BLOCKCHAIN FOR SECURE AND TRANSPARENT TRACK-AND-TRACE IN MANUFACTURING

Ernest Kurniawan · Doris Benda · Sumei Sun · Sin Guan Tan · Alex Chin

Abstract. This chapter will first introduce the reader to blockchain technology and the different types of blockchain systems. The underlying principle on how it achieves consensus and guarantees immutability of records in a completely distributed manner is explained. Some use cases where blockchain can address challenging issues in manufacturing environment are given. A particular use case of end-to-end track-and-trace including the manufacturing process on the shop floor is then explained, and showcased in [29] by implementing the blockchain-based track-and-trace system with Hyperledger Fabric 2.0. Additional challenges that need to be tackled pertaining to transaction speed and volume are described. To catalyze the adoption, seamless integration of different blockchain systems is needed. This will involve integrating a number of different blockchain protocols. A discussions and recommended considerations for blockchain interoperability, which allow exchange of information/tokens across different systems running different protocols are then given. Finally, this chapter ends with a summary where some of the best practices for a successful blockchain deployment are discussed.

Keywords. Blockchain, Track-and-Trace, Hyperledger Fabric 2.0, Interoperability

Ernest Kurniawan · Doris Benda · Sumei Sun
Institute for Infocomm Research, A*STAR, 1 Fusionopolis Way, #21-01 Connexis (South Tower), Singapore 138632
e-mail: {ekurniawan,Doris_Benda.sunsm}@i2r.a-star.edu.sg

Sin Guan Tan · Alex Chin
Advanced Remanufacturing and Technology Centre, A*STAR, 3 Cleantech Loop, #01-01, Cleantech Two, Singapore 637143
e-mail: {tan_sin_guan,Alex_Chin}@artc.a-star.edu.sg

Toro, Akhtar, Wang et al. (Eds.): Implementing Industry 4.0 - The Model Factory as the key enabler for the Future of Manufacturing.
springerlink.com

1. Blockchain Introduction

Blockchain is a distributed ledger technology that uses cryptography to secure its data against tampering, making them immutable and almost impossible to alter once recorded. The term blockchain comes from the fact that records or transactions are stored in blocks, which are then concatenated together forming a chain of blocks. Each block consists of a data portion which stores a list of records or transactions, a cryptographic hash string which acts as a signature of the block, and a header field. Included in the header field is a timestamp and the hash string from the previous block, creating an inter-dependency between blocks. The hash string of a block is unique to the content of the block, and it is generated by a cryptography algorithm such as SHA-256 [25]. Any modification to the content of the block would result in a completely different hash string. In this way, any alteration to the records or transactions within a block can be easily detected. Furthermore, for anyone to be able to tamper with the ledger undetected, he/she has to recalculate the hash of the block as well as all the subsequent blocks, which would require an enormous amount of resources that must be at least larger than half the computational power of the network as explained in the later part of this section.

An illustration of blockchain structure is given in Figure 1. The first block in the blockchain is a special block called Genesis Block. Its header does not contain any reference to the hash string of another block, and its sole purpose is to provide a solid starting point from which subsequent blocks can be built upon. For the rest of the blocks, it contains a list of transactions, a signature hash string, and a header that includes the signature hash of the previous block.

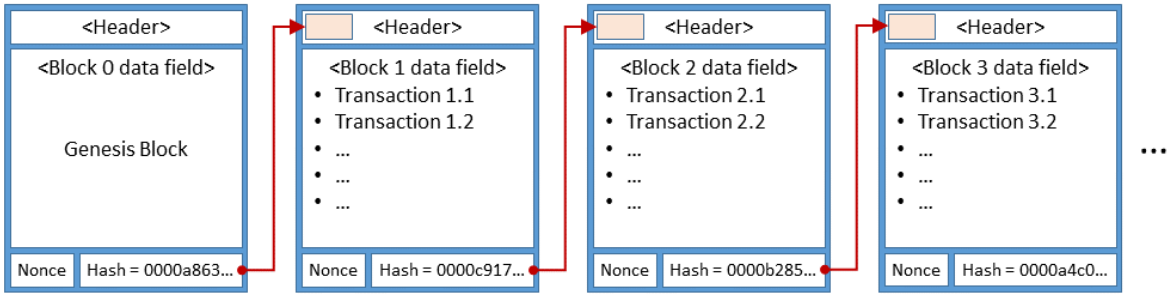

Figure 1 Illustration of Blockchain Structure

A blockchain network comprises of multiple peer-to-peer nodes, each keeping a full record of the blockchain ledger. To ensure that the state of the ledger across all nodes are consistent, a consensus algorithm is used. With the consensus algorithm, a new block can be added onto the blockchain ledger if and only if all the nodes in the peer-to-peer network agree on the validity of the block, hence keeping the state of the blockchain consistent.

Different types of consensus algorithms have been used in blockchain. We will describe next a few most popular consensus algorithms.

- **Proof of Work (PoW):** The main idea of PoW consensus algorithm is to make it computationally expensive to validate a new block such that it is difficult for an adversary to add a malicious block onto the blockchain. With PoW, the hash string of a block must satisfy a certain structure, e.g. it must start with a certain number of leading zeros. The nonce is an auxiliary bit string that are used to tune the resulting hash of a block. When a blockchain adopts PoW as its consensus algorithm, some peers in the network will serve as miners, whose role is to validate new blocks. When a new block is constructed, all of the miner peers engage in a competition to find the right nonce that produces a hash string satisfying the required structure. The first miner who successfully validates the block will receive a reward, and then broadcasts the new block to all the peers to verify and append to their copies of the blockchain. For an adversary to manipulate a blockchain, it has to hold more than half of the *computational* power of all the miners.

- **Proof of Stake (PoS):** The PoS consensus algorithm is used to address the main drawback of PoW in terms of speed and power consumption. While it is easy to compute a hash string of a block in PoW, a miner has to try many different nonce patterns before finding the right one producing the hash string with the right structure. This puts a limit on how fast new transactions can be recorded into a blockchain. In addition, the computational burden required for the mining process results in high power consumption. With the PoS consensus algorithm, instead of mining process, the peer nodes are simply required to put a bet on the new block that it has verified and believed to be valid. The block that receives the largest vote with the most bet will then be elected as the new block to be added to the ledger, and to be broadcasted to all the peer nodes to update their copies of the blockchain. With PoS, an adversary would have to hold more than half of the *voting* power of the network in order to manipulate a blockchain.

Further modification to the PoS algorithm has been proposed whereby the voter loses its stake when the block it has voted does not win the majority. Hence the integrity of the blockchain is enhanced by further discouraging misbehavior by adversarial node. Other consensus algorithms such as Proof of Authority (PoA), multiparty consensus, leader-follower model, are also available.

Being a decentralized system, blockchain does not have a single point of failure as in centralized databases. In addition, since the records stored in the blockchain are immutable and a new block can only be added with the consensus of all the peer nodes, it can establish trust between peers to perform transactions in an auditable and transparent manner.

The first successful implementation of blockchain was done in 2008 under the pseudonym Satoshi Nakamoto [24], in which a distributed transaction ledger was used as base for a cryptocurrency called Bitcoin. Blockchains have since expanded to several other areas notably in the financial, manufacturing, asset management, and supply chain industry.

There are three major classifications of blockchain, namely public, consortium, and private blockchains. Each of these classes has a different read/write access policy on the distributed ledger as well as a different required permission to join a blockchain network. In public blockchain, anyone with a computer and internet connection can participate in a blockchain network. In addition, anyone in the network has full access to read and write onto the ledger. Because of this, public blockchain has poor scalability, and the speed at which transactions can be written into the blockchain is limited. Private blockchain, on the other hand, are run by a single organization or a group of organizations who knows one another and has established a certain degree of trust among them. Only authorized party is allowed to have access to the ledger, therefore it has less stringent requirement for the consensus algorithm and result in better support to accommodate higher transaction throughput. Consortium blockchain is a compromise between public and private blockchain, whereby a large group of organizations are forming a consortium and in agreement to jointly maintain a blockchain network. Access control to the ledger is managed by the administrators from the participating organizations. A summary of the characteristics for each type of blockchain is given in Table 1.

Table 1 Types of Blockchains and Their Characteristics

Property	Public Blockchain	Consortium Blockchain	Private Blockchain
Examples	Bitcoin, Ethereum	Quorum, Hyperledger Fabric	Ripple, Hyperledger Fabric
Identity	Anonymous	Several organizations/ Known identities	Several organizations/ Known identities
Access	Permissionless (anyone can access/create transactions, run nodes, open-source smart contract code)	Permissioned (closed/controlled ecosystem)	Permissioned (closed/controlled ecosystem)
Number of Users	Millions	Hundreds of thousands	A few hundreds

Control	Fully decentralized	Partially centralized	Centralized
Tamper-proof	Nearly impossible to tamper with, No single point of failure, 100% transparent	Blockchain operated by organizations (small risk of tampering/ some point of failure)	Blockchain operated by organizations (small risk of tampering/ some point of failure)
Consensus Algorithm	Proof-of-work, Proof-of-stake	Multi-party consensus/Voting, Proof-of-authority, Selection of trusted leader nodes, Leader-follower model	Multi-party consensus/Voting, Proof-of-authority, Selection of trusted leader nodes, Leader-follower model
Finality	No finality guaranteed (Height of ledger/Number of confirmations)	Finality guaranteed (Trusted environment)	Finality guaranteed (Trusted environment)
Currency	Monetary incentives to miners (e.g. Bitcoins, Ether)	No monetary incentives required, No mining	No monetary incentives required, No mining
Efficiency/Performance	Low (5-7 transactions per sec for Bitcoin)	Medium	High
Speed	Slow (10 mins to add a block for Bitcoin)	Medium	Fast
Scalability	Low	Medium	Medium
Energy Consumption	Extremely high	Low	Low

In the next section, we will focus on the manufacturing application and describe a few use cases where blockchain can be used as a promising solution to the long-standing problems faced by the industry.

2. Manufacturing Use Cases

According to Gartner Research, the added business value of blockchain will grow to more than $176 billion by 2025, and then exceed $3.1 trillion by 2030 [10]. The manufacturing sector has been in the forefront of blockchain adoption and deployment globally, only trailing behind the financial service sector [27]. These business values in manufacturing sector are derived from blockchain capabilities in providing transparency and auditability of records, eliminating the need for intermediaries for settlements, as well as ensuring the authenticity and immutability of information. Some of the use cases of blockchain in manufacturing are listed in the following.

Supply Chain Monitoring

In manufacturing, supply chain requires horizontal integration [3] of both upstream where raw materials are sourced for production as well as downstream where end products are distributed and delivered to the consumer. By providing greater transparency to multiple stakeholders along the supply chain who engage in cross-border or cross-country transactions, bottlenecks can be quickly identified and production delay can be prevented. In this case, blockchain network serves as a reliable platform for information sharing that provides near real-time access to goods movement status along the supply chain. The availability of information sharing platform where all the participants are authenticated and records are immutable will also enable manufacturer to reliably connect to more suppliers in the upstream network. In addition, better identification of the geographic location where most demand for the product is concentrated in the downstream network can be made. This eventually leads to better selection of supplier and distributor.

By combining blockchain with Internet of Things (IoT) technology such as connected sensors, a more fine grained monitoring information along the supply chain network can be made available. For

example, the temperature of the container when the raw material is shipped, the vibration from the truck as the product is transported cross-countries, and so on, will serve as valuable information to the manufacturer to pinpoint any potential issue and quickly source for alternative. With better monitoring of supply chain and transparent data sharing enabled by blockchain, the overall efficiency and profitability of the manufacturer can be improved.

Asset Tracking

Keeping track of the movement of assets such as expensive manufacturing equipment and logging the utilization information onto the blockchain ledger will result in better visibility and transparency of asset status. This will not only provide the technicians a means to quickly check the availability of equipment, but also allows for constant monitoring of asset condition. Some of the benefits in this case include more efficient equipment utilization through better scheduling, faster identification on when the equipment is due for maintenance hence avoiding downtime due to equipment breakdown, as well as a complete audit trail of who has used the equipment in the past along with the historical maintenance data.

