Materials Today: Proceedings xxx (xxxx) xxx

Contents lists available at ScienceDirect

Materials Today: Proceedings

journal homepage: www.elsevier.com/locate/matpr

# Integration of blockchain and IoT for data storage and management

Vijay Anant Athavale [a,*], Ankit Bansal [b], Sunanda Nalajala [c], Sagaya Aurelia [d]

[a] Panipat Institute of Engineering and Technology, 70 Milestone, GT Road, Samalkha, Haryana, India
[b] Department of Computer Science & Engineering, Gulzar Group of Institutes, Ludhiana, Punjab, India
[c] Computer Science & Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India
[d] Department of Computer Science, CHRIST (Deemed to be University), Bangalore, India

ARTICLE INFO

*Article history:*
Received 18 September 2020
Accepted 24 September 2020
Available online xxxx

*Keywords:*
Blockchain
HyperLedger
IoT
Data storage

ABSTRACT

Blockchain is a fast-evolving, encrypted, immutable and decentralized technology. It is used to store different data types such as transactions, at its core. But it's not about the cryptomonetary transactions. It can also be used to store many other items, such as properties, IoT or even multimedia content, such as songs, photos and videos. On HyperLedger Fabric Blockchain, we are managing IoT info. We collect and securely send data from IoT sensors to our HyperLedger Blockchain node using the MQTT protocol. We process the data and link it to our ledger after collecting data from the sensor. We also assess our network's efficiency, taking into account different parameters such as timeout for batch, batch size and message count.

Selection and peer-review under responsibility of the scientific committee of the Emerging Trends in Materials Science, Technology and Engineering.

## 1. Introduction

There are various entities in Blockchain network like peer, orders, applications, administrators. Each of these entities needs to have a valid identity in the network. In HyperLedger each user has an X.509 certificate which serves as a valid identity for each entity. These certificates play a key role in determining the permission access to various resources in the network. A certificate is as trustworthy as the genuineness of the authority it is coming from. A Membership Service Provider (MSP) is used for specifying the rules which identify the valid identities for the various entities.

- Public Key Infrastructure PKI
- Digital Certificates
- Public Keys and Private keys
- Certificate Authority
- PKI Vs MSP

The HyperLedger configuration is the heart of the Blockchain network. We need to configure various entities peers, organizations, policies, channels and a lot more. Most of the HyperLedger configurations goes into the yml or yaml files.

HyperLedger provides the following binaries for generating various types of configurations:

- Configtxgen: This toll is used for generating the first block of the ledger also known as the genesis block. This block is necessary bootstrapping the order as well as the channel configurations.
- Cryptogen: It is used for generating the crypto material for various entities of the organization.

We use docker containers [KV17] for configuring various entities in the Blockchain Network. For our Blockchain network we have containers for the following entities:

- Peer containers
- Fabric CA (Certificate Authority) containers
- cli container to access all the peers in the network.
- Ordered container
- CouchDB containers

All these containers are a part of a single docker network. Even the chain code is executed in a separate container from the endorsing peer process.

HyperLedger provides a docker image of CouchDB that should run on the same server as that of the peer. We need a CouchDB container per each running peer. CouchDB is an optional

* Corresponding author.
*E-mail address:* vijay.athavale@gmail.com (V.A. Athavale).

https://doi.org/10.1016/j.matpr.2020.09.643
2214-7853/
Selection and peer-review under responsibility of the scientific committee of the Emerging Trends in Materials Science, Technology and Engineering.

alternative to levelDB as state database which supports rich queries as it allows chaincode values to be modeled as JSON. It allows us to query actual data content rather than just keys. We can't switch a peer using levelDB to use CouchDB due to data compatibility issues.

## 2. Block structure and blockchain structure

A HyperLedger Blockchain record or ledger comprises (Fig. 1) of two particular yet related components:

- A world state
- A Blockchain

### 2.1. World state

World state is a database that stores the current values of various states of the Ledger. Current values of these states can be easily read from the World state which avoids traversing the entire transaction log and thus saves time. Record or ledger states are represented as key-value pairs which are updated, created and erased frequently.

