ECONSTOR
Make Your Publications Visible.

A Service of

ZBW Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Ahmed, Mohamed; Taconet, Chantal; Ould, Mohamed; Chabridon, Sophie; Bouzeghoub, Amel

Conference Paper

Enhancing B2B supply chain traceability using smart contracts and IoT

Provided in Cooperation with:
Hamburg University of Technology (TUHH), Institute of Business Logistics and General Management

Suggested Citation: Ahmed, Mohamed; Taconet, Chantal; Ould, Mohamed; Chabridon, Sophie; Bouzeghoub, Amel (2020) : Enhancing B2B supply chain traceability using smart contracts and IoT, In: Kersten, Wolfgang Blecker, Thorsten Ringle, Christian M. (Ed.): Data Science and Innovation in Supply Chain Management: How Data Transforms the Value Chain. Proceedings of the Hamburg International Conference of Logistics (HICL), Vol. 29, ISBN 978-3-7531-2346-2, epubli GmbH, Berlin, pp. 559-589,
https://doi.org/10.15480/882.3110

This Version is available at:
https://hdl.handle.net/10419/228933

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.

https://creativecommons.org/licenses/by-sa/4.0/

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

Mohamed Ahmed, Chantal Taconet, Mohamed Ould, Sophie Chabridon, and Amel Bouzeghoub

Enhancing B2B supply chain traceability using smart contracts and IoT

CC-BY-SA4.0

Published in: Data science and innovation in supply chain management
Wolfgang Kersten, Thorsten Blecker and Christian M. Ringle (Eds.)
ISBN: 978-3-753123-46-2 , September 2020, epubli

Enhancing B2B supply chain traceability using smart contracts and IoT

Mohamed Ahmed [1], Chantal Taconet [2], Mohamed Ould [3], Sophie Chabridon [2], and Amel Bouzeghoub [2]

1 – ALIS – Institut Polytechnique de Paris

2 – Institut Polytechnique de Paris

3 – ALIS

Purpose: The management of B2B supply chains that involve many stakeholders requires traceability processes. Those processes need to be secured. Furthermore, quality traceability data has to be transparently shared among the stakeholders. In order to improve the traceability process, we propose to enhance blockchain based traceability architectures with the capability to detect and record well-qualified incidents.

Methodology: To achieve this goal, we propose a generic smart contract for B2B traceability data management, including transport constraints such as temperature, delay and allowing automatic incident detection and recording. We propose an architecture where data are collected by connected objects and verified and qualified before being sent to the smart contract. This proposition has been validated with medical equipment transport use cases.

Findings: As results, the proposed generic template contract can be used in various traceability use cases, well qualified incidents are transparently shared among stakeholders, and secured, qualified and verified traceability data can be used in case of claims or litigation and can facilitate also the automation of invoicing process.

Originality: The originality of this work arises from the automated B2B traceability management system based on qualified IoT data, contractual milestones and process coded in a generic smart contract, and also from the fact that traceability related data and incidents are verified and qualified in order to increase the integrated data quality.

First received: 11. Mar 2020 Revised: 2. Jun 2020 Accepted: 12 Aug 2020

1 Introduction

The collection and management of traceability data and the agreement on its management rules are today major challenges for the B2B supply chain. According to Van Dorp (2002), ISO defines the traceability as: "the ability to trace the history, application or location of an entity by means of recorded identification". Consequently, the traceability requires to store and share securely the data and the process related to an entity among all its stakeholders.

The supply chain with all its stakeholders was among the firsts use cases of the blockchain technology outside the cryptocurrency's domain, as this technology responds to the issues of securing data sharing between stakeholders. The usage of the blockchain in the supply chain helps to meet key supply chain management objectives such as cost, quality, speed, dependability, risk reduction, sustainability and flexibility as stated by Kshetri (2018).

The emergence of the blockchain has facilitated the development of smart contracts. According to Szabo (1997), smart contract designates the hard coding of all contract clauses in a hardware or software in order to be executed automatically in a secured and distributed environment. In the end of 2013, Ethereum comes with an integrated framework for smart contract development Buterin (2014). Since, it has become a standard in blockchain implementations to integrate the support of smart contracts.

Traceability solutions are proposed in many articles using the blockchain. However, many of these proposals stop at the first level of use of the blockchain, using it just as a storage medium, and thus, they do not take ad-

vantage from the automation possibilities offered by smart contracts to implement distributed, secure and reliable process of contractual milestones and incidents management.

