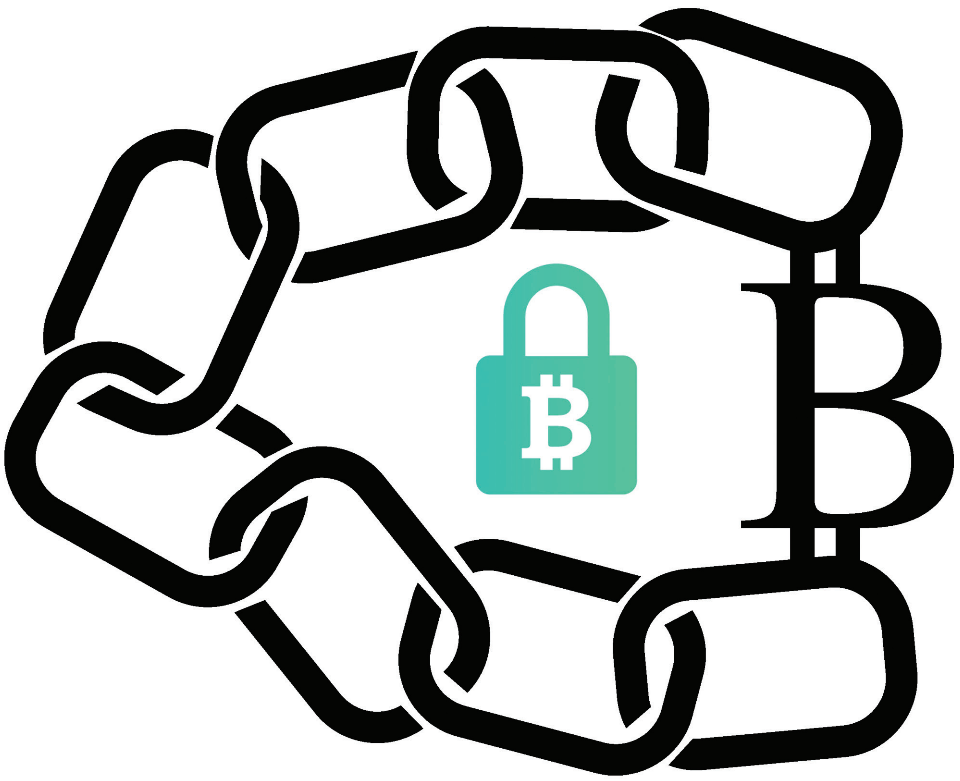
Blockchain-IoT based Management of Shared Resources between Trustless Parties

In this article, the authors discuss use cases for which blockchains and IoT devices can be used, instead of intermediaries, for Trustless parties to transact with each other. The authors also discuss the architecture and the open source technologies that were used in their solution.



Governments, communities or companies often need to track, control and manage assets and resources that they have a shared interest in. For example, in the logistics industry, a perishable item might get transported from a sender to a receiver. While in transit, the perishable item may have to be kept under a certain temperature or pressure. The item may get transported through multiple modes such as ship, train, truck, etc, and multiple transport operators might be involved before it reaches its destination. Payments have to be released to these parties if and only if they transported the item as per the contract.

Often, there is no trust between the sender, receiver and the various parties involved. This means that the parameters required to preserve a perishable item have to be tracked and controlled. And there needs to be a verifiable record of this tracking/controlling throughout the transportation period. There is also a need for a trusted way to settle payments between the various parties once the shipment has been delivered as per the contract.

Another use case is in the energy sector, where smart grids are used for the distribution and consumption of electric power. There is a need for power generators and distributors to track the supply and distribution of power, and keep records of it, to eventually reconcile and settle accounts between the producers and distributors of power. A third use case could be to manage water resources stored in dams and shared between countries/states.

