 250
20 (35)	1157	2485	76282 ± 250
20 (35)	1157	1985	64969 ± 250
20 (35)	1157	2845	83098 ± 250
20 (35)	1157	1730	53565 ± 250
20 (35)	1157	2300	65870 ± 250
20 (35)	1157	2195	72090 ± 250
20 (35)	1157	3090	89086 ± 250
20 (35)	1157	2750	78457 ± 250
20 (35)	1157	2060	65289 ± 250
20 (35)	1157	1790	56316 ± 250
20 (35)	1157	1965	60864 ± 250
20 (35)	1157	2510	71282 ± 250
20 (35)	1157	1740	52927 ± 250
20 (35)	1157	2080	63131 ± 250
20 (35)	1157	1625	58714 ± 250
20 (35)	1157	1725	56723 ± 250
20 (35)	1157	2405	80419 ± 250

Bibliography

[1] E. Heilman, F. Baldimtsi, and S. Goldberg, «Blindly Signed Contracts: Anonymous On-Blockchain and Off-Blockchain Bitcoin Transactions», vol. 9604, Feb. 2016, pp. 43–60, ISBN: 978-3-662-53356-7. DOI: 10.1007/978-3-662-53357-4_4.

[2] S. Nakamoto, «Bitcoin: A Peer-to-Peer Electronic Cash System», *Cryptography Mailing list at https://metzdowd.com*, Mar. 2009.

[3] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, «Blockchain challenges and opportunities: A survey», *International Journal of Web and Grid Services*, vol. 14, p. 352, Oct. 2018. DOI: 10.1504/IJWGS.2018.095647.

[4] N. Hackius and M. Petersen, «Blockchain in Logistics and Supply Chain: Trick or Treat?», Oct. 2017. DOI: 10.15480/882.1444.

[5] L. Hellinga, «The Gutenberg Revolutions», in. Sep. 2019, pp. 377–392, ISBN: 9781119018179. DOI: 10.1002/9781119018193.ch25.

[6] J. Serrano, «Changes in the management of information in audio-visual archives following digitization: Current and future outlook», *Journal of Librarianship and Information Science - J LIBR INF SCI*, vol. 40, Mar. 2008. DOI: 10.1177/0961000607086617.

[7] E. Codd, «A Relational Model of Data for Large Shared Data Banks», *Commun. ACM*, vol. 13, pp. 377–387, Jan. 1970. DOI: 10.1007/978-3-642-48354-7_4.

[8] S. Y. Shim, «Guest Editor's Introduction: The CAP Theorem's Growing Impact», *Computer*, vol. 45, no. 02, pp. 21–22, Feb. 2012, ISSN: 1558-0814. DOI: 10.1109/MC.2012.54.

[9] D. Colley, M. Asaduzzaman, and C. Stanier, «Investigating the Effects of Object-Relational Impedance Mismatch on the Efficiency of Object-Relational Mapping Frameworks», Nov. 2018.

[10] C. Hale. (Oct. 7, 2010). You Can't Sacrifice Partition Tolerance, [Online]. Available: https://codahale.com/you-cant-sacrifice-partition-tolerance/.

[11] M. T. Özsu and P. Valduriez, *Principles of distributed database systems.* Springer Science & Business Media, 2011.

[12] M. Shertil, «TRADITIONAL RDBMS TO NOSQL DATABASE: NEW ERA OF DATABASES FOR BIG DATA», Dec. 2016.

[13] H. Kakavand, N. Sevres, and B. Chilton, «The Blockchain Revolution: An Analysis of Regulation and Technology Related to Distributed Ledger Technologies», *SSRN Electronic Journal*, Jan. 2017. DOI: 10.2139/ssrn.2849251.

[14] M. Iansiti and K. R. Lakhani, «The Truth about Blockchain», 2017.

[15] M. Smith, «Luca Pacioli: The Father of Accounting», *SSRN Electronic Journal*, Jan. 2013. DOI: 10.2139/ssrn.2320658.

