T·MINING
BLOCKCHAIN LOGISTICS

Blockchain in Logistics

whitepaper

This paper is an introduction to blockchain. It does not explain the technical details of how blockchains work, but simplifies a number of things in order to make everything comprehensible. Simple use cases from the logistics market serve to demonstrate the possibilities of blockchain.

Author:
Pascal Verlinden
CTO T-Mining
pascal.verlinden@t-mining.be
www.t-mining.be

Version 1.3
November 2018

TABLE OF CONTENT

Introduction	4
What is blockchain?	6
Why blockchain?	8
Smart contracts	13
Tokens, accounts and balances	18
Synchronization and fault tolerance	20
Conclusion	22

INTRODUCTION

Blockchain is a technology that enables people to do business with each other in a direct - peer to peer- way, even if they are not sure they can trust each other completely.

Today, this lack of trust often requires a so-called middleman: two parties that want to do business with each other rely on a third party they can trust. A common example would be the transfer of money, which is typically handled by a bank, acting as the “middleman”. The bank checks if the sender has enough money and then makes sure it is transferred correctly from the sender to the receiver’s account.

There are many more examples of business processes that are handled by systems which are operated by mediating organizations or authorities. As they are under control of one organization, they are known as centralized systems.

Although these systems work, there are some inconveniences which make them less attractive under certain conditions.

Normally, we depend on the organization that controls the system. For instance, we need to trust that the bank will not go bankrupt or that a malicious employee finds a way to steal our money. There are quite some guarantees to protect against such risks, however, these risks can still happen.

Centralized systems must be built in a way that they are very secure. It is possible that system crashes could make it temporarily unavailable, unreliable or give an opportunity for hackers to break into the system. This requires a lot of

effort and money to safeguard centralized systems against these kinds of problems. Obviously, this comes with a cost, which you would have to pay.

In contrast, there is an alternative system, which is not based on centralization. The challenge would now be to ensure that nobody can cheat, like e.g. transferring the same money twice, aka “double spending”.

This was the original problem that was solved by Bitcoin, the first implementation of a blockchain. Bitcoin was designed to enable peer-to-peer money transactions without the need for a central system that still protected users against double spending.

While this was the original focus of Bitcoin, people explored ways to use blockchain in a broader context.

WHAT IS BLOCKCHAIN?

If we need to be sure that our transactions are processed correctly and remain unchanged (“tamper-free”) without using a central system, we need to broadcast the transactions to a network of computers that they will all get the same set of transactions, in the same order. These computers must be able to ensure that nobody can change transactions on any of the computers, without being noticed. This is - essentially - what blockchain does.

The blockchain itself is an ordered list of items that is broadcast to a set of computers that are connected in a network, with the following main characteristics:

- the items - transactions - of the list are ordered and grouped into “blocks”,
- each block is linked to the preceding block, leading to a “chain” of blocks,
- the list is “immutable”: blocks - hence transactions - that have been included in the list, can no longer be removed or changed, as this would “break the chain”,
- the computers in the network use a special “consensus” algorithm to agree on the next block of items that gets added to the list, such that all computers have the same list.

There are other characteristics of a blockchain, however, these are deliberately left out as they are not essential for understanding the concept. Let’s elaborate this a bit further.

First, what are these “items” we put on the list? Here are some examples:

- transfer 5 euro to John;
- mark a container as available for pickup;
- set a variable x to the value 1.

The first example is a typical transfer of money, like what Bitcoin does. The two other examples look different at first glance, however, they can all be considered state transitions. Transferring money changes the state of the account of the sender and the receiver. The second example changes the state of a container, whereas the last one sets the state of a variable x.

In a blockchain, users enter transactions resembling the examples above. Many computers (the “miners” / “validators”) in the blockchain network have a special function. To start, they collect these transactions into a new block and then they calculate the new state by applying all transactions in the block. Finally, they broadcast this block, which includes the ordered transactions and the calculated state, to the network.

The other computers validate this block, which includes recalculating the state based on the transactions in it, by doing this they compare with the state that is recorded in the block itself. If they do not match, the block is considered invalid and will not be accepted. If they do match, they add the block to their copy of the chain. This way all computers get the same state and chain of valid blocks.

This process may differ depending on the type of blockchain, but the objective is to make sure all computers in the network agree on the next block. Which they will then apply a state transition function so they all have the same end state.