When sharing of the asset tracking data in the blockchain is extended to the equipment manufacturer, additional service such as predictive and preventive maintenance can be enabled. Furthermore, by understanding the utilization pattern, the equipment manufacturer can apply dynamic pricing to the maintenance subscription cost that will benefit both the equipment owner and manufacturer.

Contract Management

In dealing with other organizations such as suppliers and distributors, the traditional approach for manufacturers often requires lengthy process of negotiation, payment, delivery, which involves intermediary to achieve settlement. With blockchain, a smart contract can be written to automate those processes and eliminate the need for intermediary. For example, a smart contract can be designed to automatically trigger payment to the supplier when the confirmed goods receipt has been recorded within a predefined time frame. This results in significant improvement in efficiency and eliminate delays. The smart contract can also be written to handle dispute, such that the total payable is reduced by a certain agreed amount depending on how late the goods are delivered, resulting in a much faster dispute resolution.

End-to-end Track-and-Trace

Blockchain is able to improve product traceability and provide better transparency in tracking provenance. By recording onto the blockchain ledger the history of every raw material in terms of where they are originating from and how they are sourced, the history of every step of the process in manufacturing, as well as the journey of the finished product from factory the consumer, a complete end-to-end track-and-trace is achieved. Storing the track-and-trace information onto the blockchain has added benefits whereby the identity of the organization providing the input can be verified and the records are immutable once stored, hence providing authenticity guarantee of the information. This enables a faster response to product recall for cases such as contaminated ingredients or defective parts, as the track-and-trace data can pinpoint accurately which batch of products is affected. When sharing of track-and-trace data is extended to the consumer, a better transparency about the product provenance will equip the consumer to make informed decision in selecting a product, which eventually lead to better trust and brand loyalty.

2.1. Consideration to Blockchain Adoption and Deployment

In general, any activity involving multiple parties who do not necessarily trust one another and making use of shared information to engage in coordinated activity and value exchange, will be able to benefit from blockchain. However, before deciding to use blockchain as a solution, the right type of blockchain needs to be chosen, whether public permission-less or private permissioned blockchain is more appropriate. For manufacturing use cases, private or consortium blockchain is more suitable, as the stakeholders are parties who have already established a certain degree of trust between one another. In addition, the choice of private over public blockchain brings about additional advantages as follows:

- Greater scalability to accommodate larger transaction volume
- Higher energy efficiency as consensus algorithm other than Proof of Work can be adopted
- Better control for security, privacy, and data access rights
- Flexibility for smart contract development and integration with other enterprise systems

Hyperledger [18] is the fastest-growing open-source project under the Linux Foundation since its founding in December 2015. As of date, it is the most widely used enterprise blockchain standard deployed in private/consortium-based blockchain networks. It consists of several blockchain frameworks, utility libraries, and related tools; among them is the most popular blockchain framework called Hyperledger Fabric. Therefore, Hyperledger Fabric is among the best candidates for blockchain implementations on the manufacturing floor. In Section 3, an implementation of Hyperledger Fabric 2.0 for achieving end-to-end track-and-trace in manufacturing is discussed in detail. Additional challenges for including the manufacturing process information from the shop floor into the track-and-trace data pertaining to the transaction speed and volume are also addressed.

Another important consideration for blockchain deployment is the ecosystem. For a successful deployment of a blockchain network, all stakeholder organizations will need to take part and contribute in the maintenance of the network. For this purpose, the ease of blockchain node installation and the flexibility in the options to install (whether on the cloud servers or on-premise facility) are important. Furthermore, since different organizations may be at different stages on their blockchain journey, it can be expected that some of them have already had their own blockchain deployment, which may use different platform. In order to incorporate this, it is necessary to enable interoperability between different blockchain platforms so that information sharing between them can be facilitated while maintaining the immutability of records and authenticity of transactions. Section 4 presents an in-depth discussion on blockchain interoperability, including the different types of inter-blockchain interaction and how exchanges of different ledger object types can be achieved.

Last but not least is the regulatory consideration for blockchain deployment. While most deployments of blockchain are still at an early experimental or proof-of-concept stage [2], adoption to the mainstream production system is constantly on the rise. It is expected that at some point the regulators and standardization bodies will intervene to define the course of blockchain adoption. While keeping a close engagement with the regulators to set the blockchain agenda, organizations that see values in blockchain are moving ahead with early adoption. Choosing a modular blockchain platform such as Hyperledger Fabric 2.0 will give an advantage in this case, since adjustments to the blockchain network are relatively easier to incorporate along with evolving regulation.

3. End-to-End Track-and-Trace System Implementation with Hyperledger Fabric 2.0

Improving product traceability is one of the biggest drivers of blockchain adoption in manufacturing [2]. With the growing demand for fine-grained track-and-trace of products along the supply chain, the benefits of using blockchain for manufacturing process tracking become obvious. There have been some commercial solutions for track-and-trace ([1], [21], [26], [28]). However, the focus of these commercial solutions is mainly to keep track of cross-organizational transactions, excluding the manufacturing process data from the manufacturing floor. With the trend that manufacturing is heading toward hyper-personalization where each product is made specific to the individual consumer and the batch production size can be as low as one, the importance of keeping track of the individual manufacturing step for a product becomes more prominent. The developed solution presented in this chapter is shown in [29].

One notable challenge to record each step in the manufacturing process as a blockchain transaction is the high throughput as well as high volume of transactions that the blockchain must handle. The distributed nature of a blockchain network and the requirement to maintain consensus among its peer for every recorded transaction restricts the transaction throughput and volume that can be supported. In this section, discussion on how to support high throughput and high volume transaction to enable track-and-trace system involving process data from manufacturing shop floor is presented. In particular, we

consider Hyperledger Fabric version 2.0 [14] which was released on 30th January 2020 and is the newest version available at the time of this write-up, as the blockchain platform used throughout the discussion in this chapter.

3.1. Transaction Processing in Hyperledger Fabric 2.0

Before explaining how transactions are processed and recorded, it is helpful to understand the structure and different components forming the blockchain network in Hyperledger Fabric 2.0. At high level, a Fabric network comprises of the following elements:

- Ordering service who establishes consensus on the order of transactions in the block and distribute them to the peers
- Membership service provider who maintains the mapping between a cryptographic identity with the associating entity in the network
- The ledger containing the world state of all the business objects maintained, along with the blockchain of transaction logs that lead to the current world state
- Peer nodes who keep a copy of the ledger and host a smart contract
- Chaincode smart contract that is run within a container environment (such as Docker) and executed at the Peer nodes
- Endorsement and validation policy enforcement
- Client applications that generate transactions and modify the world state in the ledger

Most blockchain systems that support smart contract such as Ethereum and Quorum, adopt order-execute architecture for transaction processing, whereby the consensus protocol first validates and orders the transactions before propagating them to all the peer nodes. Each peer node will then execute the transaction sequentially. The order-execute architecture requires the smart contract execution to be deterministic to ensure consensus is eventually reached. For this reason, domain specific language (such as Solidity in Ethereum) is required.

Hyperledger Fabric adopts a different architecture known as execute-order-validate, which allows the Chaincode smart contract to be written in standard programming language (such as Java, Go, and Javascript). This new architecture also allows parallel execution, resulting in better performance and scalability. The transaction processing in Hyperledger Fabric involves three phases as explained in the following.

Phase 1: Proposal

Transaction workflow starts with a client application generating a number of transaction proposals to be sent to one or more peer nodes in the network. There are two types of transactions that the client application might generate, namely the query and invoke transaction.

The query transaction, as the name suggests, involves reading some values from objects stored in the ledger world state without modifying it. Therefore, query transaction only needs to be sent to only one peer, and since all peers have an identical copy of the ledger world state, it does not matter which peer the transaction is sent to. Query transactions will not be sent to the Ordering service node, and therefore they are not stored into the blockchain.

The invoke transaction, on the other hand, involves reading and/or writing to the ledger world state. In this case, the endorsement policy will determine which peer nodes the transaction proposal need to be submitted to. Upon receiving the transaction proposal, the endorsing peer will simulate the execution of the transaction in the Chaincode smart contract (which can be done in parallel), and keep a read/write set. The read set contains a list of unique keys that are read by the transaction, along with their committed version numbers. The corresponding values of those keys are not stored in the read set. The write set, on the other hand, contains a list of unique keys whose values are modified by the transaction, along with their new values. For the case where the transaction deletes a particular key, a delete marker is stored in place of the new value. It is important to note that at this stage, no modification is made to the ledger world state. The endorsing peer will then sign the transaction proposals and respond to the client application. The message flow during this phase 1 transaction proposal is illustrated in Figure 2.

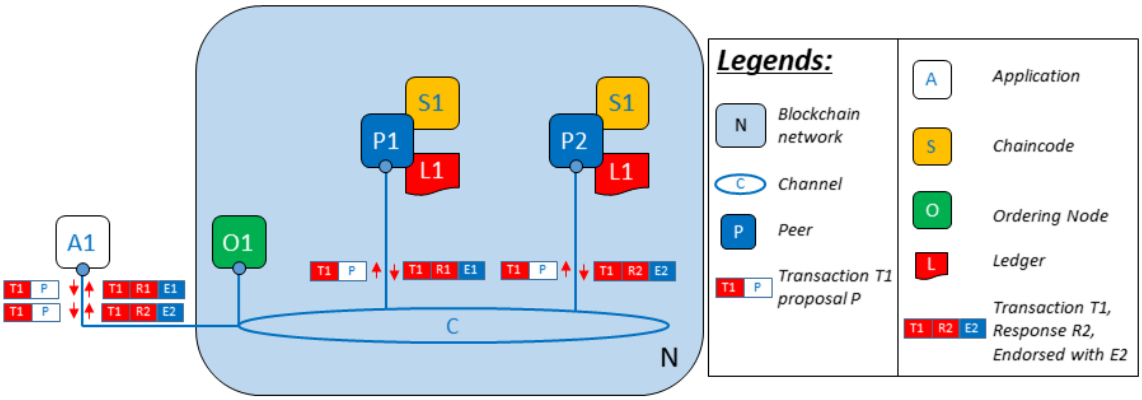

Figure 2 Illustration of Message Flow during Phase 1 Transaction Proposal Stage (Based on the example from [15]). Image credit to Hyperledger Fabric documentation, and subject to Creative Common Attribution 4.0 International License [3]

Upon receiving the endorsed transaction responses, the client application may optionally check for the consistency of all responses from the endorsing peers, and discard inconsistent transactions for early termination. Although this can improve the efficiency of the overall blockchain network, the client application is not required to perform such check; in which case the inconsistent transactions will be marked as invalid only during committing stage on phase 3 and will not alter the ledger state.

Phase 2: Ordering and Packaging Transactions into Blocks

The next step is for the client application to forward the endorsed transactions to the Ordering service node. After receiving the endorsed transactions from several client applications, the Ordering service will arrange batches of endorsed transactions into a well-defined sequence and pack them into blocks. These blocks will then be saved and distributed to all the peers to be included in their copy of the blockchain. An illustration of the message flow during the transaction order and packing phase is given in Figure 3.