[16] IOTA Foundation. (Oct. 7, 2019). Frequently asked questions. Common questions about the IOTA Protocol and Tangle., [Online]. Available: https://www.iota.org/get-started/faqs.

[17] J. Bergquist, A. Laszka, M. Sturm, and A. Dubey, «On the Design of Communication and Transaction Anonymity in Blockchain-Based Transactive Microgrids», Dec. 2017, pp. 1–6. DOI: 10.1145/3152824.3152827.

[18] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, «An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends», Jun. 2017. DOI: 10.1109/BigDataCongress.2017.85.

[19] A. Baliga, «Understanding Blockchain Consensus Models», 2017.

[20] L. Lamport, R. Shostak, and M. Pease, «The Byzantine Generals Problem», *ACM Trans. Program. Lang. Syst.*, vol. 4, Feb. 2002. DOI: 10.1145/357172.357176.

[21] M. Castro and B. Liskov, «Practical Byzantine Fault Tolerance», *OSDI*, Mar. 1999.

[22] T. Nguyen and K. Kim, «A survey about consensus algorithms used in Blockchain», *Journal of Information Processing Systems*, vol. 14, pp. 101–128, Jan. 2018. DOI: 10.3745/JIPS.01.0024.

[23] M. Sabt, M. Achemlal, and A. Bouabdallah, «Trusted Execution Environment: What It is, and What It is Not», Aug. 2015, pp. 57–64. DOI: 10.1109/Trustcom.2015.357.

[24] Intel Corporation. (Nov. 12, 2018). Journal, [Online]. Available: https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/journal.html.

[25] ——, (Nov. 12, 2018). PoET 1.0 Specification, [Online]. Available: https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/poet.html.

[26] M. Schwarz, S. Weiser, and D. Gruss, «Practical Enclave Malware with Intel SGX», in. Jun. 2019, pp. 177–196. DOI: 10.1007/978-3-030-22038-9_9.

[27] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, «On Security Analysis of Proof-of-Elapsed-Time (PoET)», Oct. 2017, pp. 282–297, ISBN: 978-3-319-69083-4. DOI: 10.1007/978-3-319-69084-1_19.

[28] D. Bit. (Mar. 8, 2018). 9 Types of Consensus Mechanisms That You Didn't Know About, [Online]. Available: https://medium.com/the-daily-bit/9-types-of-consensus-mechanisms-that-you-didnt-know-about-49ec365179da.

[29] T. Jenks. (Mar. 8, 2018). Pros and Cons of Different Blockchain Consensus Protocols, [Online]. Available: https://www.verypossible.com/blog/pros-and-cons-of-different-blockchain-consensus-protocols.

[30] A. Narayanan. (Sep. 8, 2015). "Private blockchain" is just a confusing name for a shared database, [Online]. Available: https://freedom-to-tinker.com/2015/09/18/private-blockchain-is-just-a-confusing-name-for-a-shared-database/.

[31] I.-C. Lin and T.-C. Liao, «A survey of blockchain security issues and challenges», *International Journal of Network Security*, vol. 19, pp. 653–659, Sep. 2017. DOI: 10.6633/IJNS.201709.19(5).01.

[32] K. Francisco and D. Swanson, «The supply chain has no clothes: Technology adoption of blockchain for supply chain transparency», *Logistics*, vol. 2, no. 1, p. 2, 2018.

[33] V. Buterin, «Ethereum White Paper: A next-generation smart contract and decentralized application platform», 2013.

[34] C. Clack, V. Bakshi, and L. Braine, «Smart Contract Templates: foundations, design landscape and research directions, C.D.Clack, V.A.Bakshi and L.Braine. arxiv:1608.00771. 2016», Aug. 2016.

[35] M. Bellini. (Dec. 17, 2018). Smart Contracts: che cosa sono, come funzionano quali sono gli ambiti applicativi, [Online]. Available: https://www.blockchain4innovation.it/mercati/legal/smart-contract/blockchain-smart-contracts-cosa-funzionano-quali-gli-ambiti-applicativi.