Additionally, many solutions of the state of the art propose to use premissionless blockchains in a private network (as in Lin et al. (2019), Westerkamp et al. (2018) and Hasan et al. (2019)), but these blockchains are not adapted to the B2B supply chain context. All stakeholders are well identified in the B2B supply chain and there is a certain level of trust established by contracts between them. The needs, in this specific context, are more to share securely and reliably data and processing rules among those stakeholders and manage different access levels to the shared data and process.

The combination of the IoT with blockchain traceability-based systems provides those systems with auto collected and real time field data. In the state of the art, some works propose to set up blockchains at the level of the IoT network as in Hinckeldeyn & Jochen (2018), but the blockchain with its resources needs is not adapted to the IoT network level which is resource limited. Some other works propose to integrate in the blockchain raw data captured and transmitted by the various connected objects of the supply chain without any IoT data qualification process as in Hasan et al. (2019). Those above cited two propositions are not adequate, and there is a need to provide the blockchain traceability-based systems with only relevant IoT data without outlier or redundant data.

Automation possibilities of smart contracts and the IoT auto data collection capabilities open new opportunities for enhancing the traceability with secured, transparent, reliable and shared rules and data. The traceability enhancing brings questions about incidents management. The incidents are elements of the daily life in the supply chain and the lack of secured and

transparent process for their management affects seriously the data quality of traceability systems.

The contributions of this work to enhance B2B supply chain traceability are the proposition of a generic smart contract to handle contractual milestones, IoT data, incidents creation and qualification rules. The genericity of the proposed smart contract means that it can be deployed across the majority of B2B supply chain contexts without needs for new development efforts. We propose also to use IoT data qualification servers to automatically qualify the IoT data, collected from connected objects or provided by the traceability process stakeholders, before its integration in the smart contract, in order to reduce the amount of data to be stored in the underlying blockchain. A stakeholder's confirmation process is also proposed in the smart contract for some elements such as incidents to confirm the stakeholder's involvement in those elements and their agreement on their related data.

The rest of the paper is organized as follows: in Section 2 we study the works related to supply chain traceability using smart contracts and IoT. In Section 3 we present the architecture of the proposed B2B supply chain traceability solution. Section 4 presents an evaluation of the proposed solution. Finally, we conclude in Section 5 and present some future works.

2 Related works

The usage of smart contract combined with IoT in the supply chain is a recent research trend, and several solutions have been proposed using those technologies. As state Rejeb et al. (2019), the combination of smart contracts and IoT could enhance the transparency, the confidence, the efficiency and the traceability of the supply chain. In this paper, we focus on the traceability enhancing and we analyze literature concerning the works that use smart contracts to tackle the supply chain traceability problem.

We studied the related works according to five traceability enhancing requirements. Firstly, the usage of the IoT technology (R1) which is essential to accelerate and automate the field data collection and incidents detection. Secondly, the genericity (R2) of the proposed smart contract in term of logic and manipulated entities, such as milestones, transport conditions and incidents. This means that the proposed smart contract could be applied in another supply chain traceability context without needs for further development efforts. Thirdly, the usage of contractual milestones (R3) between stakeholders, which are essentials for B2B traceability. Fourthly, the integration of an IoT data qualification module (R4), which is necessary for submitting into the smart contract only relevant IoT data, and consequently alleviate the underlying blockchain data amount. Finally, the support of traceability incidents management (R5), for example the auto detection and qualification of non-compliance with contractual milestones dates or transport conditions, which are essentials for B2B supply chain traceability systems.

Some of the related works try to resolve the supply chain traceability issues using only smart contracts without IoT as in Lin et al. (2019), Westerkampet

al. (2018), Cui et al. (2019), Yong et al. (2020), Salah et al. (2019) and Helo & Hao (2019). Yong et al. (2020) proposed a traceability solution for the vaccine supply chain, allowing to detect vaccine expiration which is an incident related to the vaccine lifecycle management rather than the supply chain process. Chang et al. (2019) tried to reengineer theoretically the tracking process and proposed to introduce control points for B2B scenarios by modifying the data structure without giving more detail on how to do that. The lack of use of IoT affect the automation and the data collection capabilities of those solutions.

Other related works take advantage of the combination of smart contract and IoT to resolve traceability problems as in Bumblauskas et al. (2020). Wen et al. (2019) proposed a privacy compliant traceability solution, but with a limited number of stakeholder roles that does not cover all the possible roles in the supply chain such as the broker role for example. The shipment management system in Hasan et al. (2019) handles only some limited package status and transport conditions that could not cover all different contexts specific needs in terms of status and transport conditions. In the model of architecture for Food Supply Chain (FSC) traceability proposed by Casino et al. (2019), the IoT data are stored only locally and a reference to the locally stored data is used in blockchain, but there is no reference in their work to an IoT data qualification process.