The list of such use cases is endless. Often these parties do not trust each other and end up appointing an intermediary third-party trusted by all of them to manage, control or track the resources, on their behalf. The intermediaries, however, may charge exorbitant fees for their services or may become

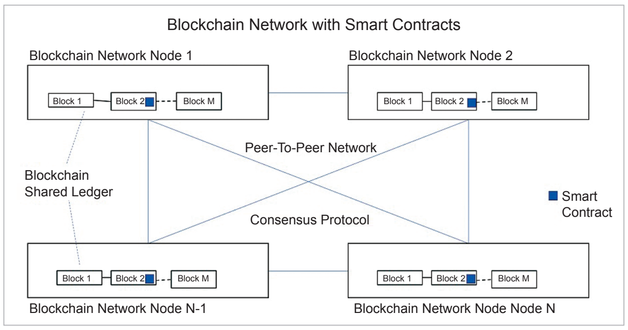

Blockchain Network with Smart Contracts
Blockchain Network Node 1
Blockchain Network Node 2
Block 1
Block 2
Block M
Peer-To-Peer Network
Blockchain Shared Ledger
Smart Contract
Consensus Protocol
Blockchain Network Node N-1
Blockchain Network Node Node N

Figure 1: Blockchain with smart contract

ineffective due to political/commercial reasons. Blockchain and IoT technologies together provide a solution which can do away with the intermediaries and autonomously track, control and manage resources in a trusted way.

Blockchain

The blockchain is a peer-to-peer network. It is based on a shared ledger (the blockchain database) and a protocol to keep the shared ledger updated. It is the technology underlying Bitcoin, the famous crypto-currency. Bitcoin was created by an unknown person nicknamed Satoshi Nakamoto. For people to accept and adopt Bitcoin as a crypto-currency, they had to be able to trust it. But, there is no government backing this crypto-currency. So, the trust has to be provided by the Bitcoin network itself.

The technology that provides such trust for Bitcoin is the blockchain. It is a shared ledger which is available on each full node of the network. In a Blockchain, each block is analogous to a ledger page and the blockchain is analogous to a ledger book. When a Bitcoin transaction happens on the network, the miner nodes verify the validity of the transaction, arrive at a consensus and update their copy of the ledger. The transactions verified and entered in the blockchain are immutable. Since blockchain is a peer-to-peer network, it does not have a central server. It has only peer level client applications, which join the blockchain network as nodes.

A blockchain provides some key benefits, which can be leveraged to provide solutions for use cases such as the ones discussed earlier in this article. Since it is a shared ledger on a peer-to-peer network, it does not have any central server which can be shut down unilaterally by any one party or authority. The records, transactions and smart contracts updated in the blockchain are immutable. They cannot be modified. The shared ledger is updated based on a consensus protocol and hence it prevents users or those transacting with Bitcoins from making fraudulent entries in the blockchain. This provides the trust needed when parties unknown to each other have to transact without an intermediary. Here, the blockchain network provides the trust.

There are a few open source platforms such as Ethereum, Hyperledger and Stellar, using which we can create a blockchain network as a public network, whereby any one can join the network. Alternatively, it can also be a private network behind the firewall of an organisation so that only those within the organisation can join it; or as a permissioned blockchain, where only those who are permitted to join the blockchain can do so.

Smart contracts

The features of Blockchain, such as decentralisation, immutability, verification, consensus and not being able to shut it down, provide a trustable network which can replace any intermediary. Soon, many people realised that the blockchain has the potential to be used for applications beyond crypto-currency such as Bitcoin. It can be used as a platform to execute or enforce contracts between unknown and hence untrusted parties in the form of smart contracts. But this needed a Turing Complete blockchain platform so that complex business logic could be written as code. The Bitcoin blockchain is not Turing Complete by design to avoid hacks. Hence, smart contracts cannot be written on the Bitcoin blockchain. This limitation gave rise to other blockchain platforms which can be used to write smart contracts, such as Ethereum, Hyperledger, etc.

Smart contracts are posted on the blockchain network in the same way we send crypto-currency. The posted contracts are added to the shared ledger (blockchain database). Smart contracts implement complex business logic as code. Each smart contract will have an address to which messages can be posted. A smart contract executes upon getting the messages posted to its address.