[36] O. Hart. (Jul. 24, 2019). Nobel Prize-Winning Economist Shares His Thoughts On Smart Contracts, [Online]. Available: https://www.youtube.com/watch?time_continue=142&v=Ee_3Nvl-1GE.

[37] M. Giancaspro, «Is a 'smart contract' really a smart idea? Insights from a legal perspective», *Computer Law & Security Review*, vol. 33, Jun. 2017. DOI: 10.1016/j.clsr.2017.05.007.

[38] P. McCorry, S. Shahandashti, and F. Hao, «A Smart Contract for Boardroom Voting with Maximum Voter Privacy», Jan. 2017.

[39] K. Christidis and M. Devetsikiotis, «Blockchains and Smart Contracts for the Internet of Things», *IEEE Access*, vol. 4, pp. 1–1, Jan. 2016. DOI: 10.1109/ACCESS.2016.2566339.

[40] M. Al-Bassam, «SCPKI: A Smart Contract-based PKI and Identity System», Apr. 2017, pp. 35–40, ISBN: 978-1-4503-4974-1. DOI: 10.1145/3055518.3055530.

[41] M. E. Porter, *Competitive advantage: Creating and sustaining competitive advantage*. New York: Free Press, 1985.

[42] J. Mentzer, W. Dewitt, J. Keebler, S. Min, N. Nix, C. Smith, and Z. Zacharia, «Defining Supply Chain Management», *Journal of Business Logistics*, vol. 22, Sep. 2001. DOI: 10.1002/j.2158-1592.2001.tb00001.x.

[43] G. Stevens, «Integrating the Supply Chain», 8, vol. 19, 1989, pp. 3–8. DOI: 10.1108/EUM0000000000329.

[44] L. D. Fredendall and E. Hill, *Basics of supply chain management*. CRC Press, 2016.

[45] M. Johnson and G. Stevens, «Integrating the Supply Chain... 25 years on», *International Journal of Physical Distribution & Logistics Management*, vol. 46, Feb. 2016. DOI: 10.1108/IJPDLM-07-2015-0175.

[46] S. Li, B. Ragu-Nathan, T. Ragu-Nathan, and S. S. Rao, «The impact of supply chain management practices on competitive advantage and organizational performance», *Omega*, vol. 34, no. 2, pp. 107–124, 2006.

[47] A. Rai, R. Patnayakuni, and N. Seth, «Firm performance impacts of digitally enabled supply chain integration capabilities», *MIS quarterly*, pp. 225–246, 2006.

[48] B. Sezen, «Relative effects of design, integration and information sharing on supply chain performance», *Supply Chain Management: An International Journal*, vol. 13, no. 3, pp. 233–240, 2008.

[49] K. Korpela, J. Hallikas, and T. Dahlberg, «Digital supply chain transformation toward blockchain integration», in *proceedings of the 50th Hawaii international conference on system sciences*, 2017.

[50] S. Saberi, M. Kouhizadeh, J. Sarkis, and L. Shen, «Blockchain technology and its relationships to sustainable supply chain management», *International Journal of Production Research*, vol. 57, no. 7, pp. 2117–2135, 2019.

[51] H. M. Kim and M. Laskowski, «Toward an ontology-driven blockchain design for supply-chain provenance», *Intelligent Systems in Accounting, Finance and Management*, vol. 25, no. 1, pp. 18–27, 2018.

[52] E. Olszewski. (May 16, 2019). Why Blockchain Matters To Enterprise (Hint: It's Not Because Of Decentralization), [Online]. Available: https://medium.com/@eolszewski/why-blockchain-matters-to-enterprise-hint-its-not-because-of-decentralization-8c38674f43c6.

[53] G. Perboli, S. Musso, and M. Rosano, «Blockchain in Logistics and Supply Chain: a Lean approach for designing real-world use cases», *IEEE Access*, vol. PP, pp. 1–1, Oct. 2018. DOI: 10.1109/ACCESS.2018.2875782.