All the aforementioned smart contracts have been developed for some specifics supply chain contexts and are not generic to be used in other contexts. Also, their proposed solution lacks methods to qualify IoT Data before its integration in the smart contract. That impacts the performance because of the huge amount of data generated by the IoT and that need to be

stored in the underlying blockchain. The incidents management also is an important part of the traceability process that has not been or not well treated in those related works.

Table 1 summarizes the studied works and how they meet the studied five requirements:

Table 1: Related works comparison

Related work	IoT (R1)	Genericity (R2)	Contractual Milestones (R3)	IoT data Qualification (R4)	Incidents Management (R5)
Lin et al. (2019), Westerkamp et al. (2018), Cui et al. (2019), Yong et al. (2020), Salah et al. (2019) and Helo & Hao (2019)	N/A	N/A	N/A	N/A	N/A
Chang et al. (2019)	N/A	N/A	B2B control points	N/A	N/A

Related work	IoT (R1)	Genericity (R2)	Contractual Milestones (R3)	IoT data Qualification (R4)	Incidents Management (R5)
Casino et al. (2019) and Bumblauskas et al. (2020)	Fulfilled	N/A	N/A	N/A	N/A
Wen et al. (2019)	Fulfilled	Generic smart contract (without transport conditions)	N/A	N/A	N/A
Hasan et al. (2019)	Fulfilled	Smart contract with hard coded shipment status and transport conditions	N/A	N/A	N/A

Related work	IoT (R1)	Genericity (R2)	Contractual Milestones (R3)	IoT data Qualification (R4)	Incidents Management (R5)
This work	Fulfilled	Generic smart contract with generic transport conditions	Contractual milestones management	IoT data qualification process	Incidents auto detection and qualification in the smart contract

3 Architecture of the proposed B2B traceability solution

As depicted in Figure 3, we propose a novel architecture that meet all the requirements defined in the related works section.

The proposed B2B traceability solution is comprised of two main parts: the generic smart contract and the server responsible of IoT data qualification. The smart contract is responsible of the contractual milestones and the incidents detection and management process. The IoT data qualification server is responsible of the qualification of the collected IoT data based on the constraints defined by the smart contract data and the IoT tag specifications.

As a relevant illustrative application, we choose some scenarios from the medical cold chain. These use cases involve generally B2B stakeholders and have specific requirements such as specific temperature transport conditions in case of transport of medical temperature-sensible products. For those reasons, they are relevant use cases for our proposed traceability solution.

In the next subsections, we present the medical cold chain use case and the selected illustrative scenarios, the generic traceability smart contract and the IoT data qualification process.

3.1 Medical cold chain use cases

The medical equipment cold chain is handled by specific transport means. Some of the medical equipments, like perishable medical diagnostic kits

used in blood tests, need to be transported under elevated transport conditions with a temperature between a minimum of +2 and a maximum of +8 Celsius degrees. Shipper generally use data loggers to collect the temperature during the transport operation data and those data loggers are returned to the shipper after the end of the transport operation to control if there has been any incident related to the temperature excursion.

Furthermore, the transport conditions and milestones data are made available by carrier through EDI or web services. In existing traceability systems, all those data are collected in a central traceability system and can then be accessed by all stakeholders.

This scenario with current systems presents several challenges that are not handled by most of traditional traceability systems. For example, the incident declaration is delayed due to the post-control of the transport condition respect after the end of the transport operation using the returned data logger. Also, the central traceability system hosted by one of the stakeholders or an external actor, without guaranties on the security and the reliability of data available in this system.

For this work, we present three implemented scenarios that show the genericity of our smart contract and how it handles different types of incidents. Those scenarios involve three main stakeholders: a shipper, a carrier and a consignee. In all these scenarios, connected IoT tags (Figure 2) accompany the shipments and collect automatically in near real time the shipment temperature and position.

The first scenario is about the transport of Cytomegalovirus (CMV) test Kit, used to diagnostic human CMV infections. This test kit needs to be transported under strict temperature condition of [+2°C,+8°C]. In this scenario

we focus on the incidents related to the non-compliance with the temperature transport condition enshrined in the shipment transport conditions.
The second scenario is about the transport of Thyroid Stimulating Hormone (TSH) test Kit, used as a diagnostic test for common condition of thyroid hormone deficiency. In this scenario we focus on incident related to the non-compliance with delivery date concluded for the shipment delivery milestone.
The last scenario is about the transport of Automatic immunohematology analyzer, and in this scenario, we focus on incident related to the damaging of transported material either at origin or during placement in the aircraft.