### 2.2. Blockchain

It is a log of all the transactions that lead to the world state. These transactions are organized into blocks which are linked together to form the Blockchain enabling us to have a record of all the history that led to the current world state. Unlike world state, it contains immutable blocks which contain a set of ordered transactions.

### 2.3. Blocks

Hyperledger Block (Fig. 2) composed of three parts:
The Block header H2 of block B2 contains the current blocks data D2′s hash, the current block number which is 2, and a copy of previous blocks hash PH1 whose block number is1.

- Block Header: Block Header consists of three fields written when creating a block.
- Block number: Block number is an integer which starts at 0 (For Genesis Block) and is increased by 1 for each new block that is attached to the Blockchain.
- Previous Block Hash: This contains the hash value of the previous block in the Blockchain which is essential for maintaining the links between blocks.

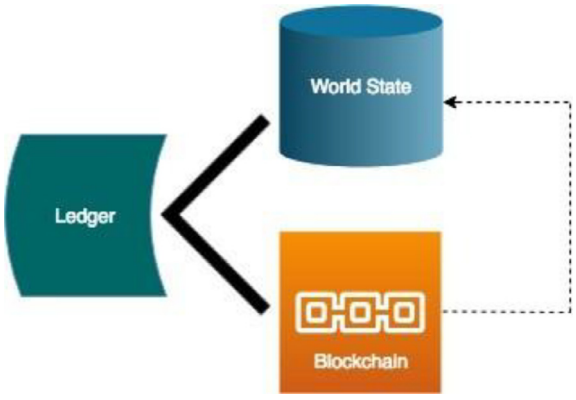


**Fig. 1.** Ledger comprises of blockchain and world state.

- Current Block Hash: This contains the hash value of all the transactions that are present in the current block
- Block Metadata: Block Metadata contains the time the block was written, public key, signature and the certificate of block writer. Thereafter, the block committer also adds a flag which tells whether a transaction is valid/ invalid. This information about the valid/invalid transaction is not used in the hash as the hash gets created when creating the block and this flag is added when writing to the ledger.
- Block Data: This section contains a list of ordered transactions. These transactions are added to block data when the block is created. These transactions have a detailed but simple and understandable structure.

### 2.4. Transactions

Changes to the World State are captured by Transactions. The block data structure of the Blocks contains the transactions as shown in Fig. 3.

The Transaction T4 inside block data D1 of block B1 contains a transaction signature S4, trans- action header H4, a transaction response R4, a transaction proposal P4, and a list of endorsements, E4. Various fields of Block data structure containing transactions is as follows:

- **Header**: The header which is represented by H4 which contains important metadata about the trans- action for example, the name and version of the appropriate chaincode.
- **Proposal**: The proposal which is represented by P4 which includes the input parameters provided to the chaincode by an application creating the proposed update of the ledger. This proposal provides a set of input parameters when the chaincode runs which, together with the current world state, determines the new world state.
- **Endorsements**: The endorsement is represented in E4, this is a list of each organization's transaction responses which are signed and that are sufficient to meet the endorsement policy. Only one transaction response is included in the transaction although there may be multiple endorsements. This is because each endorsement effectively encodes the particular transaction response from its organization which means that including any transaction response that does not match sufficient endorsement would be a waste of resource as it will get rejected and marked invalid and would not update the world state of the ledger
- **Signature**: This contains a cryptographic signature which is created by the application of the client. It is used to check that the details of the transaction were not modified as it requires the private key of the application to generate it.
- **Response**: This section captures the world state's before and after values as a Read-Write (RW-set) set. It is a chaincode output, and if the transaction is validated successfully, updating the world state will be applied to the ledger.

## 3. Implementation of proposed work

Our overall system architecture can be subdivided into two parts.