The immutability feature of blockchain ensures that this state cannot be changed afterward by anyone. As a result, everyone can “trust” the blockchain to represent the correct state as agreed upon by all participants.

Note that this description is of a high level and incomplete: technical details are left out for the sake of simplicity.

WHY BLOCKCHAIN?

To understand the advantages of blockchain, let's look at a simple example in the logistics market. Assume a container must be picked up by a barge at a deep-sea terminal in a port.

Planning the barge must be done upfront, as the visit of the barge at the terminal must be agreed upon with the terminal operator several days before the barge will arrive at the terminal. This planning process will be done in an early stage, possibly even before the container arrives at the port. As a result, the barge operator needs to know the estimated time of arrival (ETA) of the vessel carrying the container. This will ensure that the operator will get an idea of when the container will be expected to be available at the deep-sea terminal.

Although this seems like information that should be readily available, the barge planner often must send e-mails or make phone calls to the terminal operator to find out. Simultaneously, the terminal operator also must get this information somewhere, usually from the carrier operating the vessel or a vessel traffic system.

To summarize,our barge planner must spend a considerable amount of effort to find out the ETA, to plan his barge. Since it is an estimated time that is not set in stone, the planner may have to check the ETA again for updates if the vessel has yet to arrive at the terminal.

Undoubtedly, making phone calls and waiting for a response from an e-mail are not the most efficient ways to figure out something as simple as an ETA. So, how could this be improved?

If the terminal operator has a system that records the ETA of all vessels that will visit their terminal, they could then build a web application that allows our barge planner to consult the ETA.

However, this would not be very efficient. As there are many terminal operators, this approach would mean that all terminal operators would have to implement this system, which then the barge planner would have to check every single system.

In comparison, the port authority could decide to build a port community system which collects the ETA's making them available through a web interface, possibly including an API (application programming interface). This approach would be remarkably better, in fact, this kind of community system has already been implemented at several ports showing that is a viable way to collect and distribute data.

Now let's have a closer look at these approaches to find out if we can still improve this.

As these systems are centralized, there is one party that administers and controls them. An administrator can do whatever he wants: he can change the processing logic and he has full access to all data. This is not necessarily a problem, at least not in the context of an ETA, but it could be if the data is more sensitive.

We also expect the system can hold the most up to date and correct ETA's always. However, a central system is usually based on second-hand information, where they depend on other systems to get the data. As a result, they may not always have the latest information.

Moreover, a central system will typically receive data, perform some processing on it and then distribute it. Obviously, the processing part could introduce errors, and as a result, there would be no guarantee that the information they have is correct. Furthermore, the processing logic would be implemented by and under control of the organization that manages the system, but not by the users.

There are technical challenges too, as specific measures must be taken to make sure a central system can be available and reliable. Otherwise, the central system typically becomes a single point of failure, which could cause a problem if many parties rely on its availability. The system must include a robust authentication, user management, accessibility on the Internet and protection against hacking. These challenges lead to a high development and maintenance cost for the system.

At last, transport is a global activity, involving many parties located in different countries and regions. Many central systems, such as port community systems, are considered “local” meaning that they offer services related to a local port or a local community. However, our barge operator may have to go to more than one port, possibly in different countries. If each port would have an implemented system, this would mean that he would have to interact with each of these systems. This would then again introduce inefficiency, as local central systems only solve part of the problem!

Let’s see how blockchain can help us to avoid these issues.

Blockchain allows us to set up a fully decentralized system, which is a network that connects each of the organizations involved in the transport of our container, like the carrier, terminal operator and barge operator.

Each organization that wants to participate in such a blockchain network would need to install software that allows them to interact with the blockchain, and hence with the other organizations in the network. In this text, we refer to this software as a “blockchain client”.

With this software installed, the computer becomes a so-called “node” of the blockchain network. There can be different flavours of nodes, however, this is not important for the remainder of the text.

Now, let’s go back to the example with the ETA. Assume that if the correct value is known by the carrier, the carrier and the barge operator will install the blockchain client to get a node in the blockchain network. The carrier system is now able to talk to its local blockchain client to record the ETA. This is treated as a transaction that updates the state of the vessel, which will then be put in a block that is automatically broadcasted to all nodes.

As a result, our barge operator automatically gets access to the latest ETA and any future ETA updates on its node.

Let's compare this with the central system approach, by looking at the problems described above.