3.2 The smart contract

The smart contract proposed in this work handles the logic, the traceability rules and the incidents management in a generic way. In order to do that, we used generic entities (See Figure 1) and methods without any reference to a specific B2B traceability context, such as specifics stakeholders, milestones, transport condition or incidents. Therefore, this smart contract could be used in the majority of the B2B supply chain traceability contexts without needs for further modifications of its entities or methods. Additionally, the incidents automatically detected by this smart contract are well organized by their non-compliance origins (transport conditions or milestones) and their stakeholders also are well identified. This organization gives a clear view of those incidents and simplifies their management process.

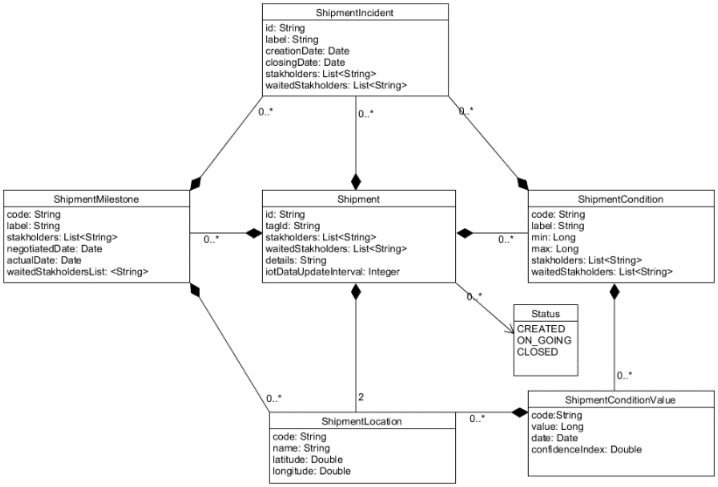

Figure 1: Entity class diagram

The main methods of the proposed smart contracts are *createShipment, updateMilestone, addIoTEvent, createIncident and confirmIncident.*

The *createShipment* method is called by the initiator of the shipment to create a new shipment. It takes as argument a description of the shipment to be created with all its related elements: description, milestones, transport conditions and stakeholders. The method initializes the shipment milestones with empty incidents list, verifies that the initiator organization is not in the waiting for confirmation stakeholders list, if that's the case remove it and initiates the shipment status. It gives as output the created shipment.

The *updateMilestone* is called by the concerned milestone stakeholders to update the milestone actual date and location. It takes as argument the

milestone to be updated and gives as output the updated milestone, see Algorithm 1. If necessary, it creates a milestone date compliance incident.

Algorithm 1: Update milestone

Input: An existing shipment id and an existing milestone related to the given shipment
Begin
if The update requestor company is in the shipment stakeholders list and also in the milestone stakeholders **then**
Retrieve and update the given milestone actual date and location using the milestone code
if The milestone actual date is after the milestone negotiated date **then**
Create a milestone non-compliance incident involving all the milestone stakeholders
end if
else
Throw error: unauthorized update
end if
End
Output: Updated milestone

The *addIoTEvent* is called by the IoT data qualification server when an IoT event (ShipmentConditionValue) is received and is eligible to be sent to the smart contract. It takes as argument the qualified IoT event and gives as output the updated shipment, see Algorithm 2. It may also generate a transport conditions compliance incident.

Algorithm 2: Add IoT event

Input: An existing shipment id and an IoT event related to an existing transport condition of the given shipment
Begin
if The update requestor company is in the shipment stakeholders list and also in the transport condition stakeholders **then**
Retrieve the concerned transport condition using its code and add the event to
the transport condition events
if The event value is not compliant with the fixed transport condition ranges **then**
Create a transport condition non-compliance incident involving all the transport condition stakeholders
end if
else
Throw error: unauthorized update
end if
End
Output: Updated shipment

The *createIncident* method is called by the shipment stakeholders to *r*eport manually other types of incidents that are not directly related to the milestone's dates or the transport conditions respect. For examples a damaged material incident. It takes as arguments the shipment's id and the incident to be created and gives as output the updated shipment.

The *confirmIncident* method is called by shipment stakeholders to confirm that they are effectively involved in the given incident. It takes as arguments the shipment id and the id of the incident to be confirmed, and gives as output the updated shipment, see Algorithm 3.

Algorithm 3: Confirm incident

Input: An existing shipment id and an existing incident id related to the given shipment
Begin
if The confirm requestor company is in the shipment stakeholders list and also in the given incident stakeholders **then**
Retrieve the concerned incident and remove the requestor company from the list
of the incident waited for confirmation stakeholders
else
Throw error: unauthorized update
end if
End
Output: Updated shipment

As instantiation examples of our generic smart contract, we use the above presented three scenarios. For all those scenarios, the stakeholders will be the shipper, the carrier and the consignee. Those stakeholders agree on the following basic milestones list: Pickup (involving the shipper and the carrier), Departure (carrier), Arrival (carrier) and Delivery (carrier and consignee). The shipper calls the smart contract method *createShipment* to create a shipment.