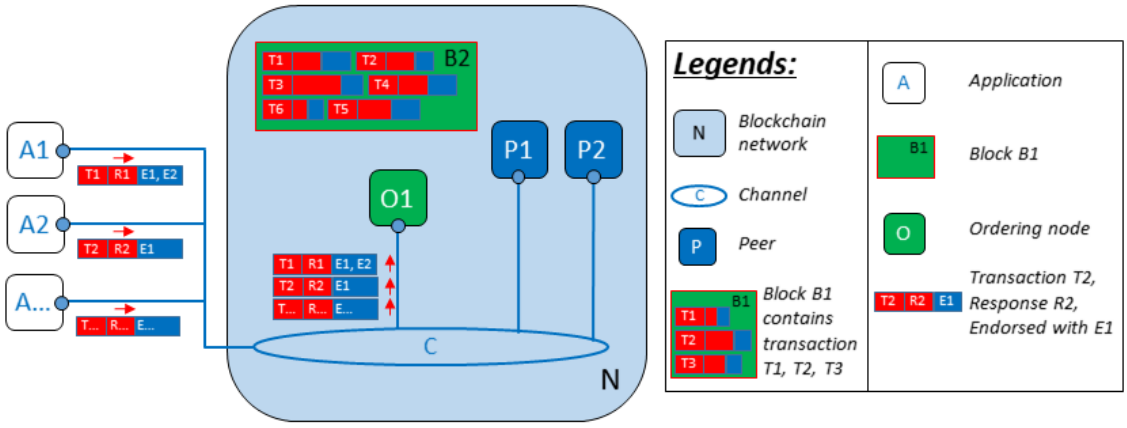

Figure 3 Illustration of Message Flow during Phase 2 Transaction Ordering and Packaging Stage (Based on the example from [16]). Image credit to Hyperledger Fabric documentation, and subject to Creative Common Attribution 4.0 International License [3]

There are two important parameters that influence the way the Ordering system node batches the endorsed transactions into blocks. First parameter is called *MaxMessageCount*, which determines the maximum number of transactions to be batched into a single block. The second parameter is called *BatchTimeout*, which determines the duration after which a new block will be created even though the total number of received endorsed transactions is still less than the *MaxMessageCount*. A new block will be created whenever either condition (total number of transactions reaches *MaxMessageCount* or total duration since the previous block creation exceeds *BatchTimeout*) is

satisfied. In Hyperledger Fabric 2.0, the default value of *MaxMessageCount* is 10, and the default value of *BatchTimeout* is 2 seconds. It is apparent that both of these parameters have significant impact to the throughput and latency of the transactions in the blockchain. Large values of *MaxMessageCount* and *BatchTimeout* can increase the throughput since less overhead is introduced in the creation of a new block. However, the latency can be large as it takes some times for the ordering system to accumulate sufficient number of endorsed transactions for the block creation. On the contrary, small values of *MaxMessageCount* and *BatchTimeout* can reduce the latency at the expense of throughput reduction.

Phase 3: Validation and Commitment

Once the new block has been distributed to all the peers, each peer will perform independent validation to the new block and confirm that the endorsement policy is fulfilled. Correspondingly, each peer will sequentially process each transaction included in the new block and check for their consistency before committing it to the ledger. This is performed with the help of the read/write sets that were generated during the first phase. In particular, for the keys included in the read set, the version number during which the transaction was simulated in phase one must match the current version of those keys prior to commitment. Otherwise, the transaction is marked as invalid and it will not be committed. This is to ensure that the value of the key that was read during the execution is still having the same value during the commit process, hence consistency of the ledger is maintained. For the keys included in the write set, the new values recorded in the set will be used to update the ledger world state, and correspondingly the version number associated with the key will also be updated. All transactions in the new block, regardless of whether or not they get invalidated, will still be stored into the blockchain for audit purposes. Within the committed block, a flag is used to distinguish the invalid transactions from the valid ones. Only the valid transactions will be used to update the world state. The invalid transactions will not update the world state. An illustration of the message flow during the validation and commitment phase is given in Figure 4.

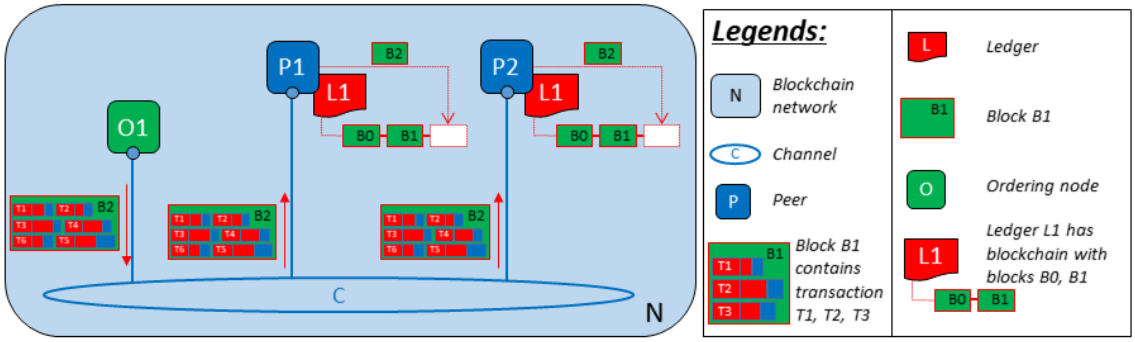

Figure 4 Illustration of Message Flow during Phase 3 Validation and Commitment (Based on the example from [17]). Image credit to Hyperledger Fabric documentation, and subject to Creative Common Attribution 4.0 International License [3].

It is worth noting that no Chaincode execution is performed during the validation process. Only endorsing nodes during the first phase of transaction proposal need to execute the Chaincode. This is beneficial in terms of achieving better confidentiality, as it keeps the logic of the Chaincode known to only the endorsing peers. In addition, by limiting the Chaincode execution to the endorsing peers, the overall scalability of the network can be improved.

As a final step, it is necessary for the client applications to be notified about the outcome of the transactions that were submitted. This is achieved through event triggering mechanism, whereby a client application can subscribe to a transaction event to be notified that the transaction has been successfully ordered, validated, and committed to the ledger. For the invalidated transactions, the client application can implement exception handling mechanism to either resubmit the transaction or aggregate it with a new transaction.

3.2. Multi Version Concurrency Control

As explained in the previous subsection, the transaction processing in Hyperledger Fabric 2.0 is a three-phase process, and it follows execute-order-validate architecture. Due to this, version control is required to ensure the state of the ledger has not been changed from the initial execution to the final commitment. This Multi Version Concurrency Control (MVCC) is critical in maintaining the consistency of the ledger and to prevent issues such as the double spending problem [23], whereby a user is able to spend his/her cryptocurrency more than once.

When multiple transactions that are interacting with the same key in the ledger are packed into the same block, there will be a higher chance for the MVCC to raise an issue. In general, any transaction that reads a value from a certain key in the ledger will generate a read set whereby the version number of that particular key is kept. In addition, any transaction that writes a new value to a certain key in the ledger will result in the update to the version number of that same key. Therefore, when the write transaction to a particular key is followed by the read transaction in the same block, an MVCC issue will be raised, invalidating the read transaction as the value of the key during simulation execution is different from the value during commit phase. Similarly, two consecutive transactions in a block that reads and then update the same key will cause MVCC issue, which will invalidate the second transaction.

In order to support the track-and-trace application of manufacturing process data, a high throughput transaction that potentially interact with the same key in the ledger need to be supported. For example, in tracking the location of a carton box as it is transported across different sections of the conveyor belt, multiple transactions will be generated to update the carton location within a short period of time. When these transactions are packaged into the same block, MVCC issue will be raised, resulting in a lot of invalidated transactions.

One possible workaround to avoid the MVCC issue is to control the number of transactions to be packed into a block. By setting the *MaxMessageCount* parameter to one, each block will contain only one transaction, and MVCC issue is removed altogether. However, although this will improve the latency of a transaction, this will have detrimental impact on the overall throughput of the blockchain due to higher overhead in block creation. Other methods to add secondary execution phase after order and validation phase has also been shown to improve the throughput of Hyperledger Fabric and resolve MVCC issue [12].

Our solution to avoid the MVCC issue is to divide into small pieces the world state object data structure (which for the case of Hyperledger Fabric 2.0 is stored as couch DB document) until each piece can be updated without having to read the state of the object before writing it to the ledger, i.e. each small piece of the couch DB document consists only of one key and one value. This value field can be overwritten when updating without having to read any other value field from the ledger, because no other value field is part of this couch DB document. This approach works very well in manufacturing because we can guarantee that between two updates there will always be a small time passed in between, i.e. two updates cannot be done at arbitrary small time interval between each other on a conveyer belt section, therefore there is always a small positive time passed between the two updates.

Figure 5 illustrates the approach of dividing the object data structure for “Carton Box ID” into two, whereby a new small object data structure “Carton Box ID_Location” is used to prevent MVCC issue. The “Location” value field in the “Carton Box ID_Location” object can be overwritten when updating without having to read any other value field from the ledger. Therefore, the “Location” field can be updated frequently without causing MVCC issues in this object. In contrast, the “Location” value field in the “Carton Box ID” object cannot be updated frequently because the other value fields have to be read before updating the object causing MVCC issues if the object is updated several times per block.

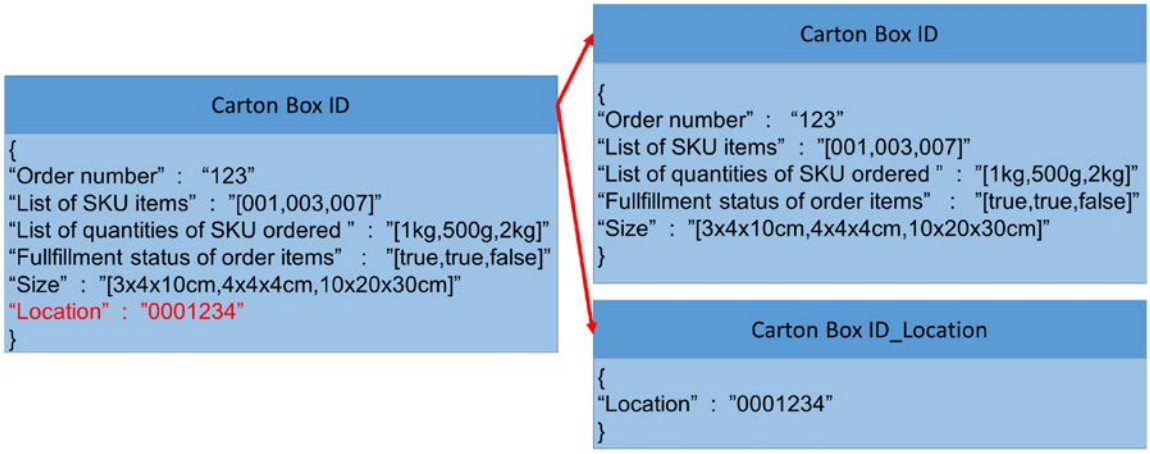

Figure 5 Illustration of the data structure to avoid MVCC issues.

3.3. Latency and Throughput

For track-and-trace application in manufacturing, different elements in the manufacturing floor generate trackable information, some of which are sensitive to delay. As such, latency and throughput are two important considerations when storing the track-and-trace data. This subsection will introduce the concept of latency and throughput in blockchains to deepen the understanding of the correlation between the write transaction latency and the *BatchTimeout*/*MaxMessageCount* value, and hence the likelihood that an MVCC conflict will occurs. In addition, the concept of a network-wide ledger view [13] is introduced in this section which is a critical component for discussing blockchain interoperability in Section 4.

3.3.1. Latency

Without loss of generality, we assume that the number of nodes N in the blockchain network is known. This assumption is justified for permissioned blockchains since the network is run by a small number of known nodes controlled by an organization or several organizations. For networks with unknown number of nodes, such as public blockchains, the reader is referred to [19] for a more detailed discussion. Fair comparison of blockchain latencies can only be done if the deployment conditions are similar, especially with respect to the number of nodes participating in the consensus. The complexity of consensus algorithms increases with the number of nodes and it also depends on the deployment conditions, such as on premise or on a wide geographically distributed network. These factors have to be considered for a fair comparison.

Read transaction latency (cf. Figure 6) is defined as the duration passed from the initial request by the client until its reply is received. This usually involved the interaction with exactly one network node since every node stores the complete history of the ledger locally. In contrast, write transaction latency (cf. Figure 6) has to take a network-wide ledger view approach. It is defined as the duration passed from the initial request by the client until its effects are committed to the ledger of a given threshold of nodes in the network. It is considerably higher than the read transaction latency since time for propagation of the transaction to several nodes and an often massive message exchange during the consensus algorithm have to be accounted for.