With blockchain, there is no central authority that has full control aseach company that joins the network will have access to the latest state and the ability to enter new transactions.

Any processing of the data can be programmed into the blockchain, if it is validated and agreed upon all parties and unable to be changed by a single party. To understand this better, we will go into detail on this topic further in the paper.

Transactions are cryptographically signed by the user who enters the transaction. Due to this, we will now have the ability to determine who has entered an ETA into the system, and whether the information is indeed the correct source of the data.

The blockchain can be programmed to make sure that this ETA can only be updated by the user who is in charge, i.e. the owner of this information. As well as, when an update is made, the blockchain protocol will automatically broadcast it to the other nodes in the network. Therefore, the data is automatically kept up to date, based on inputs from the user who owns the data, without having to go through an intermediate system that may cause delays or processing errors.

What about availability? Under the hood, blockchain systems replicate the transactions, like the ones to set an ETA, to all nodes in the network. Nodes that join the network will automatically synchronize, and as a result, we get a system that is much more fault tolerant and has out-of-the-box high availability. Once you set the ETA, this value is not recorded on your node only: it resides "on the network", and when your node would crash, the value is not lost: when you bring your node back on, it will catch up with the other nodes in the network.

Authentication and user management are native features of a blockchain, and the blockchain consensus algorithm protects against hacking and unexpected behaviour. For a better understanding of this concept, we will revisit it later in this document.

Regarding the global aspects of transport, it should be clear that a blockchain is global “by definition”. It is a network of nodes, that connects all peers, wherever they are in the world, and all peers can communicate with each other, all using the same protocol.

As a result, if our barge planner needs to know the ETA of a ship in a local port and a port in a neighbouring country, he does not have to do anything to make this possible: the only requirement is that the organizations that supply the ETA in both ports have joined the network.

This way, our blockchain indeed has an answer to all of the issues mentioned before.

The remainder of this paper discusses features as implemented in blockchains based on the Ethereum Virtual Machine. Ethereum is a specific blockchain implementation, which includes a programming language and a virtual machine (VM) that can run the programs written in this language.

SMART CONTRACTS

Assume a forwarder oversees a container transport from a deep-sea terminal to a warehouse. He works with a trucking company that does the actual transport. A dispatcher from the trucking company will assign the transport to one of the drivers, who will then go to the terminal, pick it up, and bring it to the warehouse to deliver it.

Several parties are involved, e.g. the carrier, terminal operator, forwarder, and trucking company. Each of them perform actions that change the state of the container, which must be known by the other parties so they can do their part of the job correctly.

The carrier will “release” the container that involves a numerous amount checks, for example, whether all payments are done.

When the ocean vessel arrives at the terminal, the terminal operator will discharge the container from the vessel. The terminal will need to know that the container has been released. When both the release and the discharge have been done, the container is available for pickup.

The truck operator will assign a driver to pick up the container when it is available for pickup. The driver will then pick up the container and deliver it at its final destination. After delivery, the truck operator can send an invoice to the forwarder.

The forwarder will want to know if the container has been picked up and delivered, which means they can send an invoice to the shipper.

Each party has its own information system to support their operations. Typically, this system will record data about the container and keep track of all actions they perform on the container. Since all parties depend on one another, they must exchange information which resides in their local systems. Today, this exchange of information is mostly done by means of messages, which can take many forms: EDIFACT, e-mail, pdf, Excel, even phone calls.
This has several disadvantages:

- a great part of the processing of these messages is still done manually, because there is a large amount of e-mails being sent, either with data in the e-mail body itself, or in attached files. In many cases, this data is unstructured, which makes it hard to process automatically. Needless to say, that manual processes are time consuming and error-prone;
- e-mails are often sent by a human user, so there is no guarantee that the user sends this e-mail right away when a change occurs. This greatly affects the efficiency of the operations of the different parties, even the financials, since invoicing often cannot be done unless the latest status information is available;
- parties may agree to exchange information with structured messages like XML, which helps to solve the problem of manual processing. Each party will then have to implement an interface for receiving and processing the messages. Often, this will lead to multiple interfaces that need to be implemented and maintained, which is rather costly;
- implementing interfaces locally by each party also means that incoming messages are processed differently by each party. It is possible that a party has a slightly different interpretation of the specification of the interface, which may lead to differences in processing, or even errors. So, there is no guarantee that processing the same message yields the same result in each system.