In the first scenario, we only have a temperature transport condition with a minimum of +2°C and a maximum of +8°C, which results in the following transport condition instance: («TEMP (code)», «Temperature (label)», «2 (min)», «8 (max)», «the shipper and the carrier as temperature transport condition stakeholders»). When an out-temperature range value (10 for example) is received by the smart contract, it creates automatically an incident related to the temperature transport condition with the following information: («incident auto generated id (id)», «Non-compliance with Temperature transport condition [2,8], the received value was 10 (Label)», «the received Temperature date (Creation date)», «the shipper and the carrier as incident stakeholders (the incident stakeholders are the same as the transport condition stakeholders)»).

In the second scenario, we have a delivery date of 12/03/2020 at 13h00 for the shipment delivery milestone, which results in the following milestone instance: («DLV (code)», «Delivery (Label)», «12/03/2020 at 13h00 (negotiated date)», «the carrier and the consignee (delivery milestone stakeholders)»). When the smart contract receives an actual date which is after the negotiated date (12/03/2020 at 16h00 for example), it automatically creates an incident related to the delivery milestone with the following information: («incident auto generated id (id)», «Non-compliance with Delivery milestone negotiated date 12/03/2020 at 13h00, the received actual date was 12/03/2020 at 16h00 (Label)», «The received milestone actual date (Creation date)», «the shipper and the carrier as incident stakeholders (the incident stakeholders are the same as the milestone stakeholders)»).

In the last scenario, the incident related to the damaging of transported material is reported manually. For example, the shipper calls the smart contract method *createIncident* and gives all the incident related information,

for example: (« Damaging of transported material (Label)», «The formal date of the incident (Creation date)», «the carrier as incident stakeholders (the incident stakeholders designated by the shipper)»).
The above presented smart contract, in contrast to the existing smart contracts in the state of the art, is more adapted to the B2B traceability context, with its integrated milestones, IoT data and incident management. The genericity of this smart contract allows its deployment in various B2B traceability context without needs of further development efforts.

3.3 The IoT Data collection and qualification

The architecture of our traceability solution is designed to automatically detect traceability incidents by the smart contract based on the data collected by the IoT. Due to the encryption and the replication of blockchain data, the blockchain is not a good storage support for huge data amount. The data generated by the connected objects is not directly integrated into the smart contract. An IoT data server is used to improve the IoT data quality and send to the smart contract only qualified IoT data, see Figure 2.
We consider the following approaches for enhancing IoT data quality: outlier detection, data cleaning and data deduplication as stated by Karkouch et al. (2016). Other aspects of the IoT data qualification such as interpolation and data integration need to be considered in future work.

In this work, we consider as outlier data, every value that is outside the ranges of the possible values defined by the sensor's specifications. The outlier data are not sent to the smart contract. The data cleaning is performed through the comparison of the format of the received IoT data with the expected data format and the verification of the IoT data date which should be valid and not a future date.

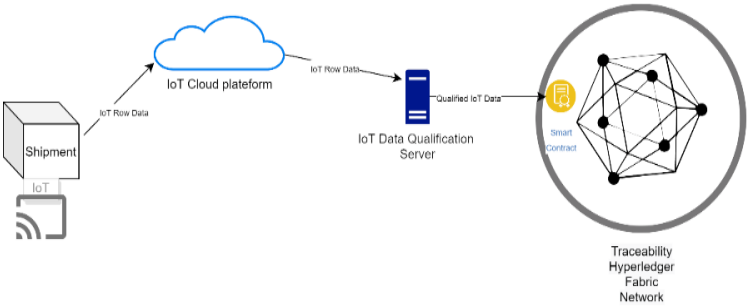

Figure 2: IoT Data qualification process

The IoT data deduplication is performed by an IoT data filter used to reduce the number of IoT events sent to the smart contract. This filter takes as argument the IoT event to qualify, the last received IoT event and the shipment. It gives as output the list of events sent to the smart contract, see Algorithm 4.