A client application connected to the blockchain network can receive messages from real-world assets such as IoT devices and post them to the smart contract. On getting the message, the smart contract executes and sends the result back to the client application in the form of an asynchronous event. This event can be used to control/manage the real-world assets.

In a way, smart contracts are analogous to the standing instructions given to banks to perform tasks such as transferring money to another account on specific dates, paying utility bills, etc. But, smart contracts implement more complex business logic and leverage the properties of the blockchain such as decentralisation, immutability and trust.

IoT

IoT (Internet of Things) is a network of things in the physical world. These things may be devices which have sensors within them or attached to them. In the context of IoT, a 'thing' is a server which is capable of getting sensor data from the device and sending it to the backend applications. A 'thing' server can be anything – a car, a refrigerator, a machine, a surveillance camera, a fan, a light, etc. IoT uses a local gateway server to connect the various thing servers in a building, car, etc, to the backend application. Thing servers use various communication technologies such as RFID, NFC, Wi-Fi, Bluetooth, and

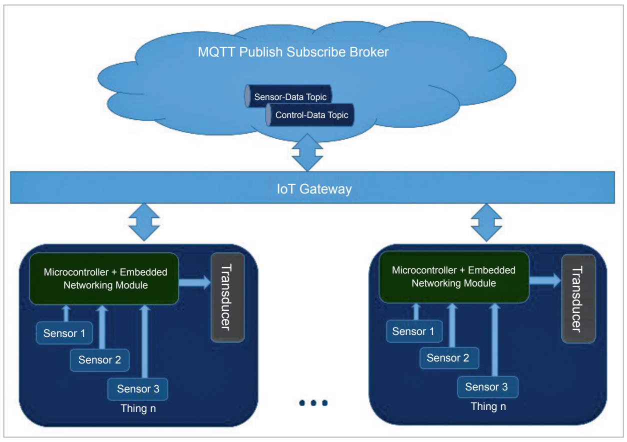


Figure 2: IoT thing server architecture

ZigBee to send data to the gateway server. Thing servers can also have wide area connectivity such as GSM, GPRS, 3G and LTE. On top of this communication infrastructure, IoT uses the MQTT protocol to connect the thing servers with the backend applications.

MQTT messaging protocol

The IoT thing servers may generate various types of data. Different backend applications might be interested in different types of data. The data exchange between the thing servers and the backend applications should happen asynchronously. There should not be any dependency on both of them to be alive at the same time to send and receive data.

To address this kind of a need, a publish/subscribe broker can be used in the architecture. HiveMQ is an open source publish/subscribe broker which uses the MQTT protocol to talk to the IoT thing servers on the one hand and to the backend applications on the other. The MQTT protocol works on the basis of topics. IoT thing servers can register topics with the HiveMQ broker. They will publish sensor data as messages on those topics. The backend servers will subscribe to those messages.

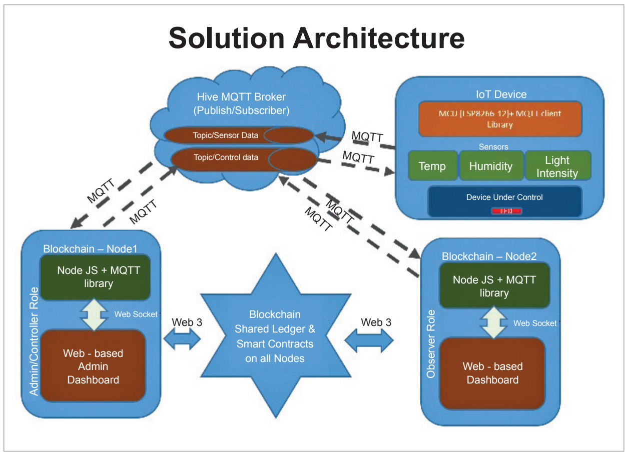


Figure 3: IoT-blockchain PoC architecture

Once an IoT thing server publishes a message on a specific topic, the HiveMQ broker sends that message to all the backend servers which have subscribed to that topic. Similarly, if the backend servers want to send any control information to the IoT thing servers, they can register the control topics with the HiveMQ broker and the IoT thing servers can subscribe to those topics. This will establish a two-way communication between the IoT things and the backend servers using the MQTT protocol and the HiveMQ broker.

