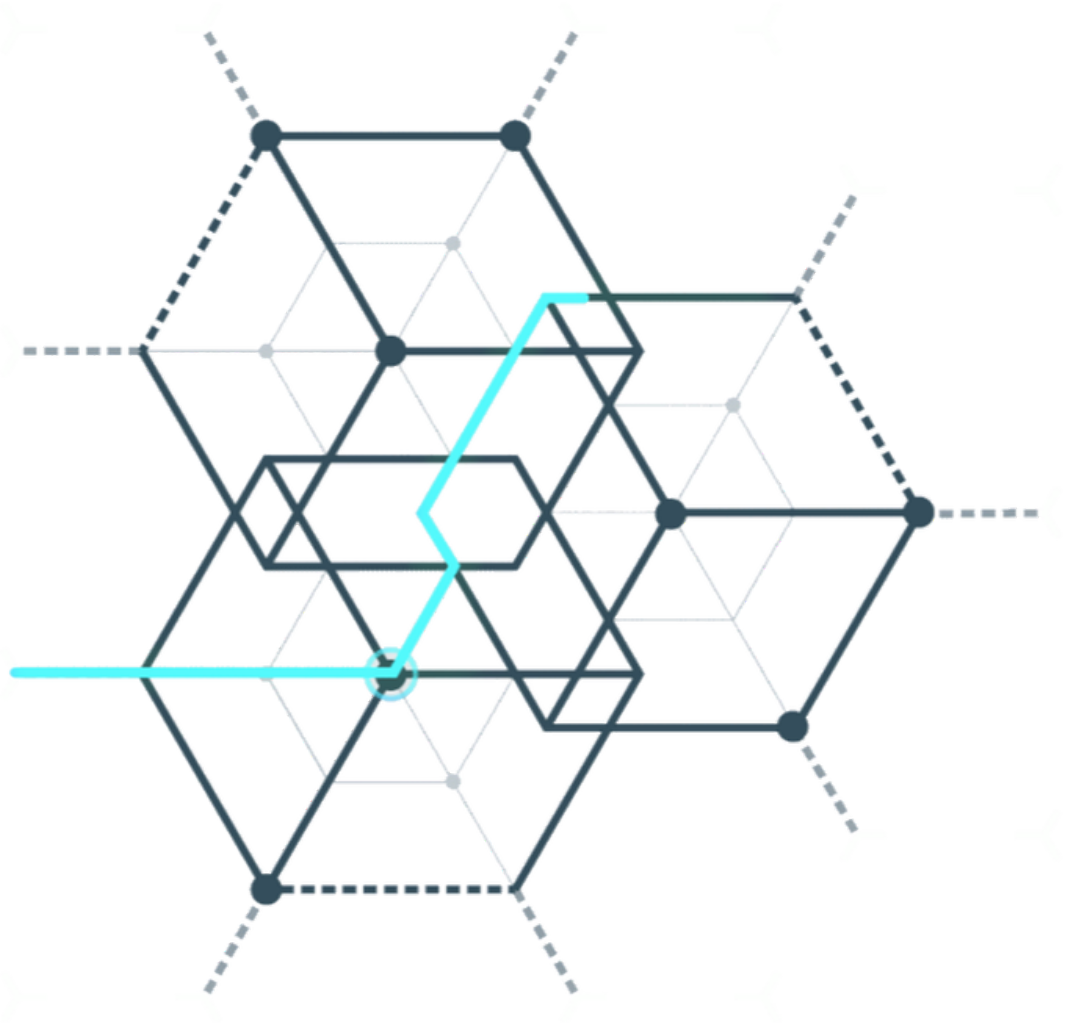
Blockchain Architecture



Team Finney

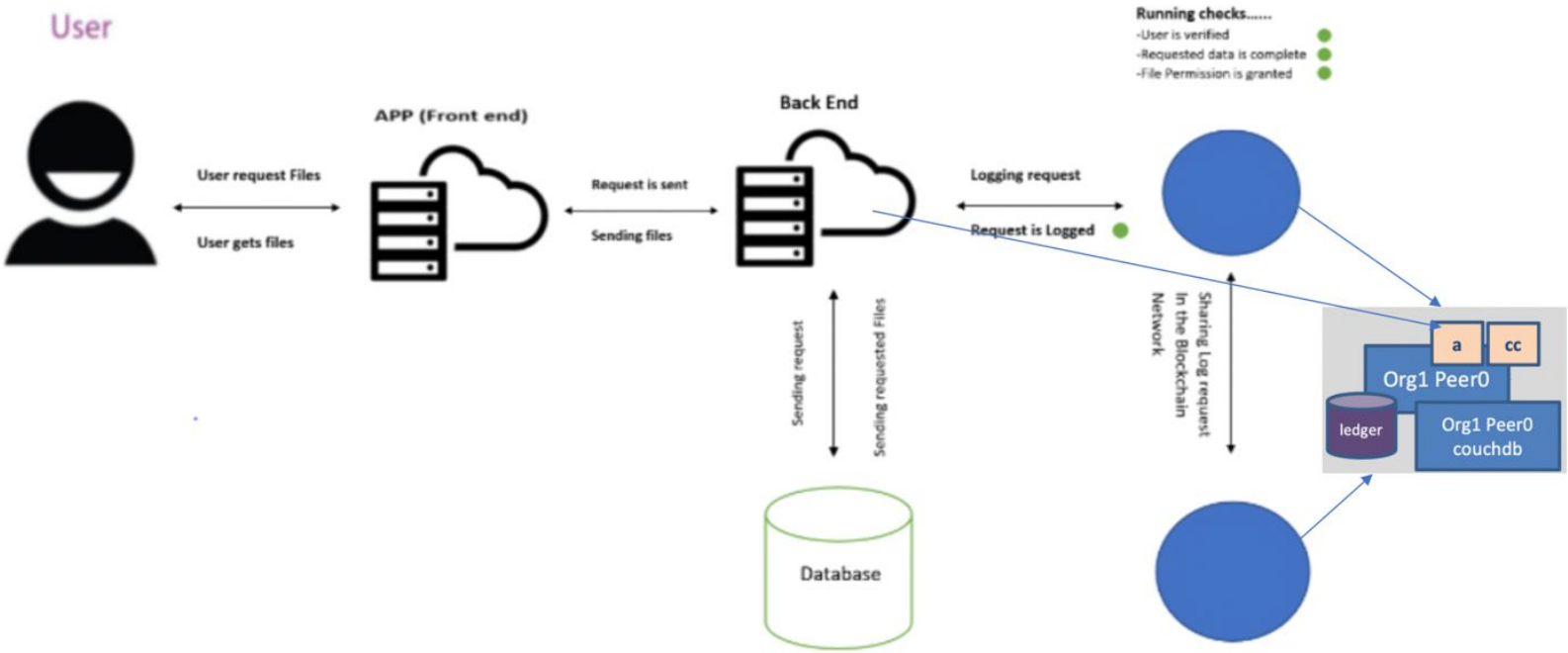
Jan Dorresteijn	500713066
Daan Frank	500780674
Rami Matte	500732896
Dylan Wesselink	500746936

05-03-2020
Hogeschool van Amsterdam

Table of contents

1. Data and transaction model

In this chapter, a brief explanation over the architecture of the Hyperledger network will be done based on a use case and a diagram which represents the interactions and transactions that are taking place between all network participants and assets.



This overview represents both client and user's interaction with the blockchain network. Clients has to become part of the hyper ledger in order to interact with the network. By creating user accounts and assigning a private and a public key, users will be able to participate and make transactions to the hyper ledger network. In addition multiple clients (hosts) could also be a part of the network by contributing as a node to make the blockchain network faster and more secure.
In order to get a better understanding of how this interaction with the network happens, a simple use case will be used:

To use the network, users must log into the app interface using credentials that are provided by the administrator and then submit a request for viewing a file. The backend will communicate through an API or a hook with the main node to check whether this file exists and meets the permission requirements so it can be sent back. If everything's checks out,

user activity will be logged and submitted to the network ledger. Data files will be sent back encrypted to the user interface where it gets decrypted and displayed to the user.

This table below shows the participants, assets and transactions for each party that is contributing to the network.

Participants	**Assets**	**Transactions**
Clients (main hosts)	Blockchain network-Servers-databases-ledger	Grant permission- log activities- monitor activities- establishing connection with the blockchain network- encrypt documents content- create document hash (title)- establishing consensus with other parties on the network.
Network users	Processing power-authority-ledger	Upload documents-edit documents and share documents across the network

2. Logic (Smart Contracts)

2.1 Business rules

For our use case we have set up a couple of business rules;

- The user needs to be connected to the network to participate
- The user needs to be logged into the network
- The user needs to have the right permissions to create/read files
- When a file is created, it needs to be logged on the blockchain
- When a file is read, it needs to be logged on the blockchain

2.2 Events

We have implemented the CRUD functionality into our application; Create, Read, Update and Delete. We have only used create and read because this goes into the main goal of our use case. It should be ensured that it is only possible to create and read. It should never be possible to edit or delete files. When a file is uploaded a log is created on the blockchain, this is the same for reading a file but then the log is updated.