Algorithm 4: IoT Data Filter

Input: An existing shipment id or shipment tag number and a new IoT event value

Begin

Retrieve the correspondent transport condition ranges and the shipment IoT data timeout interval

if The shipment IoT data timeout interval is not elapsed **then**

if The new IoT event value is in the ranges **AND** the last sent value to the smart contract was outside the ranges **OR** the new IoT event value is outside the ranges

AND the last sent value to the smart contract was in the ranges **then**

send the new IoT event value to the smart contract

send also the previous received value to the smart contract, if it has not been already sent

end if

else

send the new IoT event value to the smart contract

end if

End

Output: Events sent to the smart contract

---

4 Implementation, Test and Evaluation

In this section we present an implementation of the smart contract proposed in this work. Some performance tests and results are also presented, in order to prove the ability of the proposed architecture to be deployed in real live production scenarios. Finally, we evaluate the proposed architecture based on performance test results.

4.1 Implementation

The proposed smart contract has been implemented using Hyperledger Fabric Java Chaincode. Hyperledger Fabric is a permissioned blockchain implementation designed for enterprise purposes. It presents many advantages in comparison with the other permissioned blockchain implementations, among them : a parametrized consensus protocol, a node architecture based on the notion of organization to establish a trust model more adapted to the enterprise context and the support of Go, Javascript and Java for smart contracts writing.
For the development and the deployment of our smart contract, we have used the following software versions:

Table 2: Test software versions

Software	Version
Hyperledger Fabric Docker Images Tag	1.4.6
Hyperledger Fabric Java Chaincode	1.4.3
Hyperledger Fabric Gateway Java	1.4.1
Docker	19.03.6
Java	1.8.0

For deployment purpose, we implemented a simplified Hyperledger Fabric architecture (Figure 3) with three stakeholders interacting with the blockchain: a shipper, a carrier and a consignee. In this architecture, the IoT data

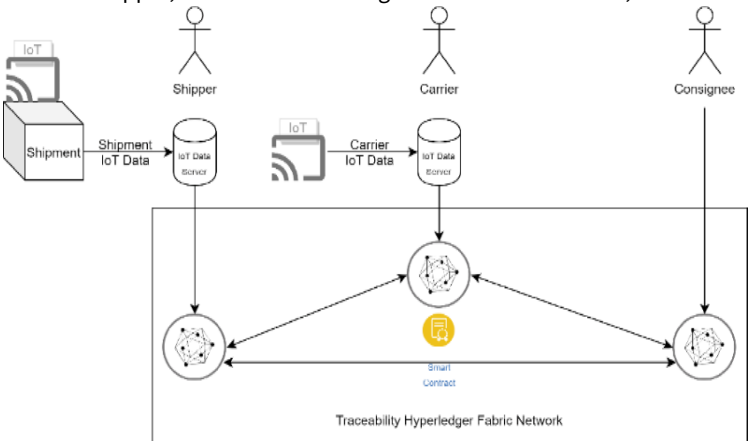

Figure 3: Global architecture of the solution

sent by the shipment IoT tag or by the carrier is qualified in local IoT Data Servers before its integration in the smart contract.

The stakeholders have been created as independent Hyperledger Fabric organizations. Each organization has the following components: Certificate Authority, responsible of the organization user certificates management; Tow peers, with a local CouchDB database for each peer. One of the two peers is designated as the endorser peer, which is responsible of the correct execution of the smart contract on the organization side.

All the endorsers peers are connected to a channel called « my-channel ». The transaction order is handled by one ordering service node, see Figure 4.

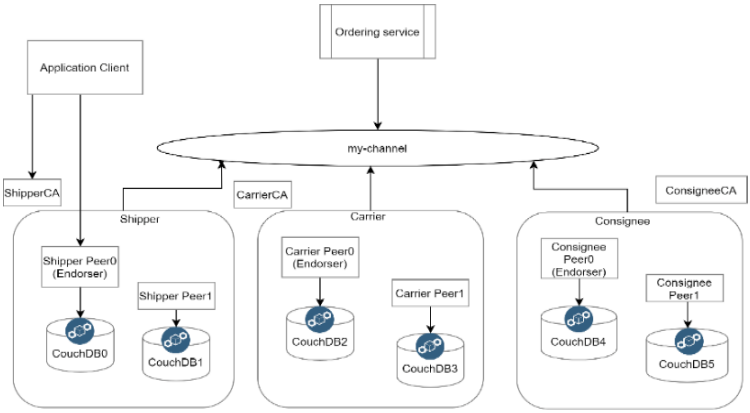

Figure 4: Detailed architecture of the Test Hyperledger Fabric Network

4.2 Test and evaluation

The objective of this subsection is to test the performance of the proposed smart contract and the IoT data qualification module, in order to show that

they can be used in a real live traceability system. It's not a subsection about the Hyperledger Fabric performance which has been already treated by Androulaki et al. (2018) and Yuan et al. (2020). For our smart contract POC, the performance tests objective is to prove that the response time of the main smart contract methods is tolerable. In order to evaluate the response time tolerance, we defined a user tolerable response time of maximum 3 seconds, based on the work of Zhou et al. (2016).