PoC

We did a Proof of Concept (PoC) implementation using IoT, blockchain and MQTT to control a transducer. The building blocks of the PoC, and the open source technologies with which they were implemented, are explained below:

- A cloud based HiveMQ server was used as the MQTT publish/subscribe broker. On this broker, the IoT thing server registers a topic on which it publishes the sensor data and the blockchain node registers a topic on which it publishes the control data. In addition, the IoT thing server subscribes to receive the control data while the blockchain node subscribes to receive the sensor data. The HiveMQ broker delivers the published messages to subscribers.
- ESP8266-12 was used as the IoT thing server. This had a temperature, humidity and light sensor. It registered the sensor data topic with the cloud based HiveMQ MQTT broker and published the sensor data on this topic. It subscribed to the control messages from the blockchain node.
- A Node.js Web application, which subscribes to the sensor data topic on the HiveMQ broker for getting the sensor data, was used. It registers the topic on which it publishes the control messages on the HiveMQ broker. Subsequently, it publishes the control messages received from the blockchain. This Web application was connected to an Ethereum blockchain using the web3 JavaScript library. It registers the smart contract on the blockchain. It also posts the sensor data to the smart contract and watches for the asynchronous control messages from the smart contract. The control messages received from the smart contract are published through the HiveMQ broker.
- The smart contract was written using the Solidity language and compiled using a Solc compiler. The compiled smart contract is posted to the Ethereum blockchain using the web3 JavaScript library from the blockchain node.

Logistics use case

Let us again look at the logistics use case mentioned at the beginning of this article and see how the solution can be implemented. The perishable items are packed in a container in which a thing server with the temperature/pressure sensors is fitted. The sensor data is published through the MQTT broker.

Continued on page 81...

The various transport operators together could form a consortium and start a blockchain network. In this case, the blockchain is a permissioned one, where the consortium controls who can join its blockchain. Only members of the consortium may be allowed to join the blockchain network. Alternatively, the public Ethereum blockchain network can be used and the smart contract can be deployed on it.

The transport operator may run a Web application on his node with an easy-to-use user interface to create the smart contract on the blockchain. The sender gets into a contract with the transport operator. The terms and conditions of the transportation and payment are coded as a smart contract and deployed on the blockchain. The payment may be locked up in an escrow account.

The sensor data from the shipped container is received by the blockchain nodes and posted to the smart contract, which verifies the recorded temperature/pressure parameters as per the codified terms of the contract. Upon successful delivery of the item, the smart contract will trigger the payment from the escrow account. The payment will be completed in near real-time.

In future, micro payments between machines and M2M (machine-to-machine) communication without human intervention will find wide application. Today's centralised client-server world is being augmented with decentralised, peer-to-peer, disintermediated digital solutions. Blockchains with smart contracts and IoT are the evolving technologies which will drive such an exciting world.

Acknowledgements

The authors would like to acknowledge and thank Sriram Rajagopalan and Sarav Daman of Wipro Limited for their critical review of the article and contribution during the implementation of the PoC. END

References

[1] *https://www.ethereum.org/*
[2] *https://www.hyperledger.org/*
[3] *https://www.stellar.org/*
[4] *http://mqtt.org/*
[5] *http://www.hivemq.com/demos/websocket-client5.*
[6] *https://en.wikipedia.org/wiki/Internet_of_things*
[7] *http://solidity.readthedocs.io/en/develop/*
[8] *https://www.npmjs.com/package/solc*
[9] *https://slock.it/*

By: Venkatachalam Subramanian and Sumanta Basu

Venkatachalam Subramanian is a principal consultant in talent transformation, Wipro Limited, Bengaluru. He has 20 years of experience in the IT industry.

Sumanta Basu is a senior architect in the communications vertical, Wipro Limited, Bengaluru. He has more than 12 years of experience in the IT industry.