A smart contract enabled blockchain can avoid these problems. A smart contract is a unit of software with permanent storage for recording data and a set of functions for manipulating this data. If you're a programmer, you can compare it with a class.

For instance, you can define a smart contract that represents a container. The storage records properties of the container, such as its type, its owner, and its container number. The contract implements functions for setting and querying

these properties. You can write functions to write any business logic, including e.g.

1. validations;
2. restricting who can perform certain actions, like updating an ETAs, or implementing access control.

Ethereum VM based blockchains include a high-level language that enables us to write these smart contracts and to implement complex logic as required by the business processes amongst multiple parties.
A very important characteristic of these contracts is that they are deterministic: given the same input, they always yield the same result.

Smart contracts get deployed on the blockchain, which will automatically make them available on each node of the blockchain network. As a result, the participants of the network get access not only to data, but to the shared business logic.

Users can call the functions of these smart contracts through their blockchain client. Unless it's a simple query, such a call is treated as a transaction, which will then be put in a block and broadcasted to all nodes. The transaction is then executed on each node by the smart contract, and since it is deterministic, the result will be the same on each node.

We can use smart contracts instead of messages for communication between multiple parties. The data of the message can be represented by fields that are kept in the contract storage, and the contract can implement functions for setting and getting the values of these fields.

Let's illustrate this with a simple example.

Suppose we want to send messages about the availability of a container. We can define a contract representing the container, with the following fields:
the container number;

1. a state variable indicating if the container has been released by the carrier;
2. a state variable indicating if the container has been discharged by the terminal operator;

3. and another state variable indicating if the container is available for pickup.

We also define functions in the contract to:

1. create the container contract with its container number;
2. set the state variables;
3. query the state variables.

Now, the carrier can create a new container contract with its container number. This contract will then get deployed onto the entire network. Once the carrier has done the necessary checks, he can release the container, by calling the appropriate function of the container contract. The other parties will then know the container has been released.

Next, the terminal operator discharges the container from the vessel and then calls another function of the contract to change the state of the container. Other parties, such as the forwarder and trucking company, will immediately see the new state changes and will know that the container has been released and discharged. Afterwards, they can decide to assign a driver to pick up the container. This would replace the need for sending a message to inform other parties that the container is available.

It is important to mention that all parties are now using the same protocol, which was implemented by the blockchain, are now able to get the same view of up to date information. To update the other parties, the carrier will no longer need to send a message to the terminal operator, forwarder and trucking company. The carrier will only need to create a transaction stating that the container has been released on his own local node.

To improve this system, we can add the ability to validate the container number. For example, whenever the carrier would create a new container, the contract will ensure that the container number is correct. If not, it will reject the creation. This will ensure that our blockchain network is protected against incorrect information. If you were to view this as a message, this would mean that you cannot send a message with the wrong container number. However, with traditional messaging, it would then be up to every single sender who wants to send data that includes a container number, to implement this same validation.

Using smart contracts has several advantages, such as efficiency, rather than working with the current tasks such as countless e-mails, manual processing and complex interfacing.

When parties participate in a blockchain network, they can agree on smart contracts to define the data they need, including the validations and business logic for managing this data. All parties that are connected to the blockchain and that participate in the contract, can collaborate through that same business logic and easily exchange the information they need.

Obviously, there is no longer a need to implement multiple interfaces with several systems, since the blockchain network connects all parties and enables peer-to-peer communication. This way, we get a single protocol that can be used by all parties.

Smart contracts go further than pure data exchange, because they also include business logic. This way, some of the processing that is typically performed by all parties upon receiving data, can be implemented in the contract, as long as parties share functionality in addition to data.

For instance, we can add a function that automatically derives the availability state of a container when the carrier has released the container and when the terminal operator has discharged the container. Then, the contract can automatically change the containers state as available for pickup.

By putting this logic into the contract, parties agree on the conditions for availability, which is then automatically enforced by the contract. As a result, no party must implement this functionality in its own system anymore.

Finally, parties no longer need to wait for an e-mail or signed paper documents before they can make an invoice or start another process. Parties that are connected directly through the blockchain will always get access to the latest status.

TOKENS, ACCOUNTS AND BALANCES

When you read about blockchain, you'll notice that the different implementations have a so-called cryptocurrency, or token. The best-known example is Bitcoin, which has a native token that is called a Bitcoin which you can use for payment. Another example would be, Ethereum, whose native token is called Ether. One of the basic transactions we are able to perform is transferring tokens to someone else, which means the blockchain must have a way to keep track of our tokens.