Regarding the IoT qualification module, the tests objective is to prove that this module helps to reduce considerably the amount of IoT data to be sent to the smart contract.

For the test purpose, one machine has been used, and all the architecture components have been deployed on this machine using Docker. The test machine has the following characteristics:

Table 3: Test machine characteristics

Characteristic	Details
OS	Ubuntu 18.04.4 desktop amd64
CPU	1 CPU Intel(R) Core™ i7-8565U
RAM	8G
Virtual Disk	50G

We set the Hyperledger Fabric block creation timeout to 1 second and the maximum number of transactions per block to 15. This means that after the reception of a new transaction, the system will trigger the block creation

either after a time wait of 1 second or after a total number of 15 new transactions is reached. In this POC, we use the Raft consensus algorithm (Ongaro & Ousterhout (2014)), with a unique ordering service node.

The *createShipment*, *updateMilestone* and *addIoTEvent*, the three main methods of the smart contract have been tested using three batches of shipments. A first batch of 500 shipments, a second batch of 1000 shipments, and a last batch of 2000 shipments. The average response time of the three methods was around 1.4 seconds. It's a good result for the POC regarding the single machine used to deploy all the architecture components. However, further tests are needed to confirm the performance of this architecture in a real distributed environment with network constraints and more stakeholders, since those elements could impact the architecture performance.

The proposed IoT Data Qualification module has also been tested 1000 times using a series of 1000 randomly generated temperature values. The shipment IoT data interval was set to 60 minutes and there was a timeslot of 3 minutes between every two values of the temperature series. Those parameters have been chosen from a specific context of shipment IoT tag that use Sigfox technology to send a message of 3 temperatures values every 10 minutes. This gives around 1000 temperature values received for 2 days of IoT data collection.

As result of those tests, the percentage of retained IoT data to be sent to the smart contract by the IoT data qualification server, depends on the test series values. For normal series with only in-ranges temperatures, and abnormal series with only out-ranges temperatures, this percentage is equal to the IoT values timeslot divided by IoT update interval defined in the shipment, which gives 5% in our test case (3/60). In case of a mixed series, the

percentage of retained IoT data depends on the IoT values timeslot, the IoT update interval defined in the shipment and the number of out-ranges temperatures. For our test case, with a series of around 36% of out-ranges values, the percentage of retained IoT data was around 31%.

The proposed IoT data qualification method allows getting into the smart contract only the pertinent IoT data for the traceability management and reduce considerably the number of IoT events to be stored in the underlying blockchain. However, many other aspects of the data quality have not been covered in this work such as the heterogeneity of sources (for example shipper IoT data vs carrier IoT data) and need to be considered in future work.

5 Conclusion and future work

In this paper, in order to enhance B2B supply chain traceability, we have proposed, implemented and tested a generic smart contract for B2B traceability. This smart contract handles contractual milestones, IoT data and the auto detection and qualification of traceability incidents. We developed also a solution for the qualification of IoT data before its integration in the smart contract.

The architecture proposed in this paper could be enhanced by future work on the data to be stored outside the blockchain (off-chain data), for example by using IPFS technology (Benet (2014)) to share and synchronize off-chain data and to store traceability attachments files, in a distributed architecture. Also, other aspects of the IoT data quality such as the handling of imperfect data, the data integration and interpolation, need to be considered in future work. Additionally, we have proposed some manual confirmation methods in this work. The impacts of those methods on the traceability process automation should be studied versus the trust that those methods add to shipment, milestones and incidents data.

Furthermore, this work has some managerial implication such as the need to subscribe the negotiated milestones in the stakeholder's service contract, the approval of connected objects to be used for data collection, some data manual confirmation process and the management by exception through a clear vision of elements that are non-compliant with the stakeholders concluded service contract.

Finally, the research on the use of smart contracts and the Internet of Things (IoT) suffers from several limitations out of the scoop of this work, among them: the difficulty of deploying blockchain based solutions, the maturity of blockchain technology, and the securing of IoT data collection and processing.

References

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolic, M., Cocco, S. W. & Yellick, J. (2018), Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains, pp. 1–15. arXiv: 1801.10228. URL: http://arxiv.org/abs/1801.10228

Benet, J. (2014), 'IPFS - Content Addressed, Versioned, P2P File System', arXiv e-prints p. arXiv:1407.3561.