[54] M. P. Caro, M. S. Ali, M. Vecchio, and R. Giaffreda, «Blockchain-based traceability in Agri-Food supply chain management: A practical implementation», in *2018 IoT Vertical and Topical Summit on Agriculture-Tuscany (IOT Tuscany)*, IEEE, 2018, pp. 1–4.

[55] F. Yiannas, «A new era of food transparency powered by blockchain», *Innovations: Technology, Governance, Globalization*, vol. 12, no. 1-2, pp. 46–56, 2018.

[56] M. Lammi. (Mar. 29, 2018). Project SmartLog: blockchain in logistics, [Online]. Available: https://smartlog.kinno.fi/articles/project-smartlog-blockchain-logistics.

[57] K. Sadouskaya, «Adoption of Blockchain Technologyin Supply Chain and Logistics», 2017.

[58] K. Biswas, V. Muthukkumarasamy, and W. L. Tan, «Blockchain based wine supply chain traceability system», in *Future technologies conference*, 2017, pp. 1–7.

[59] S. A. Abeyratne and R. P. Monfared, «Blockchain ready manufacturing supply chain using distributed ledger», 2016.

[60] J.-H. Tseng, Y.-C. Liao, B. Chong, and S.-w. Liao, «Governance on the drug supply chain via gcoin blockchain», *International journal of environmental research and public health*, vol. 15, no. 6, p. 1055, 2018.

[61] S. Figorilli, F. Antonucci, C. Costa, F. Pallottino, L. Raso, M. Castiglione, E. Pinci, D. Del Vecchio, G. Colle, A. Proto, *et al.*, «A blockchain implementation prototype for the electronic open source traceability of wood along the whole supply chain», *Sensors*, vol. 18, no. 9, p. 3133, 2018.

[62] Intel Corporation. (Nov. 12, 2018). Transactions and Batches, [Online]. Available: https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/transactions_and_batches.html.

[63] K. Olson, M. Bowman, J. Mitchell, S. Amundson, D. Middleton, and C. Montgomery, «Sawtooth: An Introduction», Jan. 2018.

[64] The Hyperledger Performance and Scale Working Group. (Oct. 7, 2019). Hyperledger Blockchain Performance Metrics, [Online]. Available: https://www.hyperledger.org/resources/publications/blockchain-performance-metrics.

[65] M. Rubino. (Oct. 31, 2018). A 5 node Hyperledger Sawtooth cluster with IntKey and PoET. 'docker-compose -f sawtooth5.yaml up', [Online]. Available: https://gist.github.com/mtrubs/defd83fb30e3bca85fb76be01d813f04.

[66] Intel Corporation. (Jul. 24, 2019). Using Docker for a Sawtooth Test Network, [Online]. Available: https://sawtooth.hyperledger.org/docs/core/nightly/1-2/app_developers_guide/docker_test_network.html.

[67] ——, (Nov. 12, 2018). Events and Transaction Receipts, [Online]. Available: https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/events_and_transactions_receipts.html.

[68] P. Sitoh. (Dec. 4, 2018). What are the differences between Ethereum, Hyperledger Fabric and Hyperledger Sawtooth?, [Online]. Available: https://medium.com/coinmonks/what-are-the-differences-between-ethereum-hyperledger-fabric-and-hyperledger-sawtooth-5d0fc279d862.

[69] A. Le Hors. (Apr. 23, 2019). Demystifying Hyperledger Fabric ordering and decentralization, [Online]. Available: https://developer.ibm.com/articles/blockchain-hyperledger-fabric-ordering-decentralization/.

[70] chris-j-h. (Aug. 24, 2018). Quorum Whitepaper, [Online]. Available: https://github.com/jpmorganchase/quorum/blob/master/docs/Quorum%5C%20Whitepaper%5C%20v0.2.pdf.

[71] R. G. Brown, «The corda platform: An introduction», 2018.