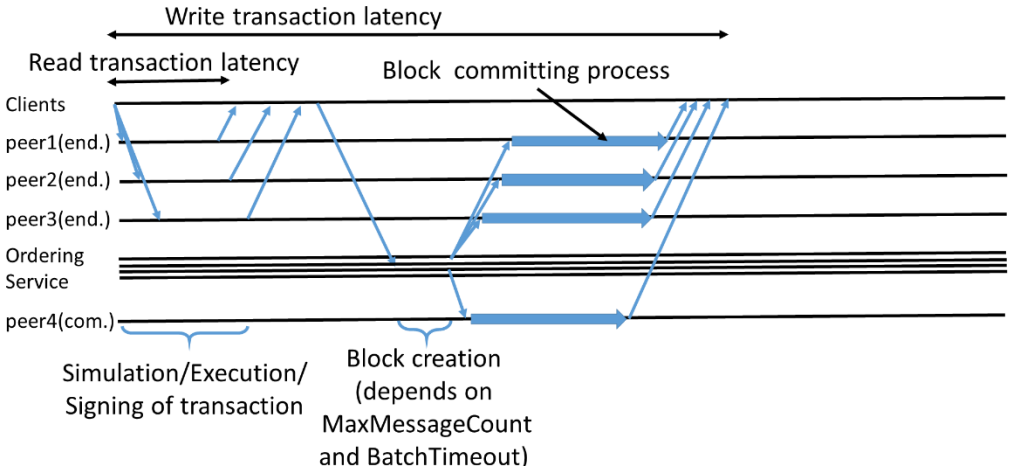

Figure 6 Timing Diagram of Read Transaction and Write Transaction Latency. Peer1(end.), Peer2(end.), and Peer3(end.) are endorsing peers (executing smart contracts and committing new blocks) and Peer4(com.) is a committing peer (only commits new blocks).

Blockchains with probabilistic finality even have to take into account time needed to resolve forks until a consistent network-wide ledger view can be achieved. Also, the consensus algorithm is usually the limiting factor in terms of scalability and reducing latency in blockchain networks. Hyperledger Fabric relies on the crash-fault[1] Raft consensus algorithm that implements a leader-follower model, which guarantees finality at low message overhead. No more than one leader (leader ordering service node) exists at any time in the network. The mechanism as presented in the demonstration [20] starts with all endorsed transactions being forwarded to the leader node that then creates an ordered block of transactions which is propagated and committed by every other node. Heart-beat messages emitted by the leader will prevent follower nodes from promoting themselves into candidate states. In case the heart-beat message ceases, asynchronous voting will determine a new leader among the nodes in candidate states.

Figure 7 and Figure 8-Figure 9 depict the write transaction latency in blockchains with immediate finality and probabilistic finality (for a known total number of nodes), respectively.

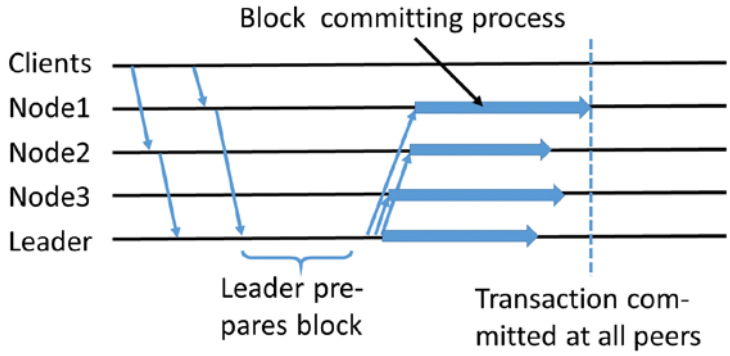

Figure 7 Representation of the write transaction latency in immediate finality blockchains. In case of Hyperledger Fabric, the endorsement process has to be included in the write transaction latency but it is omitted for simplicity in this figure.

[1] It is planned to upgrade the Raft consensus algorithm to a fully byzantine fault tolerant (BFT) ordering service in the future.

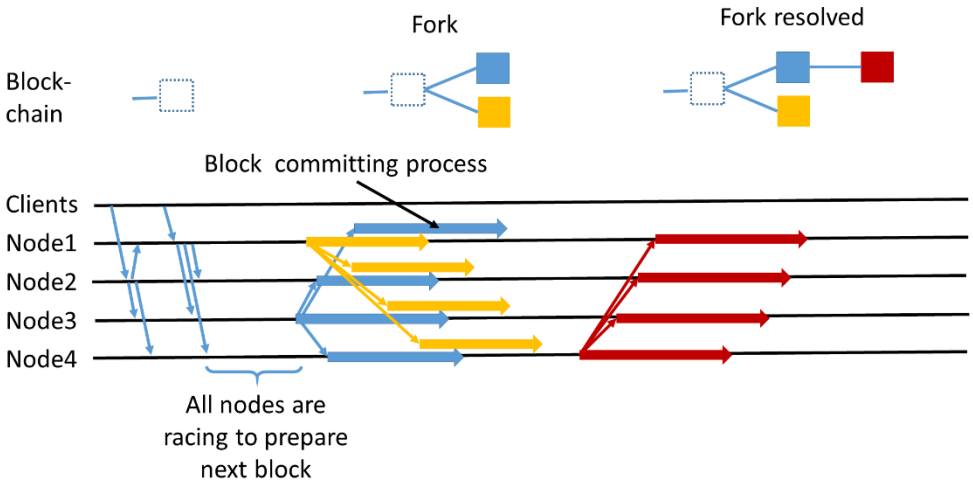

Figure 8 Representation of the write transaction latency in probabilistic finality blockchains.

Node	Write transaction latency at each node
Node 1	20 s
Node 2	10 s
Node 3	60 s
Node 4	15 s
Node 5	10 s

Threshold	Write transaction latency
20%	10 s
40 %	10 s
60 %	12 s
80 %	14 s
100 %	23 s

Figure 9 An example calculation is given to express the write transaction latency as part of thresholds of nodes.

For the immediate finality scenario illustrated in Figure 7, the clients are sending two transactions which are forwarded to the leader node which creates a new block. A network-wide ledger view is crucial because not all nodes will commit the block at the same time. Hyperledger Fabric consensus algorithm guarantees immediate finality. Hence, write transaction latency should be measured at a 100% node threshold to ensure the transaction is fully committed at every node in the network.

Meanwhile, for the probabilistic finality scenario illustrated in Figure 8, the clients are sending two transactions and all nodes will try to find a valid block concurrently. Node 1 and Node 3 both find different valid blocks which causes the blockchain to fork. Only after node 4 finds the next block on top of the block propagated by Node 3, the fork can be resolved. The block propagated by Node 1 is a stale block because it is not part of the longest/best chain anymore. Since the blockchain reaches consistency eventually, the write transaction latency can only be expressed as part of thresholds of nodes (such as the one shown in Figure 9).

3.3.2. Throughput

Read transaction throughput is measured as reads per second. It involves only a single node and is therefore not a stringent metric for blockchain performance. Write transaction throughput is the prime metric for blockchain performance. It is the rate at which valid transactions are committed with respect to a network-wide view. Due to the inherent characteristics of distributed ledgers, it has to be measured across the network rather than at a single node. Equally important, since transactions are either validated or invalidated during the commit process, and only valid transactions are pushing the deterministic state machine forward, write transaction throughput only includes validated transactions rather than the total amount of transactions processed by the network. This can link back to the MVCC issue described previously. For example, a Hyperledger Fabric network that processes *10,000* transactions per second should not be considered as a high performing one if *9,999* of them are invalid due to MVCC conflicts. The current Hyperledger Fabric throughput capabilities have been pushed to *20,000* transactions per second with FastFabric [11] by caching unmarshalled blocks during the

commit phase, hence reducing expensive I/O operations on the ledger. It also applies other improvement techniques such as separating the commitment and endorsement processes to dedicated nodes, replacing the world state database with a lightweight hash table, parallelizing the validation process, and separating the transaction header from its payload to only send the transaction IDs instead of full transactions during the consensus messaging protocol. Among these improvements, the caching of unmarshalled blocks had the greatest effects in increasing the overall throughput.

3.4. Design Methodology

In this subsection, the design methodology to implement end-to-end track-and-trace system using Hyperledger Fabric 2.0 is given, considering the techniques discussed earlier to support high transaction throughput and volume associated with tracking of process data on the manufacturing floor. For simplicity, a typical packaging line in an order fulfillment center is used as example, whereby a system of conveyor belt is set up to transport carton boxes from the carton dispenser to the loading station. A robotic picker located next to the loading station will then request for a tote bin containing the Stock Keeping Unit (SKU) product to be transported to the pick station, and correspondingly the robot will transfer the requested quantity of the SKU product to the carton box. Once all the SKUs in the order have been placed into the carton, the conveyor belt will transport the carton box to the sealing station for final packaging.

The proposed design methodology involves several steps as follows:

1. Understand the process flow and identify the information collection points
2. Categorize the track-and-trace parameters to frequently updated and static ones
3. Define the data structure and their interrelationship
4. Develop the Chaincode smart contract according to the data structure defined in step 3
5. Create the Application Programming Interface (API) to interface the manufacturing floor with the blockchain network, as well as facilitate the presentation of information to the user via Graphical User Interface (GUI)

Detailed description for each of the steps above is presented in the following.

3.4.1. Understand Process Flow

For the packaging line in the order fulfillment center example that is considered, the process begins when a customer places an order to the system. The order specifies the SKU products and the quantity requested. The system will then calculate the number of carton boxes required to pack the SKUs in the order, plan the sequence and placement position of each item within the carton box, and instruct the packaging line to process it accordingly. For each carton box, a dispenser will feed a new empty box to the conveyor belt to be carried to the loading station. Along the conveyor belt, as the empty carton box moves to the loading station, proximity sensors detect the current location of the box. At the same time, the tote bin containing the SKU being ordered will be transported to the pick station through another fast-buffer magnetic levitation system. The fast-buffer system updates the current location of the tote bin.

Once the carton box has arrived at the loading station and the tote bin containing the requested SKU has arrived at the pick station, the robotic arm system will pick the requested quantity of SKUs from the tote bin to the carton box. The next requested SKU in the order list will be similarly processed once the current SKU order is complete.

After the carton box has been filled with all the requested SKUs, the conveyor belt will transport the completed box to the sealing station for final packaging. Again, multiple proximity sensors will track the current location of the box as it travels from the loading station to the sealing station. The process is completed once the finished box is sealed.

3.4.2. Categorize the Track-and-Trace Parameters

Using the process definition, the parameters that are relevant for track-and-trace can be identified. In this simplified example, the parameters to be tracked are mainly the location of

carton boxes and the tote bins as they are transported along the conveyor and fast-buffer system, respectively. The associated time stamp along with the location tags are also relevant to be tracked. In addition, the time stamp when the robotic arm system picks the SKU from the tote bin and places it to the carton box, as well as the time stamp when the completed carton box is sealed are also relevant for track-and-trace.

Out of those parameters to be tracked, the ones that are subject to frequent updates will need to be identified, as they are susceptible to MVCC issue. From the process definition, it is apparent that the location parameters of the carton boxes and tote bins are the most frequently updated, considering the speed of the conveyor belt and the number of proximity sensors installed along the conveyor tracks. As such, special attention need to be given to these parameters when defining the data structure for implementation into the Chaincode smart contract.

3.4.3. Define the Required Data Structure

In defining the data structure, different entities along with their attributes need to be specified as separate object definition. From our example, it is apparent that customer and manufacturer are the two main entities that need to be defined. For manufacturer, the attributes include the unique manufacturing identity (ID), along with other details such as company name, the address location, contact number, email address, and so on. Similarly, the customer's attribute includes customer ID, the details such as customer name, the address location, contact number, email address, and so on. On top of that, the manufacturer and customer will be the actors in the blockchain network that will interact with the smart contract. As such, their roles and identities shall be defined by blockchain organization, and each of them would have to be associated with cryptographic key (a pair of private and public key).

When a customer creates a new order, a new customer order object needs to be created, which will be captured as a blockchain state. The attributes of this customer order object includes a unique order number, the fulfillment status, the customer ID, the list of carton boxes (each containing a list of SKUs included in the order), the manufacturing company ID associated with the SKU ordered, as well as the date and time when the order is placed. When the order is being fulfilled in the manufacturing line, the carton box and the tote bin will be updated, each of which would also need to be defined as blockchain state. For the carton box, the attributes include a unique carton box ID, the associated order number, a list of SKUs along with their quantities ordered, the box size, the fulfillment status, and the current location. For the tote bin, the attributes include the unique tote bin ID, the tote capacity, the SKU that is contained in the tote along with the current quantity, the current location, and also some additional attributes inherited from the supplier-related information. Out of those attributes from the carton box and tote bin, the location attribute needs to be given special treatment as they are subject to frequent update.