- IoT System
- Blockchain System

### 3.1. Blockchain system setup

Our Hyperledger Blockchain setup (Fig. 4) consists of two organizations namely org1 and org2 and each organizations having two

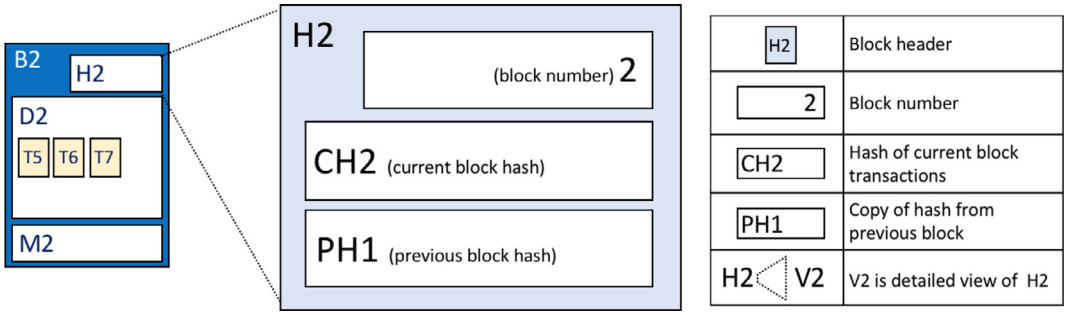


**Fig. 2.** HyperLedger block Structure.

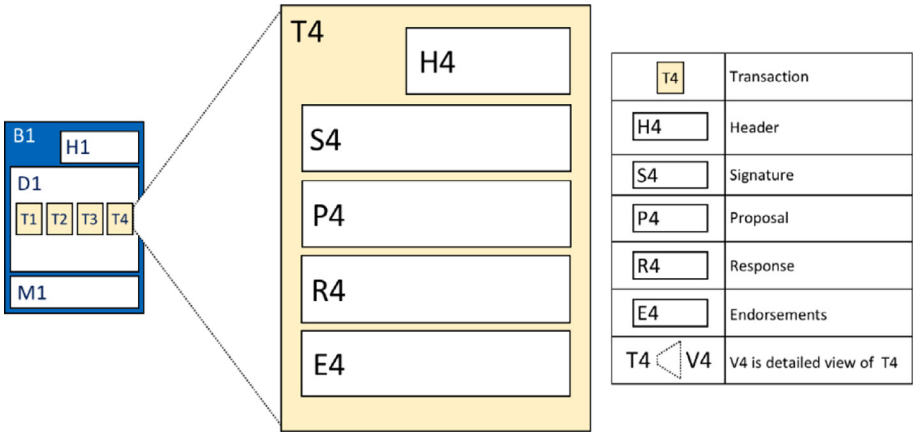


**Fig. 3.** Transaction block.

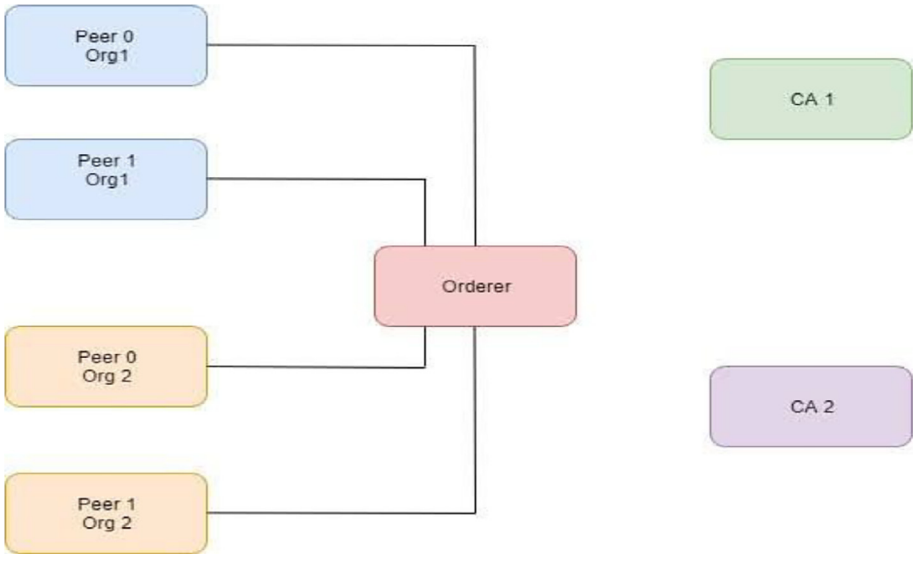


**Fig. 4.** HyperLedger network setup.

peers each. We also have a Certificate Authority for each organization named CA1 and CA2. Our peers are named as peer 0 and peer 1. The domain org1 and org2 are used to differentiate between the peers of different organizations.