Bumblauskas, D., Mann, A., Dugan, B. & Rittmer, J. (2020), 'A blockchain use case in food distribution: Do you know where your food has been?', International Journal of Information Management 52, 102008. URL: http://www.sciencedirect.com/science/article/pii/S026840121930461X.

Buterin, V. (2014), 'Ethereum: A next-generation smart contract and decentralized application platform'. Accessed: 2020-04-28. URL: https://github.com/ethereum/wiki/wiki/White-Paper

Casino, F., Kanakaris, V., Dasaklis, T. K., Moschuris, S. & Rachaniotis, N. P. (2019), 'Modeling food supply chain traceability based on blockchain technology', IFAC-PapersOnLine 52(13), 2728 – 2733. 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019. URL: http://www.sciencedirect.com/science/article/pii/S2405896319316088

Chang, S. E., Chen, Y.-C. & Lu, M.-F. (2019), 'Supply chain re-engineering using blockchain technology: A case of smart contract based tracking process', Technological Forecasting and Social Change 144, 1 – 11. URL: http://www.sciencedirect.com/science/article/pii/S0040162518305547

Cui, P., Dixon, J., Guin, U. & Dimase, D. (2019), 'A blockchain-based framework for supply chain provenance', IEEE Access 7, 157113–157125. Hasan, H., AlHadhrami, E., AlDhaheri, A., Salah, K. & Jayaraman, R. (2019), 'Smart contract-based approach for efficient shipment management', Computers&Industrial Engineering 136, 149 – 159. URL: http://www.sciencedirect.com/science/article/pii/S0360835219304140

Helo, P. & Hao, Y. (2019), 'Blockchains in operations and supply chains: A model and reference implementation', Computers & Industrial Engineering 136, 242–251.
Hinckeldeyn, J. & Jochen, K. (2018), (short paper) developing a smart storage container for a blockchain-based supply chain application, in '2018 Crypto Valley Conference on Blockchain Technology (CVCBT)', pp. 97–100.

Karkouch, A., Mousannif, H., Al Moatassime, H. & Noël, T. (2016), 'Data quality in internet of things: A state-of-the-art survey', Journal of Network and Computer Applications.

Kshetri, N. (2018), '1 blockchain's roles in meeting key supply chain management objectives', International Journal of Information Management 39, 80.
URL: http://search.proquest.com/docview/2059164679/

Lin, Q., Wang, H., Pei, X. & Wang, J. (2019), 'Food safety traceability system based on blockchain and epcis', IEEE Access 7, 20698–20707.

Ongaro, D. & Ousterhout, J. (2014), In search of an understandable consensus algorithm, in '2014 USENIX Annual Technical Conference (USENIX ATC 14)', USENIX-Association, Philadelphia, PA, pp. 305–319. URL: https://www.usenix.org/conference/atc14/technicalsessions/presentation/ongaro

Rejeb, A., Keogh, J. & Treiblmaier, H. (2019), 'Leveraging the internet of things and blockchain technology in supply chain management', Future Internet 11, https://www.mdpi.com/1999–5903/11/7/161.

Salah, K., Nizamuddin, N., Jayaraman, R. & Omar, M. (2019), 'Blockchain-based soybean traceability in agricultural supply chain', IEEE Access 7, 73295–73305.

Sigfox technology (n.d.). URL: https://www.sigfox.com/en/what-sigfox/technology

Szabo, N. (1997), 'Formalizing and securing relationships on public networks', First Monday 2(9). URL: https://ojphi.org/ojs/index.php/fm/article/view/548

Van Dorp, K. (2002), 'Tracking and tracing: a structure for development and contemporary practices', Logistics Information Management 15(1), 24–33.
URL: https://doi.org/10.1108/09576050210412648

Wen, Q., Gao, Y., Chen, Z. & Wu, D. (2019), A blockchain-based data sharing scheme in the supply chain by iiot, in '2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)', pp. 695–700.

Westerkamp, M., Victor, F. & Küpper, A. (2018), Blockchain-based supply chain traceability: Token recipes model manufacturing processes, in '2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)', pp. 1595–1602.

Yong, B., Shen, J., Liu, X., Li, F., Chen, H. & Zhou, Q. (2020), 'An intelligent blockchainbased system for safe vaccine supply and supervision', International Journal of Information Management 52.

Yuan, P., Zheng, K., Xiong, X., Zhang, K. & Lei, L. (2020), 'Performance modeling and analysis of a hyperledger-based system using gspn', Computer Communications 153, 117 – 124.
URL: http://www.sciencedirect.com/science/article/pii/S0140366419306474

Zhou, R., Shao, S., Li, W. & Zhou, L. (2016), How to define the user's tolerance of response time in using mobile applications, in '2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)', pp. 281–285.