Another entity that also need to be defined is the SKU, whose attributes include the barcode number, brand name, product name, and description. For each product, the SKU definition only need to be defined once, and therefore they belong to blockchain constant type.

Figure 10 shows the diagram of the designed data structure and their relationship. The solid lines connecting the different objects shows the relationship between them, with the numbers at each end representing the cardinality of each object in the relationship. The dashed line represents the mapping between parameters from the different object, and those parameters with asterisk symbol represents the key parameter that is unique for every instance. The attributes that are highlighted in light blue color are the frequently updated ones that require special attention. Namely, a separate object data structure will be defined for each one of them in the Chaincode so that each piece can be updated without having to read the state of other attributes in the object, hence avoiding MVCC issue.

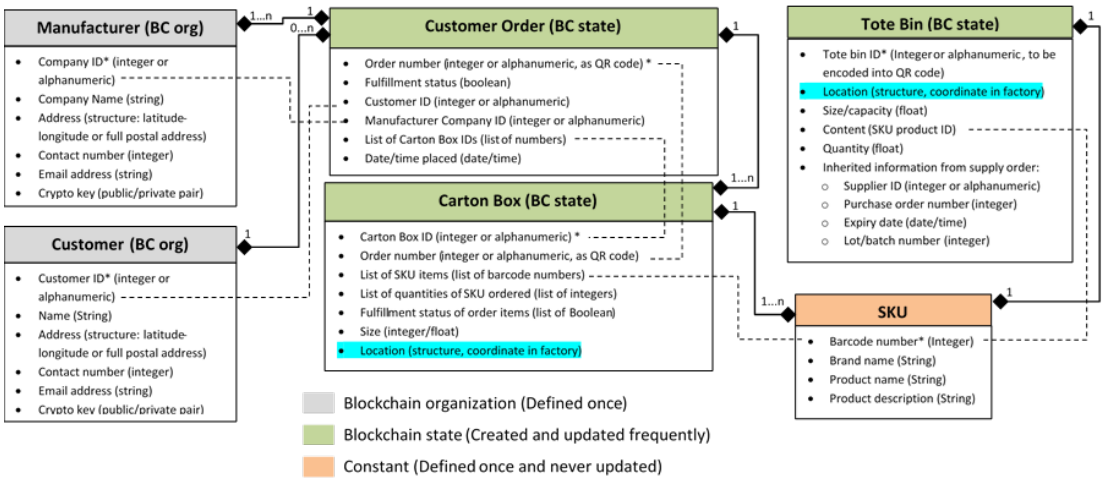

Figure 10 Data Structure of Different Objects and Their Interrelationship

3.4.4. Develop Chaincode Smart Contract

The Chaincode smart contract is a computer program that is used by client application to interface with the world state ledger as well as the blockchain. The main functionality of this Chaincode is to create a new instant, update, as well as query the blockchain state. For the order fulfillment example considered, it is used to create a new customer order instance whenever a customer submits a new order. This order instance, which is kept as a blockchain state in the world state ledger, keeps track of the carton boxes and their SKU contents, as well as the fulfillment status of the order. For the carton boxes, which are also kept as a blockchain state, the Chaincode will create a new instance whenever a new customer order requires a carton box to pack the ordered SKUs. The Chaincode smart contract will also be used to update the locations of the carton box as well as the tote bins containing the ordered SKUs as they move along the conveyor belt.

The network diagram of the Hyperledger Fabric 2.0 implementation of the order fulfillment example is illustrated in Figure 11.

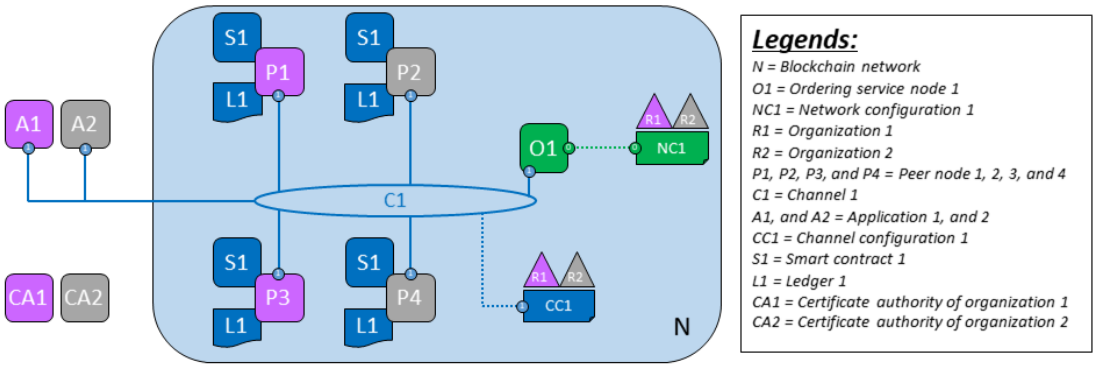

Figure 11 Hyperledger Fabric Network Diagram

In the network diagram shown in Figure 11, there are two organizations that are participating in the consortium. Each organization maintains their own certificate authority, and each one of them deploys two peer nodes onto the network. Each peer node maintains a full copy of the ledger, and installs the Chaincode smart contract. The track and trace application can then interact with the world state ledger and the blockchain using Chaincode smart contract through one channel with a single ordering system configuration.

3.4.5. Develop the Necessary API and Complete System Integration

Once the Chaincode smart contract has been established to interface with the world state ledger and the blockchain, the final step is to create an API to connect it with the application. There are two applications needed for our example, the first one is the blockchain interface to connect the various manufacturing process activities on the factory floor so that the corresponding transactions are stored in the blockchain ledger. The second application is the web application server, which communicates with the web client at the customer browser, and creates a new customer order instance in the ledger each time the customer submits a new order. This web application server will also query the world state ledger to get a real-time status of the order fulfillment and to track different objects on the factory floor such as carton boxes and tote bins. The overall system connectivity architecture is shown in Figure 12.

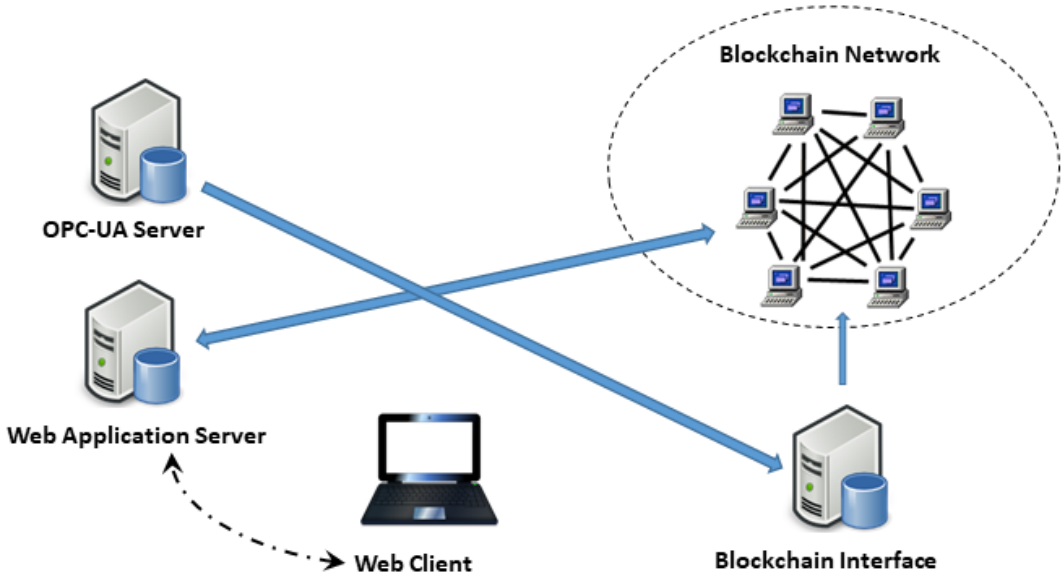

Figure 12 System Connectivity Architecture

To implement the blockchain interface, a connection with the OPC-UA server is established. In most manufacturing floor, the OPC-UA is used as communication protocol across different parts of the machines[2], such as the conveyor belt, robotic arms, packing and sealing stations, and so on. In order to keep track of the process data on the manufacturing floor, the OPC-UA server is a centralized location where those information can be tapped. Therefore, the API developed for the blockchain interface can be implemented using a Python script that subscribes to the relevant tags in the OPC-UA server running at the factory floor. Each time the relevant tag value is updated, which indicates a certain process is happening on the factory floor (e.g. a carton box is passing through a proximity sensor along the conveyor belt), a transaction is submitted to the blockchain network to track the activity and to update the ledger state. The updating of carton box and tote bin locations, the placement of SKUs into the carton box by the robotic arm, and the final sealing of the carton box are updated through this blockchain interface.

For the second application, an Apache web application server is used to host the web form where the customer can place an order. For each order placed, the web server will connect to the blockchain network to submit a transaction to create a new customer order object along with the information such as the list of SKUs ordered. The web application server is also used to host the web page from which the customer can check the real-time status of the order at it is being fulfilled on the factory floor. To get this status information, the web server will periodically send a query request to the blockchain to get the most recent ledger state

[2] For legacy PLC systems that are non-compliant to OPC protocol, additional hardware such as IoT edge device and IoT gateway can be used to tap the relevant signals for track-and-trace and connect to the blockchain interface.

corresponding to the requested order. It is important for these requests to be submitted as query rather than invoke transaction in order to avoid causing MVCC issue as explained earlier.

4. Blockchain Interoperability

Enabling seamless access of information across different blockchain platforms run by the different stakeholders in the supply chain, such as suppliers, manufacturers and logistics companies, through blockchain interoperability is paramount to increase the adoption rate of blockchain technology. The intra and interconnectivity of the manufacturing process with several external and internal business processes will expose it to a pantheon of diverse blockchains and distributed ledger technologies in the future. In general, interoperability functionalities have to provide support for reading ledger data, executing state changes on the ledger, and publishing/subscribing to events emitted by the connected blockchains. There will be no one blockchain ruling them all because applications have to cater for such a diverse spectrum of use cases. Blockchains as part of the manufacturing process have to cater for distributed identity DID (public as well as private DIDs), verifiable credentials, and zero-knowledge proofs leveraging networks, such as the Sovrin network, Hyperledger Indy, Hyperledger Aries, and Hyperledger Ursa, to cater for the internet of things (IoT) and sensor data leveraging networks, such as IOTA, and to cater for track-and-trace solutions leveraging general-purpose blockchains, such as Hyperledger Fabric. For example, an energy, storage, and computationally constrained IoT sensor cannot run a full blockchain node and hence requires a very different blockchain network enabling lightweight clients and sharding than a track-and-trace blockchain node that requires fast ledger updates. The current blockchain landscape is fragmented consisting of isolated blockchain silos, trying to compete against each other rather than promoting interoperability.

4.1. Available blockchain interoperability solutions in the market

Blockchain interoperability has just picked up speed and is expected to mature and develop at a fast pace. The most promising solutions available at time of this write-up are presented in the next sections. The focus will be on solutions that interact with Hyperledger.

4.1.1. Hyperledger Quilt

The earliest blockchain interoperability solutions are focusing on cryptocurrencies, the dominant application area at that time. Quilt implements the Interledger Protocol Version 4 (ILPv4 [22]), a mechanism for secure payments across several non-trusted connectors[3]. The protocol is only for tokens/coins[4] and provides a point-to-point connection between two parties. Prepare packages are sent from the sender to the receiver via connectors, the commitment to pay any of these money packages is linked to the condition that the connector receiving such a prepare package can prove (providing pre-image of the hashlock) that the receiver was paid (cf. Figure 13).