2.3 Privacy and Security

In this chapter privacy and security aspects will be covered, think about aspects like accessing the network, how is data processed on the network and how to become a node on the hyperledger network

2.4 Integration

This application is running fully on Hyperledger network, meaning its fully decentralized and the central authority is equally distributed between all nodes.
To become a node or to take apart of this network as a host, connecting own CMS or sharepoint is a pre because this application will hock to the backend of an excited system and run a terminal between the hyperledger and the CMS. After becoming node it is possible to join users to the network by assigning them private and a public key by the administrator.

each node has a ledger. A ledger is basically a database that keeps track of all network activities when transactions are submitted by users. This distributed ledger ensure network immutability because it considers to be the single point of truth and it allows parties to trust each Other without the need to know each other.

More nodes means more security because the central authority will be for example distributed among 10 node instead of 3 which means if two nodes are inline for a faulty transaction then this transaction will be considered to be true. In the case of 10 nodes it will be much more difficult and it also has more advantages like faster transaction because the network bandwidth which will allow transactions to go much faster up to a certain point because then it will be slower due to the amount of nodes that have to reach consensus .

2.5 Security

Users must log in with credentials that are provided by administration in addition java web tokenization is implemented meaning double layers of protection and identification is required in order to use the network. The network is running on hyperledger so performing Ddos attack is not possible unless 51% of the network is controlled by 1 node.
In addition, acquiring data files is strictly encrypted both ways, security aspects are pushed back by the node (organization) it self because this application is not more than a hock that is running on existing cms system meaning the security of the application is totally depend on the node.

2.6 Privacy

Privacy in this sense not important because users data, logs, activities and data files are encrypted with two fish 256 bit encryption which is a very strong encryption mechanism. Its also GDPR approved because users data and files aren't stored on the network, only encrypted activity logs. Users are able to perform CRUD operations within the organization safe environment.

3. Architecture organisation

In this chapter the architecture, the amount of peers, and the general working of the consensus/endorsement in our network will be discussed. Due to the fact that this project is based upon the earlier made egg use case created by Marcio. In our case, we've set this project up to be related to our blockchain minor project. This project is about logging the activity of the users. The only difference was that this project has to be finished in Hyperledger Fabric. Our project was made using multiple different frameworks, so this was a new experience for us all.

Our systems works with multiple peers. Peers in multiple forms. Also in physical forms. This could be traced back into peers set up in to virtual machines.

We've used the BFT algorithm. This is the same algorithm used in the Bitcoin currency. The BFT (Byzantine Fault Tolerance) is: A **Byzantine fault** (also **interactive consistency**, **source congruency**, **error avalanche**, **Byzantine agreement problem**, **Byzantine generals problem**, and **Byzantine failure**[1]) is a condition of a computer system, particularly distributed computing systems, where components may fail and there is imperfect information on whether a component has failed. The term takes its name from an allegory, the "Byzantine Generals Problem",[2] developed to describe a situation in which, in order to avoid catastrophic failure of the system, the system's actors must agree on a concerted strategy, but some of these actors are unreliable.

In a Byzantine fault, a component such as a server can inconsistently appear both failed and functioning to failure-detection systems, presenting different symptoms to different observers. It is difficult for the other components to declare it failed and shut it out of the network, because they need to first reach a consensus regarding which component has failed in the first place.

Individual contribution table

Name	tasks	Evidences	Explanations
Jan	● Programming	Can be found in the repository	The programming is the main part of the assignment Jan took this part because he is the most experienced developer from our group
Dylan	● Documentation: Architecture organisation ● Repository setup	Can be found in the document	Dylan explained how many peers and organisations were used but also how the consensus/endorsement works in our network He also set up the repository with installation and usage instructions
Rami	● Documentation: Data and transaction model Privacy and security Integration	Can be found in the document	Rami explained how the participants assets and transactions are modelled. He also talked about the authentication and authorization. But he also explained how the blockchain will interact with external systems.
Daan	● Documentation: Logic (smart contracts) Network hosting ● Contribution table ● Document layout	Can be found in the document	Daan has explained what the business rules are, which types of events are generated and who consumes them. He also described the chosen hosting strategy. Daan also made the contribution table and the layout of the document