### *3.2. IOT system setup*

Our IoT system (Fig. 5) consists of the DHT 11 sensor connected to the raspberry pi via a breadboard. The raspberry pi has the MQTT broker running on it. We also have a Linux machine running the MQTT Client and our HyperLedger Blockchain Network.

### *3.3. Performance comparison under different transaction load*

The parameters that we have taken into consideration for analysis are as follows:

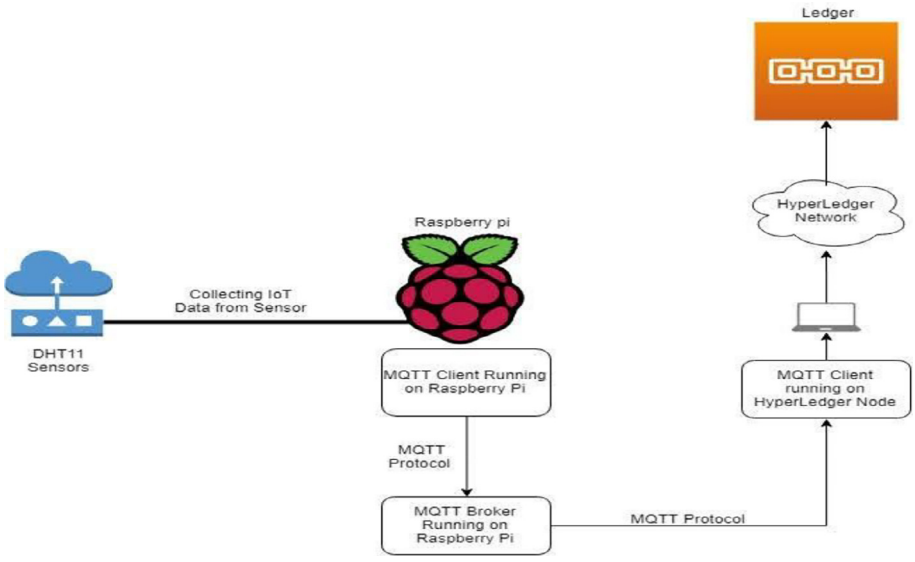


**Fig. 5.** IoT system setup.

- Batch Timeout: This is the amount of time the orderer waits before creating a batch.
- BatchSize:

Max Message Count: This is the maximum number of messages a batch can contain.

Absolute Max Bytes: This is the maximum size in bytes of messages that can be allowed in a batch.

Preferred Max Bytes: The preferred maximum number of bytes allowed for the serialized messages in a batch. A message larger than the preferred max bytes will result in a batch larger than preferred max bytes.

## 4. Analysis and results

### *4.1. Transaction latency*

Transaction latency is the time difference between when the transaction gets submitted and the time the transaction gets committed across the network, confirming the transaction. Unlike lottery-based consensus protocols like Bitcoin or Ethereum where the finality of the transaction is determined using a probabilistic approach, Fabric's consensus process leads to a deterministic finality approach. The time to start the fabric network with all the docker containers ranges between 28 and 51 s.

### *4.2. Performance comparison under different transaction load*

Table 1.

**Table 1**
Number of records vs. delay in writing.

| Number of records | Delay in writing (ms) |
|---|---|
| 50 | 155 |
| 100 | 246 |
| 250 | 612 |
| 500 | 1037 |
| 750 | 1469 |
| 1000 | 2394 |
| 1250 | 4127 |
| 1500 | 4681 |

The start-up execution time for the below configuration was 31 s.