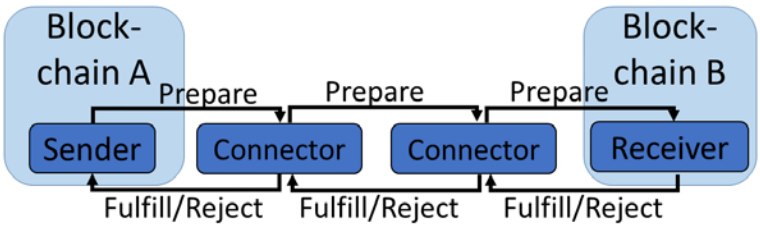

Figure 13 Illustration of Connector Architecture of Interledger Protocol

[3] A non-trusted connector can be used because if the connector receives a new money package (prepare package), it cannot steal as the connector does not know the pre-image of the hashed timelock. Only after the money package reaches the receiver, the receiver reveals the pre-image.

[4] It is only applicable for fungible assets (cf. fungible asset definition in Section 4.2.2).

The receiver acknowledges the received prepare money package by revealing the pre-image of the hashlock. The money transfer can then be settled by sending fulfill packages in backwards direction, revealing the pre-image of the hashlock to any connector along the line. In case the receiver rejects the payment or does not reveal the pre-image before the timelock expires, a reject package is propagated in backward direction and no money package is settled. Payments can be made from one currency (e.g. Bitcoin (BTC)) to another currency (e.g. Ether (ETH)) in a different ledger where the connectors decide on the exchange rate.

The ILP routing can be compared with the well-known IP routing. In IP routing, IP packages (data packages), are routed to different IP addresses. In ILP routing, Interledger protocol packages aka ILP packages (money packages) are routed to different ILP addresses. The basic ILPv4 is meant for high volumes of low-value money packages, as such the transferred value is split into several low-value money packages. The ILP generalizes the concept of Hashed Timelock Contracts (HTLCs) and provides several options called Hashed Timelock Agreements (HTLAs) for the intermediate money transfers between sender/connector, connector/connector, and connector/receiver. There are HTLA types that support larger money transfers and others that can be used if the underlying blockchain technology does not support hashlocks and timelocks natively. Some HTLA types are conditional payment channels (using HTLCs), on-ledger holds/escrows (using HTLCs), simple payment channels, and trustlines. On the one hand, the different HTLA types bear different bilateral risks[5]. On the other hand, long on-ledger HTLC timeouts bear the risk that funds are locked up at a fixed exchange rate that either provides a free American-style call option (i.e. the receiver is the same person as the sender, either keeping its original currency by waiting for the timelock to expire or executing the exchange depending on whether or not the exchange rate changes in their favor) or the connectors have to charge a high fee for offsetting the risk they bear.

4.1.2. Cosmos and Polkadot

Cosmos, and Polkadot connect several blockchains residing in zones, and denoted as parachains through a central Cosmos Hub, and a central Relay Chain in a hub-and-spoke architecture, respectively (cf. Figure 14 - Figure 15). Cosmos focus is on transferring tokens/coins, and Polkadot focus is on transferring data and assets and not just tokens/coins[6]. The ATOM, and DOT are the native tokens in Cosmos, and Polkadot, respectively. Communication is achieved with the inter-blockchain communication protocol (IBC) and the cross-chain message passing (XCMP) protocol in Cosmos and Polkadot, respectively. Blockchains running on a Tendermint consensus algorithm/Cosmos SDK, or a Substrate-based architecture/Substrate SDK, can run natively on Cosmos and Polkadot, whereas more complex pegs and bridges are needed to connect to other blockchains with non-finality consensus algorithm.

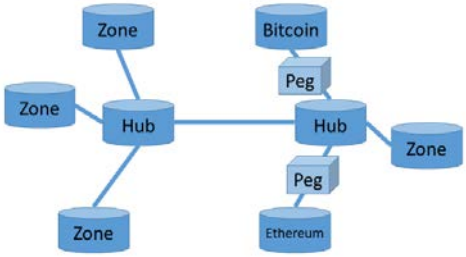

Figure 14 Cosmos Architecture

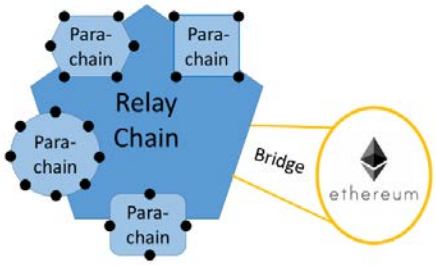

Figure 15 Polkadot Architecture

[5] On-ledger HTLCs do not bear risk of any party not settling outstanding money packages, whereas in trustline, one party bears the risk that the other party is not settling the outstanding balance. Hence, trustlines are only suitable for trusted parties or business relationships that have other mechanisms in place to incentivise honest behavior.

[6] There is active development going on which will expand the capabilities of all the projects mentioned in this chapter. The reader is encouraged to cross-check the current capabilities of the mentioned projects.

Nonetheless, the different design of private and public blockchains with respect to their finality and time needed to confirm a transaction is challenging[7]. Bitcoin considers a block as confirmed after 6 blocks, which can take on average around one hour, resulting in a huge discrepancy in transaction confirmation time between private and public blockchains. The different speeds among private and public blockchains for settling transactions has to be handled by peg-zones and bridges in Cosmos and Polkadot, respectively. An exchange can only be as fast as the slowest blockchain involved in the exchange. Permissioned blockchains are governed by a consortium and transactional privacy is of utmost concern. These matters complicate the interaction between permissioned and permissionless blockchains as these properties are likely to be curtailed when interoperating the two. The native tokens and the associated transactional fees in Cosmos and Polkadot might also be reluctantly adopted by private blockchains.

The zones/pegs in Cosmos are responsible to maintain their own set of validator nodes and individual blockchains are responsible to provide valid proofs about their blockchain states. The tokens/coins sent in Cosmos carry that proof with them. The validity and availability of blockchain states are a community effort in Polkadot and the responsibility is shared among all validator nodes in the whole network. The zones/pegs in Cosmos have a higher sovereignty with increased flexibility in design and operation of the blockchains compared to the parachains/bridges in Polkadot, at a security cost that if not enough validator nodes in a zone/peg are honest, the zone/peg is compromised. Meanwhile, Polkadot security requires that enough validator nodes have to be honest in the whole Polkadot network rather than a single parachain/bridge.

Hyperledger Burrow is based on the Tendermint consensus algorithm and has a gateway to connect to Cosmos. Polkadot has just been released this year (2020) and integration into the Hyperledger greenhouse is yet to be seen.

4.1.3. Hyperledger Cactus

Hyperledger Cactus is designed for permissioned blockchain networks at the moment but development for interoperating with permissionless blockchains are planned in the future. The four blockchain platforms: Hyperledger Fabric, Corda, Quorum and Hyperledger Besu, can interoperate through Hyperleder Cactus at the moment. Ledger specific plug-ins are used to connect different ledgers to Hyperledger Cactus.

There are two inherent problems[8] that have to be solved when connecting different blockchains:

1) How to provide a proof of the networkwide[9] ledger state of a connected blockchain from the outside?

---

[7] Because most public blockchains have a probabilistic finality, an atomic exchange might not be atomic anymore if the transaction was included into a stale block. The longer the private blockchain waits, the less likely it is that the corresponding exchange transaction becomes stale in the public blockchain.

[8] There is an alternative approach for an outside entity A to verify the state of a connected blockchain if this connected blockchain uses Merkle Trees to store its blockchain state. An outside entity A can store the Merkle Tree roots from the headers of committed blocks of a connected blockchain locally to verify any state claims about the connected blockchain. Any untrusted entity can then provide a state of the connected blockchain, such as a specific account balance on the connected blockchain, because the outside entity A can act as a lightweight client and use concepts like simple payment verification (SPV) to verify that the state claim provided by the untrusted entity is valid. SPV can be done without checking the entire blockchain history. Polkadot uses this approach in its Relay Chain and the BTCRelay on the Ethereum blockchain uses this approach as well. Private blockchains do not always keep track of their state through Merkle trees and signatures produced by nodes participating in such private blockchains are rarely understood by outside parties not participating in the network. For that reason, the design principle of Cactus is to rely on the canonical validator node signatures for verifying proofs of blockchain states. Since Cactus should be able to incorporate any type of blockchain in the future, Cactus cannot use the approach based on Merkle Trees.

[9] A networkwide ledger view means that all network nodes have to be considered to derive the state of the blockchain which means that it is not the state of just one single blockchain node.

2) How can other entities verify a given proof of the state of a connected blockchain from the outside?

The Cactus consortium operates for each connected blockchain a group of validator nodes, which as a group provides the proofs of the state of the connected ledger (cf. Figure 16). The group of validator nodes runs a consensus algorithm to agree on the state of the underlying blockchain. Since a proof of the state of the blockchain is produced and signed by several validator nodes[10] with respect to the rules of the consensus algorithm, the state of the underlying blockchain is evaluated networkwide.

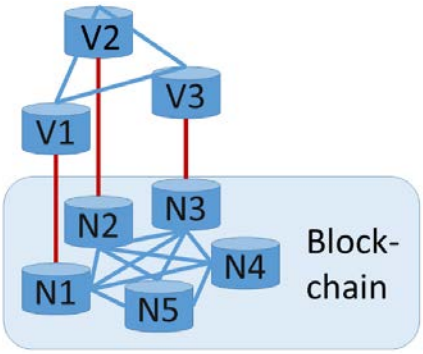

Figure 16 Validator Node in Hyperledger Cactus

The validator nodes are ledger-specific plug-ins, hence depending on the connected blockchain architecture a smart contract on the connected blockchain may or may not be used to enable the ledger-specific functionalities necessary for a validator node to observe the ledger state to finalize a proof. The validator nodes are easier discovered from the outside than the blockchain nodes. Hence, the benefit of operating the Cactus network to enable blockchain interoperability relies on the fact that for any cross-blockchain interaction the same type of validator node signatures can be used. That means, the cross-blockchain interaction can be done canonically with the validator node signatures in Cactus rather than having to deal with many different ledger-specific types of blockchain node signatures.

Outside entities (verifier nodes) can request and register the public keys of the validator nodes of a blockchain network that they want to connect to. Therefore, they can verify the signed proofs of the state of the blockchain since they have the public keys of the validator nodes. This implies that the verifier nodes trust the validator nodes as such they trust the Cactus consortium operating the validator nodes.

4.2. Terminology of blockchain interoperability

This section acts as a building block to describe the different flavors of blockchain interoperability. Concrete use cases will be built on these simple foundation blocks leveraging a mix of them simultaneously and even expanding to several blockchains interacting concurrently. To qualify and quantify blockchain interoperability, we will derive these building blocks by leveraging without loss of generality two blockchains A and B. Consistent terminology is derived to provide guidelines on describing blockchain interoperability. As a person who is considering implementing a blockchain interoperability solution for a concrete business use case, this will help to identify relevant example use cases and suitable blockchain technologies that are aligned with the given concrete business use case.

[10] The validator nodes in Hyperledger Cactus have similarities with trusted third party intermediaries. The terminology trusted third party intermediaries, federation/notary schemes are used when a blockchain can retrieve the state of another blockchain through these intermediaries.

4.2.1. Homogeneous and heterogeneous blockchain interoperability across deployments and blockchain platforms

A full-stack blockchain application has several layers, notably the application, blockchain, and deployment layer (cf. Figure 17).

Figure 17 Layer stack of a full-stack blockchain application.

Homogeneous and heterogeneous blockchain interoperability can concern the blockchain platform and deployment platform (cf. Figure 18). With respect to the blockchain platform, two blockchains are homogeneous if they are similar in design and functionality. The transition between homogeneous and heterogeneous blockchains is continuous rather than discrete. Such as blockchains running on a Tendermint consensus algorithm or Substrate-based architecture, running natively on Cosmos and Polkadot enabling interoperability as they share common features, respectively. Ethereum, Quorum, and Hyperledger Besu also share common features based on the Ethereum virtual machine (EVM) and the solidity smart contract language making them a starting point of interoperability adventures.