Therefore, users get an "account", which includes a balance of the tokens owned by the user which gets updated by the transactions sent.
All accounts will get an address on the blockchain.

Each user will get a public and private key. The address of the user's account is based on the public key. However, transactions sent by a user are signed with his private key. Verification of the signature will be done with the public key of the user.

Each account has a balance that holds the number of tokens controlled by the account which can be altered by sending transactions.

In Ethereum, smart contracts also get an account that contains the contract code. Sending a transaction to a contract account triggers the execution of the contract, using the additional data (payload) provided with the transaction as input.

Ethereum allows to build a so-called token system, with tokens representing value. These tokens can then be used as a payment method among the users of the system. As a result, a smart contract based blockchain can be used to facilitate the business processes among trading parties, including payment for their services.

This means for instance that several parties can agree to collaborate for transporting a container, defining the rewards for their services to each other in a smart contract and including the logic in the contract to automatically pay these rewards as long as the conditions for payment are met.

For instance, when a truck driver delivers the container at the final destination, once the receiver of the container accepts the delivery then the payment can be done. This can be implemented in a smart contract that allows the truck driver to record the delivery and for the receiver to confirm the delivery. When these conditions are met, the contract would then automatically trigger payment by transferring the number of tokens as agreed upon in the contract, to the trucking company.

Note that the concepts above simplify the way people and companies can do business with each other, by the means of a smart contract that defines the rules and automatically sends transactions on the blockchain. This is further simplified by the fact that authentication of users is based on cryptographic keys instead of a password scheme, as used in most applications. The blockchain will then automatically perform payment transactions which eliminates the need for integration with complicated payment services (which in turn cost money).

SYNCHRONIZATION AND FAULT TOLERANCE

Blockchains have remarkable characteristics that usually require additional effort to implement in traditional and cloud based systems.

Blockchains act as replicated state machines, meaning that on each node of the network, they establish the same state based on a set list of ordered transactions. As a result, it is possible for each new node that participates in the network can get its own copy of the transactions and state. So, if a new transport company wants to participate, they only need to get their own node. The blockchain will then ensure the company gets the latest state, as there is an automatic synchronization of the state and data.

Additionally, if the company's node would crash, nothing is actually lost. Since the state is kept on the network, the company does not lose any data. When their node is restored, it will synchronize again with the other nodes. Thanks to this, the company will get a fault-tolerant and disaster proof system.

Admittedly, blockchains can go even further. They can use a consensus algorithm that is "byzantine fault tolerant". This means blockchains can

withstand hardware failures, network congestion, disconnection and malicious attacks such as hacking.

All of this is built into the blockchain implementation, which means that it becomes a lot easier to build robust and safe systems rather than with the traditional frameworks and tools that are used today.

CONCLUSION

Blockchain is a new technology. As it is a hype now, many talk about it, but few will understand it completely and see how it could impact companies.

This text went into depth of the potential benefits in the logistics market, with the use of simple examples to illustrate how it can be used and how effective it can be. Often, the advantages of new technology are not clear in the beginning. In the past, we were able to write letters and post them, but in hindsight it's clear that e-mail was not a gradual improvement. However, it has unlocked a multitude amount of possibilities and innovations. Internet has changed our lives drastically with daily use and the ability to connect globally. As blockchain is a fairly new and disruptive protocol on top of the Internet, which further eases and improves the way we collaborate, with smart contracts that define how the collaboration will take place and how it will be rewarded. All of this is used without the need for mediators, which in the past, introduced extra complexity, cost, lower efficiency and a dependency on these mediators that we don't always want.

However, this is not to say that centralized systems are bad. In many cases, they are set up in order to facilitate the collaboration of companies can do a good job. With that being said, that should not stop us from looking for improvements as it is a normal technological evolution.

Transport and logistics is by nature decentralized: several companies collaborate, compete and depend on each other, peer-to-peer. Blockchain seems a natural fit for this market.

T-Mining is a Belgian based start-up offering a Smart Contract Framework for Logistics. It enables logistic platforms to develop blockchain-enabled software solutions easier & faster by leveraging several ready-made components.

Do not hesitate to contact us via www.t-mining.be for comments on this paper or to discuss your business needs.

***