- BatchTimeout: 2 s
- BatchSize:

Max Message Count: 10
Absolute Max Bytes: 99 Mega Bytes
Preferred Max Bytes: 512 Kilo Bytes
Fig. 6. Fig. 7. Table 2.

The start-up execution time for the below configuration was 44 s.

- BatchTimeout: 4 s
- BatchSize:

Max Message Count: 20
Absolute Max Bytes: 99 Mega Bytes
Preferred Max Bytes: 512 Kilo Bytes
Fig. 8. Fig. 9. Table 3.

The start-up execution time for the below configuration was 34 s.


**Fig. 6.** Number of records vs. delay in writing.

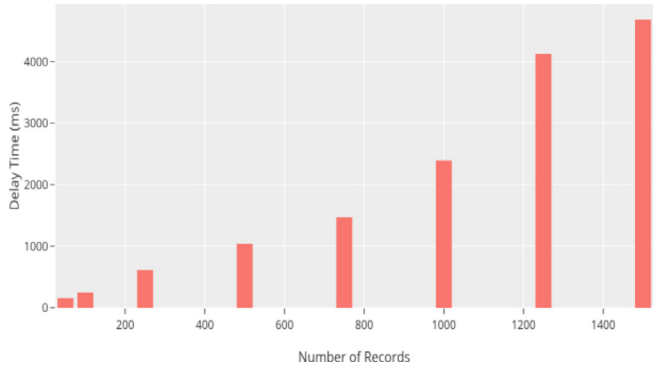


**Fig. 7.** Number of records vs. delay time.

**Table 2**
Number of records vs. delay in writing.

| Number of records | Delay in writing (ms) |
|---|---|
| 50 | 84 |
| 100 | 169 |
| 250 | 369 |
| 500 | 785 |
| 750 | 1157 |
| 1000 | 1787 |

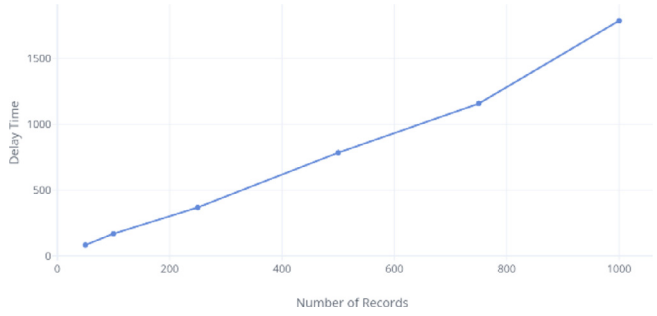


**Fig. 8.** Number of records vs. delay in writing.

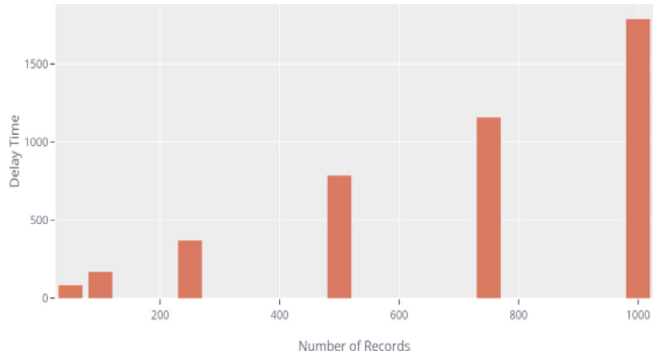


**Fig. 9.** Number of records vs. delay time.

**Table 3**
Number of records vs. delay in writing.

| Number of records | Delay in writing (ms) |
|---|---|
| 50 | 111 |
| 100 | 172 |
| 250 | 432 |
| 500 | 772 |
| 750 | 1287 |