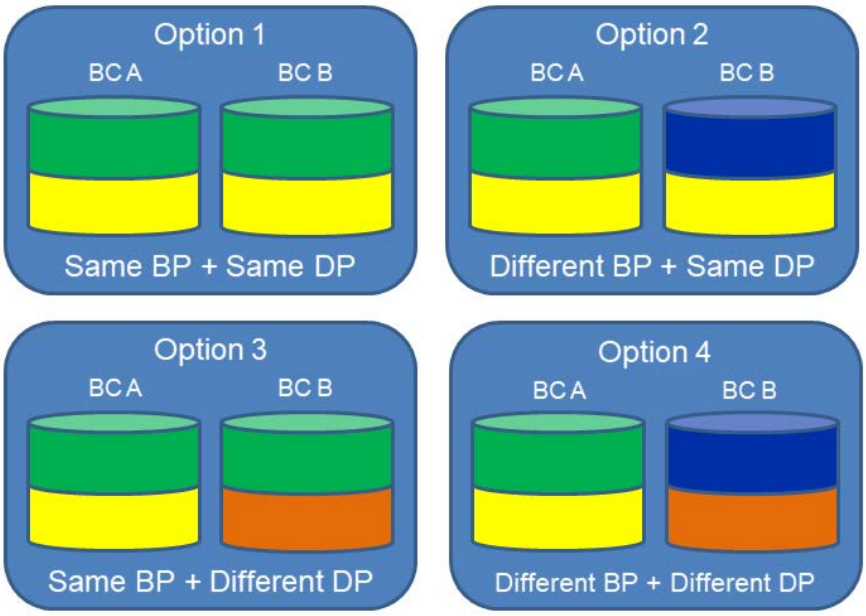

Figure 18 Homogeneous and heterogeneous blockchain interoperability between blockchain A (BC A) and blockchain B (BC B) across blockchain platform (BP) and deployment platform (DP).

4.2.2. Ledger object types

To describe typical interaction pattern between different blockchains, the types of objects stored on a ledger have to be distinguished. The following three types of objects stored on a ledger are differentiated as follows:

- FA: Fungible asset (value token/coin) – cannot be duplicated on different ledgers
- NFA: Non-fungible asset – cannot be duplicated on different ledgers[11]
- D: Data – can be duplicated on different ledgers

Difference between a fungible (FA) and a non-fungible asset (NFA)

A fungible asset is an asset that can be used interchangeably with another asset of the same type, like a currency. For example, a 1 USD bill can be swapped for any other 1 USD bill. Cryptocurrencies, such as ETH (Ether) and BTC (Bitcoin), are FAs. A non-fungible asset is an asset that cannot be swapped as it is unique and has specific properties. For example, a car is a non-fungible asset as it has unique properties, such as color and price. CryptoKitties or a product that is tracked on the blockchain in a supply chain are NFAs. There are two different standards for fungible and non-fungible assets on the Ethereum network (ERC-20 Fungible Token Standard [6] and ERC-721 Non-Fungible Token Standard [7].

Difference between an asset (FA or NFA) and data (D)

Unicity applies to FAs and NFAs meaning it guarantees that only one valid representation of a given asset exists in the system. It prevents double spending of the same token/coin in different blockchains. The same data package can have several representations on different ledgers while an asset (FA or NFA) can have only one representation active at any time, i.e. an asset exists only on one blockchain while it is locked/burned on all other blockchains. If fundamental disagreement persist in the community about the purpose or operational upgrades of a blockchain, a hard fork can split a blockchain creating two representations of the same asset to coexist. For example, Bitcoin split into Bitcoin and Bitcoin Cash in 2017. Forks are not addressing blockchain interoperability, so the definition of unicity applies in a blockchain interoperability context. A data package that was once created as a copy of another data package might divert from its original one over time because different ledgers might execute different state changes on this data package.

4.2.3. Blockchain interoperability types

Blockchain interoperability implementations can be classified into following types:

- Ledger transfer (cf. Figure 19 - Figure 20)
  An asset gets locked/burned on one blockchain and then a representation of the same asset gets released in the other blockchain[12]. There are never two representations of the same asset alive at any time. Data is an exception since the same data can be transferred to several blockchains. There are one-way or two-way ledger transfers depending on whether assets can be transferred only in one direction from a source blockchain to a destination blockchain or assets can be transferred in and out of both blockchains with no designated source blockchain and destination blockchain.
- Atomic swap (cf. Figure 21)
  A write transaction is performed in blockchain A concurrently with another write transaction in blockchain B. There is no asset/data/coin leaving any blockchain environment. The two blockchain environments are isolated, but due to the blockchain interoperability implementation, both transactions are committed atomically. That means either both transactions are committed successfully or none of the transactions are committed at all.

[11] There might be use cases where it is desired to duplicate a NFA on different ledgers. Nonetheless, we stick to the terminology that a NFA cannot be duplicated on a different ledger in this chapter, because a NFA can be represented as a data packages on different ledgers in such cases. Data is a superset of NFAs.

[12] In case of data, the data can be copied from blockchain A to blockchain B. It is optional if the data is removed from the world state of blockchain A after copying.

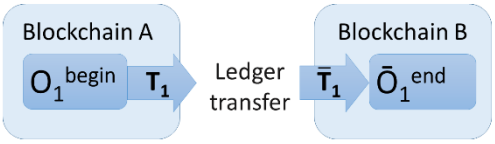

Figure 19 One-way ledger transfer

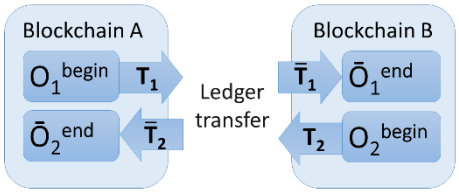

Figure 20 Two-way ledger transfer

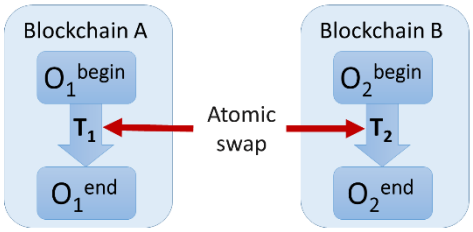

Figure 21 Atomic swap

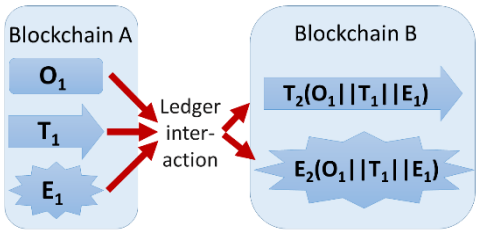

Figure 22 One-way ledger interaction

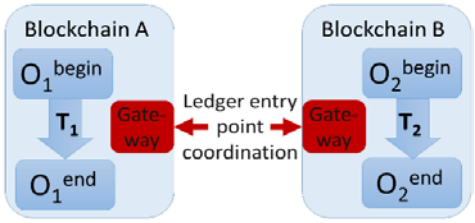

Figure 23 Ledger entry point coordination

O_1^{begin} = State of object O_1 at the beginning of transaction

O_1^{end} = State of object O_1 at the end of transaction

$\underline{O}_1^{end}$ = Representation of object O_1 at the end of transaction in another blockchain

T_1 = Transaction 1

$\underline{T}_1$ = Representation of transaction 1 (T_1) in another blockchain

O_1 = Object 1

E_1 = Event 1

O_2^{begin} = State of object O_2 at the beginning of transaction

O_2^{end} = State of object O_2 at the end of transaction

$\underline{O}_2^{end}$ = Representation of object O_2 at the end of transaction in another blockchain

T_2 = Transaction 2

$\underline{T}_2$ = Representation of transaction 2 (T_2) in another blockchain

$T_2(O_1||T_1||E_1)$ = Transaction 2 depends on O_1 or T_1 or E_1

$E_2(O_1||T_1||E_1)$ = Event 2 depends on O_1 or T_1 or E_1

- Ledger interaction[13] (cf. Figure 22)
  An action[14] happening on blockchain A is causing an action on blockchain B. The state of blockchain A causes state changes on blockchain B. There are one-way or two-way ledger interactions depending if only the state of one of the blockchains can affect the state on the other blockchain or both blockchain states can affect state changes on the corresponding other blockchain.
- Ledger entry point coordination (cf. Figure 23)
  This blockchain interoperability type concerns end-user wallet authentication or authorization enabling read and write operations to independent ledgers from one single entry point. Any read or write transaction submitted by the client is forwarded

[13] The process in blockchain A and the process in blockchain B can be seen to happen consecutively in ledger interactions, as opposed to concurrently in atomic swaps.

[14] An action can be either a read/write transaction that is performed on blockchain A or an event that is emitted by blockchain A.

to the corresponding blockchain and then committed/executed as if the blockchain would be operate on its own.

The ledger transfer has a high degree of interference between the blockchains since the livelihood of a blockchain can be reduced in case too many assets are locked/burned in a connected blockchain. The ledger interaction has a high degree of interference between the blockchains since the state of one blockchain can affect the state of the other blockchain. Atomic swaps have less degree of interference between the blockchains since all assets/data stay in their respective blockchain environment. The ledger entry point coordination has no degree of interference between the blockchains since all transactions are forwarded and executed in the corresponding blockchain as if the blockchain would be operated in isolation.

4.2.4. Burning/Locking of assets

To guarantee unicity, an asset (NFA or FA) has to be burned or locked before being released in another blockchain. Locked assets can be unlocked in case the asset is retransferred back to its original blockchain, whereas the burning of assets is an irreversible process. Burning of assets usually applies more to tokens/coins (FAs) and can be seen as a donation to the community since the overall value of the currency increases. Standard cryptocurrency exchange platforms rely on atomic swaps and do not burn FAs. Cryptocurrency exchange platforms relying on ledger transfers are either locking or burning the FAs, for example, the migration of a cryptocurrency to another blockchain or the upgrade of ETH from Ethereum 1.0 to Ethereum 2.0.

Burning of assets can be implemented as follows:

- Assets are sent to the address of the coinbase/generation transaction[15] in the genesis block. A coinbase/generation transaction is in every block of blockchains that rely on mining. It is the address where the reward for mining the block is sent to. Hence, this will burn the tokens/coins in the address of the miner that mined the genesis block. In many blockchain platforms, it is proven that nobody has the private key of this special address. For example, Ethereum uses the address "0x00...0" in the genesis block. This accidentally caused ETH to be sent to this address because in case the user did not change the "address_to" field in an ETH transfer from the default address of "0x00...0", the address was sometimes used causing the ETH and other ERC20 tokens being unintentionally burned. The amount of ETH already burned in this address can be viewed in [5]. Another example, the 50 BTCs (Bitcoin) reward subsidy in the Bitcoin genesis block to the miner can never be spent, but people started to send BTCs to the genesis block as a donation to Satoshi. These donations are spendable if Satoshi possesses the private key which many people doubt.
- Tokens/Coins are subtracted from the user account as well as optionally from the total token/coin supply value.

4.3. Blockchain interoperability use cases

Table 2 provides use case examples for the different ledger object types and blockchain interoperability operation types.

Table 2 Fundamental building blocks of blockchain interoperability.

Object type of Blockchain A	Object type of Blockchain B	Ledger Transfer		Atomic swap	Ledger Interaction		Ledger entry point coordination
		1-way	2-way		1-way	2-way	
D	D	Ex. 1	-	-		-	Ex. 5
FA	FA	Ex. 2	Ex. 3	Ex. 3	Ex. 4	-	Ex. 5
NFA	NFA	-	-	-	-	-	Ex. 5

[15] Alternatively, any address from that asset which cannot be recovered anymore can be used. A verifiable proof for the irreversible characteristic of that address should be given.

FA	D	-	-	-	-	-	Ex. 5
D	FA	-			-		
NFA	D	-	-	-	-	-	Ex. 5
D	NFA	-			-		
NFA	FA	-	-	Ex. 6	-	-	Ex. 5
FA	NFA	-			-		

In Table 2, FA, NFA, and D stand for a fungible asset, a non-fungible asset, and data, respectively. The description of each of the examples are given in the following.

- Example 1: One-way ledger transfer of data facilitates the migration of blockchain data from one blockchain to another. Existing distributed ledger technology can then migrate to the most advanced, and best blockchain platform available in the market at any time. A fluent data migration to any possible blockchain provider prevents vendor lock-in. It should be highlighted that transferring the data objects into another ledger is considerably easier compared with migrating the transactional history of the blockchain. The latter needs some mechanism to replay the transactional history of the data objects in the destination blockchain.
- Example 2: One-way ledger transfer of tokens/coins (FAs) is proposed by Ethereum 2.0. The upgrade from Ethereum 1.0 to Ethereum 2.0 will take place in several phases from 2020-2022. Old ETH tokens can then be deposited into a smart contract to transfer them by one-way, non-reversible transactions into the Beacon chain. The Beacon chain is the coordination layer in Ethereum 2.0 trying to solve the scalability problem by enabling sharding. In addition, an upgrade to a PoS consensus algorithm will be performed.
- Example 3: Fiat or cryptocurrency exchange platforms backed by blockchains can be implemented as a two-way ledger transfer of FAs or as an atomic swap of FAs. Therefore, blockchain interoperability is an inherent part of cryptocurrency trading. For example, new BTC or ETH can only be generated by mining a block[16]. Hence, a Bitcoin or Ethereum exchange platform has to rely on atomic swaps rather than two-way ledger transfers because it is not possible to create ETH or BTC on the fly. Another example, the terminology sidechains are often used when assets are moved to another chain to increase the scalability and throughput of the main blockchain by trading the assets on the sidechain [30].
- Example 4: Ledger interactions are proposed in various flavors. For example, cross-chain oracles are smart contracts that read the state of another blockchain before acting on it. Alternatively, a smart contract can wait until an event happens on another blockchain before acting on it. Asset encumbrance has also been proposed by locking up assets on blockchain A with unlocking conditions depending on actions happening in blockchain B. Another example, the BTCRelay[17] is a smart contract on the Ethereum mainnet that can verify Bitcoin transactions by storing the Merkle tree roots of blocks from the Bitcoin network locally in the Ethereum mainnet. The BTCRelay is a light client on the Bitcoin network enabling to use BTC for paying in Ethereum DAPPs. There is no ETHRelay on the Bitcoin network because Bitcoin lacks the functionality of Ethereum due to the scripting language in Bitcoin not being Turing complete and the missing support for smart contracts.
- Example 5: Ledger entry point coordination requires an interoperable end-user wallet authentication/authorization process enabling read and write operations to independent

[16] Bitcoin total supply only consists of block rewards. Ethereum total supply consists of block rewards of mining regular blocks as well as mining uncle blocks and roughly 72,000,000 ETH generated in the genesis block that were distributed to participants of the Ethereum presale [8].

[17] The terminology relay is used when a chain can retrieve the state of another blockchain through read, write, or event listening operations directly rather than relying on third party intermediaries. The terminology is also used in the central Relay Chain in Polkadot. Relays are considered more complicated to implement than trusted third party intermediaries/federation/notary schemes.

ledgers from one single entry point. The user might even access the different blockchains with the same cryptographical credentials. For example, an end user is interested in the provenance and authenticity of his/her purchased product. The product left independent traces in blockchains run by the producer, the manufacturer, the logistics company, and the retailer, which can be augmented for a seamless and holistic user experience containing the complete supply chain history of the product from one single entry point using the cryptographical credentials of the product.

- Example 6: An atomic swap between a blockchain optimized to manage NFAs and a blockchain optimized to manage FAs is shown in this example. Alice wants to rent a car from Bob. Alice and Bob are both participating in a car ownership blockchain (NFA) and a cryptocurrency blockchain (FA). The atomic swap ensures that the ownership of the car is conveyed to Alice in the NFA blockchain if and only if Bob is paid in the FA blockchain. Hashed Timelock Contracts (HTLC) can be used to implement atomic swaps if the underlying ledger supports hashlocks and timelocks. Escrow contracts can also be used. Traditionally, an independent third party called an escrow agent acquires both assets and checks if all terms and conditions of the exchange have been fulfilled by both parties before releasing the assets to the new owner. This is modeled by deploying escrow smart contracts on both of the blockchains since assets are not meant to leave the blockchain environment in this example.
  Several steps process is needed to lock both assets in escrow contracts, to verify that the corresponding blockchain has correctly locked the asset and this state change is reflected in a network-wide view and not just on one node before the atomic swap protocol can unlock the assets to the new owners [13]. It should be kept in mind that any point in the multi-step atomic swap protocol can fail. If this happens, the blockchain states have to be reverted to its original states. The atomic swap protocol by itself can be implemented as a smart contract on an independent blockchain for bookkeeping reasons so that every executed step in the protocol is immutably stored on a blockchain. In case of failure, such a setup can also immutably store at which step the protocol failed and the reasons, providing a transparent view of the atomic swap process to any participant.
  In general, many blockchain interoperability protocols (not just atomic swaps) can include intermediate blockchains. The Cosmos Hub, and the Relay Chain are basically blockchains with their own consensus algorithm and block validation process in Cosmos, and Polkadot, respectively. There is also an independent blockchain in each peg-zone, and bridge in Cosmos, and Polkadot, respectively. Hyperledger Cactus (Fujitsu codebase) also proposes a Hyperledger Fabric blockchain instance called ConnectionChain as intermediate blockchain which records transactions along the multi-step interoperability protocol. Executing an HTLC involves two parties, the sender and the receiver. Hence, each HTLC is completely isolated from other HTLCs so that an intermediate blockchain is not necessarily required.
  There are several caveats which concerns an escrow smart contract transferring tokens/coins to another address[18]. On the one hand, there is no guarantee that the address that the tokens/coins are transferred to can be accessed by an entity. If no one is in control of the private key of that address, or if that address is a contract account[19] in Ethereum which by chance was not designed to handle such tokens/coins, the transferred tokens/coins are irreversibly burned. On the other hand, if that address is a contract account in Ethereum, it could maliciously or accidentally have a fallback()/receive() function implemented that runs out of gas or throws an exception causing ETH transfers to be reverted. ETH might then be irreversibly burned in the escrow contract address. In general, the pull-over-push pattern is recommended to shift the risk of transferring ETH to the user by relying on the

[18] In general, these caveats apply to any process transferring tokens/coins to another address not just escrow contracts.
[19] There are externally owned accounts (EOAs) and contract accounts (CAs) on the Ethereum mainnet.

user to proactively execute a withdraw function [9]. Nonetheless, this pattern causes delays if the user is not acting in a timely manner or not acting at all.

5. Summary

In this chapter, a blockchain implementation of secure and transparent track-and-trace in manufacturing using Hyperledger Fabric 2.0 has been described. Starting with a brief introduction of blockchain and the various types of blockchains available, we then discussed various use cases in manufacturing. With the increasing trend to include process data on the factory floor as part of the track-and-trace information, especially for hyper-personalized production, we have discussed in detail the extra consideration to use blockchain, including the transaction throughput, latency, and concurrency control. Using Hyperledger Fabric 2.0, the design methodology for an exemplary case of order fulfillment center is explained. Finally, to expand the blockchain network for creating a larger ecosystem, the ability to integrate blockchains running different protocol is of paramount importance. This chapter ends with a discussion on blockchain interoperability, which describes the current available solution, the different interoperability types, as well as some of the use cases that will help in selecting the most suitable interoperability solution to use.

References

[1]. Amazon AWS Managed Blockchain for Supply Chain. https://aws.amazon.com/managed-blockchain/

[2]. Capgemini Research Institute: “Does Blockchain Hold the Key to a New Age of Supply Chain Transparency and Trust?,” (2018) Available online. https://www.capgemini.com/wp-content/uploads/2018/10/Digital-Blockchain-in-Supply-Chain-Report.pdf (Cited 22 June 2020)

[3]. Creative Common Attribution 4.0 International License. https://creativecommons.org/licenses/by/4.0/

[4]. Economic Development Board, Singapore: “The Singapore Smart Industry Readiness Index,” (2017) Available online. https://www.edb.gov.sg/content/dam/edbsite/news-and-resources/news/advanced-manufacturing-release/the-sg-smart-industry-readiness-index-whitepaper.pdf (Cited 22 June 2020)

[5]. Ethereum: “Address 0x0000000000000000000000000000000000000000,” Available online. https://etherscan.io/address/0x0000000000000000000000000000000000000000 (Cited 16 June 2020)

[6]. Ethereum: “ERC20 Standard,” (2020) Available online. https://eips.ethereum.org/EIPS/eip-20 (Cited 16 June 2020)

[7]. Ethereum: “ERC721 Standard,” (2020) Available online. https://eips.ethereum.org/EIPS/eip-721 (Cited 16 June 2020)

[8]. Ethereum: “Ether Supply,” Available online. https://etherscan.io/stat/supply (Cited 16 June 2020)

[9]. Ethereum: “Pull-over-push-pattern,” Available online. https://solidity.readthedocs.io/en/latest/common-patterns.html#withdrawal-from-contracts (Cited 16 June 2020)

[10]. GartnerResearch: "Forecast: Blockchain Business Value, Worldwide, 2017-2030,” (2017) Available online. https://www.gartner.com/en/documents/3627117 (Cited 22 June 2020)

[11]. Gorenflo, C, Lee, S., Golab, L., and Keshav, S.: FastFabric: “Scaling Hyperledger Fabric to 20,000 Transactions per Second,” in IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 455–463, 2019

[12]. Gorenflo, C, Golab, L, Keshav, S: “XOX Fabric: A hybrid approach to transaction execution,” (June 2019) Available online. https://arxiv.org/abs/1906.11229 (Cited 22 June 2020)

[13]. Hyperledger Cactus: “Whitepaper Version 0.1 (Early Draft),” Available online. https://github.com/hyperledger/cactus/blob/master/whitepaper/whitepaper.md (Cited 16 June 2020)

[14]. Hyperledger Fabric: Distributed Ledger Software. https://www.hyperledger.org/use/fabric

[15]. Hyperledger Fabric 2.0 documentation: “Peers and Orderers, Phase 1: Proposal,” Available online. https://hyperledger-fabric.readthedocs.io/en/release-2.0/peers/peers.html (Cited 22 June 2020)

[16]. Hyperledger Fabric 2.0 documentation: “The Ordering Service: Phase two: Ordering and packaging transactions into blocks,” Available online. https://hyperledger-fabric.readthedocs.io/en/release-2.0/orderer/ordering_service.html (Cited 22 June 2020)

[17]. Hyperledger Fabric 2.0 documentation: “Phase three: Validation and commit,” Available online. https://hyperledger-fabric.readthedocs.io/en/release-2.0/orderer/ordering_service.html (Cited 22 June 2020)

[18]. Hyperledger: “Advancing Business Blockchain Adoption through Global Open Source Collaboration,” Available online. https://www.hyperledger.org

[19]. Hyperledger: “HL Whitepaper Metrics,” (2018) Available online. https://www.hyperledger.org/wp-content/uploads/2018/10/HL_Whitepaper_Metrics_PDF_V1.01.pdf (Cited 16 June 2020)
[20]. Hyperledger: “Raft Consensus Algorithm Demo,” Available online. http://thesecretlivesofdata.com/raft/ (Cited 16 June 2020)
[21]. IBM TradeLens. https://www.tradelens.com
[22]. Interledger: “Interledger,” Available online. https://interledger.org/ (Cited 16 June 2020)
[23]. Karame, G O, Androulaki, E, Capkun, S: “Double-Spending Fast Payment in Bitcoin,” in ACM Conference on Computer and Communications Security (CCS), In Proceedings, 2012
[24]. Nakamoto, S: “Bitcoin: A Peer-to-Peer Electronic Cash System,” (2008) Available online. https://bitcoin.org/bitcoin.pdf (Cited 22 June 2020)
[25]. NIST: “Secure Hash Standard (SHS), FIPS PUB 180-4,” (August 2015) Available online. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf (Cited 22 June 2020)
[26]. Oracle intelligent track and trace. https://www.oracle.com/applications/blockchain/track-and-trace.html
[27]. PWC: “How Can Blockchain Power Industrial Manufacturing?,” (2018) Available online. https://www.pwc.com/us/en/industries/industrial-products/library/blockchain-industrial-manufacturing.html (Cited 22 June 2020)
[28]. Project Provenance. https://www.provenance.org/
[29]. Showcase of end-to-end track-and-trace Blockchain System on the manufacturing floor. Available online. https://youtu.be/rQC_lmt7ZSo (Cited 9 September 2020)
[30]. Sidechain: “Sidechain,” Available online. https://docs.plasma.group/en/latest/src/plasma/sidechains.html (Cited 16 June 2020)