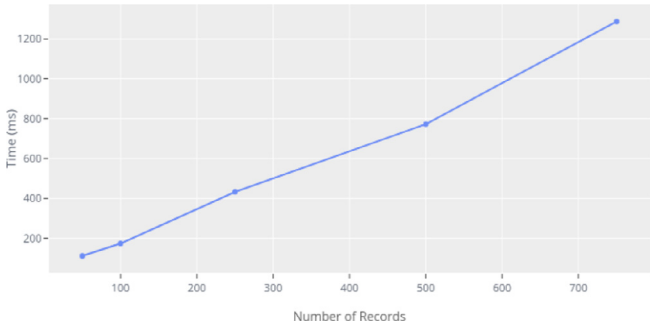


**Fig. 10.** Number of records vs. delay in writing.

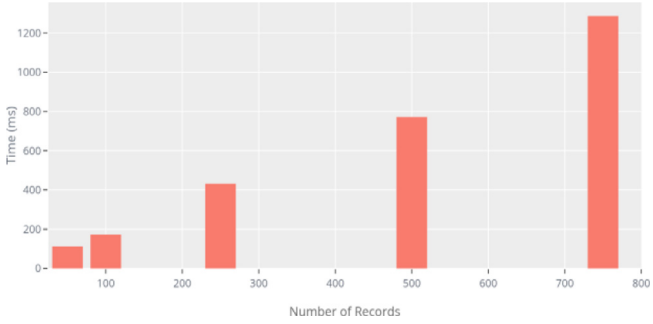


**Fig. 11.** Number of records vs. delay time.

- BatchTimeout: 1 s
- BatchSize:

  Max Message Count: 5
  Absolute Max Bytes: 99 Mega Bytes
  Preferred Max Bytes: 512 Kilo Bytes

Fig. 10. Fig. 11.

Comparing the performance of All 3 parameters sets: Fig. 12.

As we can see from the above graphs that the time for writing increases as the number of records or the transactions increases. We also see that the performance varies as test our network with various parameters.

We can see the performance for parameter set 1 is better than the parameter set 2 and 3. For parameter set 1, we are using a batch timeout of 2 s and max message count of 10. This means that a block would be added to the ledger if either the number of messages has reached 10 or the time of 2 s is passed whichever is earlier.

For parameter set 2, we use a batch timeout of 4 sec and max message count of 20. So increasing the batch timeout and max message count does not necessarily increase the performance. In fact not only increasing but decreasing the batch timeout and max count value doesn't necessarily improve the performance as can we can see in the graph, the performance of set 2 and 3 is very similar where we have used batch time out of 1 sec and max message count of 5. We need to maintain a balance between the batch timeout and the max message count for the optimal performance.

Another observation is that our orderer is able to handle a maximum number of transactions for the parameter set 1 followed by the parameter set 2 and parameter set 3. This reason for this is that for set 3 we are using max message count of 5 which is the lowest for the 3 sets. This

means that if a load of transactions is more than the batch timeout will become less relevant as the messages received by orderer

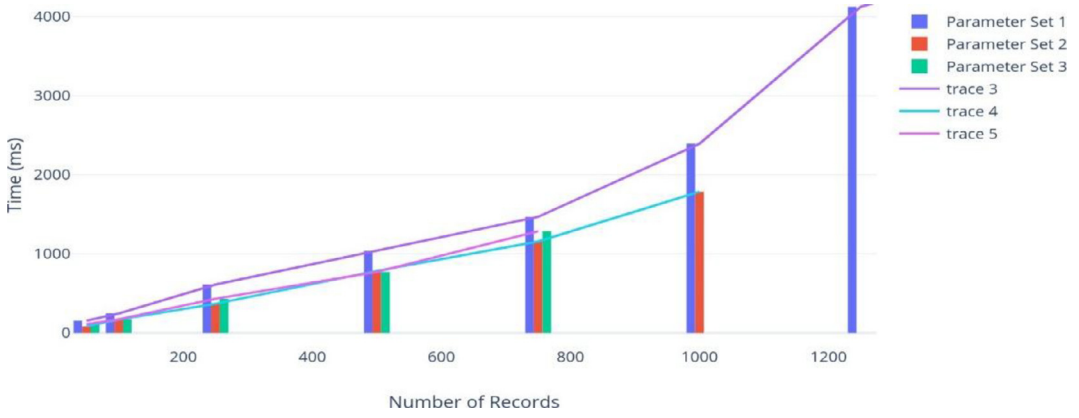


**Fig. 12.** Number of records vs. delay time.

is much before the batch timeout. But since the orderer takes some time to create a block and then send it to peers some transactions may get timeout as the orderer is not able to handle it within the required time frame.

We can conclude that batch timeout, max message count, absolute max bytes, preferred max bytes play a key role in the performance of the network. Increasing one parameter does not necessarily mean improvement in performance. A balance is to be maintained between these values depending on the load of transactions expected.

## 5. Conclusion

Our IoT Blockchain implementation works successfully which allows us to store IoT data which is trustworthy and verified. Our network works well for a decent number of transactions but having just one orderer limits its performance in terms of the number of transaction it can handle. On Average, our solo configuration orderer is able to handle about 750–1500 transactions depending on various parameters like the batch timeout, batch size and other.

We conclude that HyperLedger fabric platform can be used for managing IoT data and the inbuild endorsements, encryption and signing ensures the security, validity, and integrity of the data. All the transactions in HyperLedger are TLS encrypted which ensures that data cannot be compromised during the internal network communications.

## CRediT authorship contribution statement

**Vijay Anant Athavale:** Conceptualization, Methodology, Software, Data curation. **Ankit Bansal:** Writing - original draft, Validation, Visualization. **Sunanda Nalajala:** Investigation, Supervision, Software. **Sagaya Aurelia:** Supervision, Software, Writing - review & editing.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Further Reading

[1] Pierre-Louis Aublin, Sonia Ben Mokhtar, and Vivien Qúema. Rbft: Redundant byzantine fault tolerance. In 2013 IEEE 33rd International Conference on Distributed Computing Systems, pages 297–306. IEEE, 2013.
[2] Richard Gendal Brown, James Carlyle, Ian Grigg, and Mike Hearn. Corda: an intro- duction. R3 CEV, August, 2016.
[3] Dht11 & dht22 sensors temperature and humidity tutorial using arduino - https://howtomechatronics.com/tutorials/arduino/dht11-dht22-sensors-tem perature- and-humidity-tutorial-using-Arduino/.
[4] P.P. Fathima Dheena, Greema S. Raj, Gopika Dutt, S. Vinila Jinny. IoT based smart street light management system. In 2017 IEEE International Conference on Circuits and Systems (ICCS), pages 368–371. IEEE, 2017.
[5] Dijiang Huang, Chun-Jen Chung, Qiuxiang Dong, Jim Luo, Myong Kang. Building private blockchains over public blockchains (pop): an attribute-based access control approach. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pages 355–363. ACM, 2019.
[6] Urs Hunkeler, Hong Linh Truong, Andy Stanford-Clark. Mqttsa publish/subscribe protocol for wireless sensor networks. In 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08), pages 791–798. IEEE, 2008.
[7] Kaivan Karimi, Gary Atkinson. What the internet of things (IoT) needs to become a reality. White Paper, FreeScale and ARM, pages 1–16, 2013.
[8] Chris Khan, Antony Lewis, Emily Rutland, Clemens Wan, Kevin Rutter, Clark Thompson, A distributed-ledger consortium model for collaborative innovation, Computer 50 (9) (2017) 29–37.
[9] Kitti Klinbua, Wiwat Vatanawood. Translating tosca into docker-compose yaml file using antlr. In 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), pages 145–148. IEEE, 2017.
[10] Paul Le Noac'H, Alexandru Costan, Luc Bouge. A performance evaluation of apache kafka in support of big data streaming applications. In 2017 IEEE International Conference on Big Data (Big Data), pages 4803–4806. IEEE, 2017.
[11] Ermin Sakic, Wolfgang Kellerer, Response time and availability study of raft consensus in distributed sdn control plane, IEEE Trans. Netw. Service Manage. 15 (1) (2017) 304–318.
[12] Harish Sukhwani, Joše M Martínez, Xiaolin Chang, Kishor S Trivedi, and Andy Rindos. Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric). In 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS), pages 253–255. IEEE, 2017.