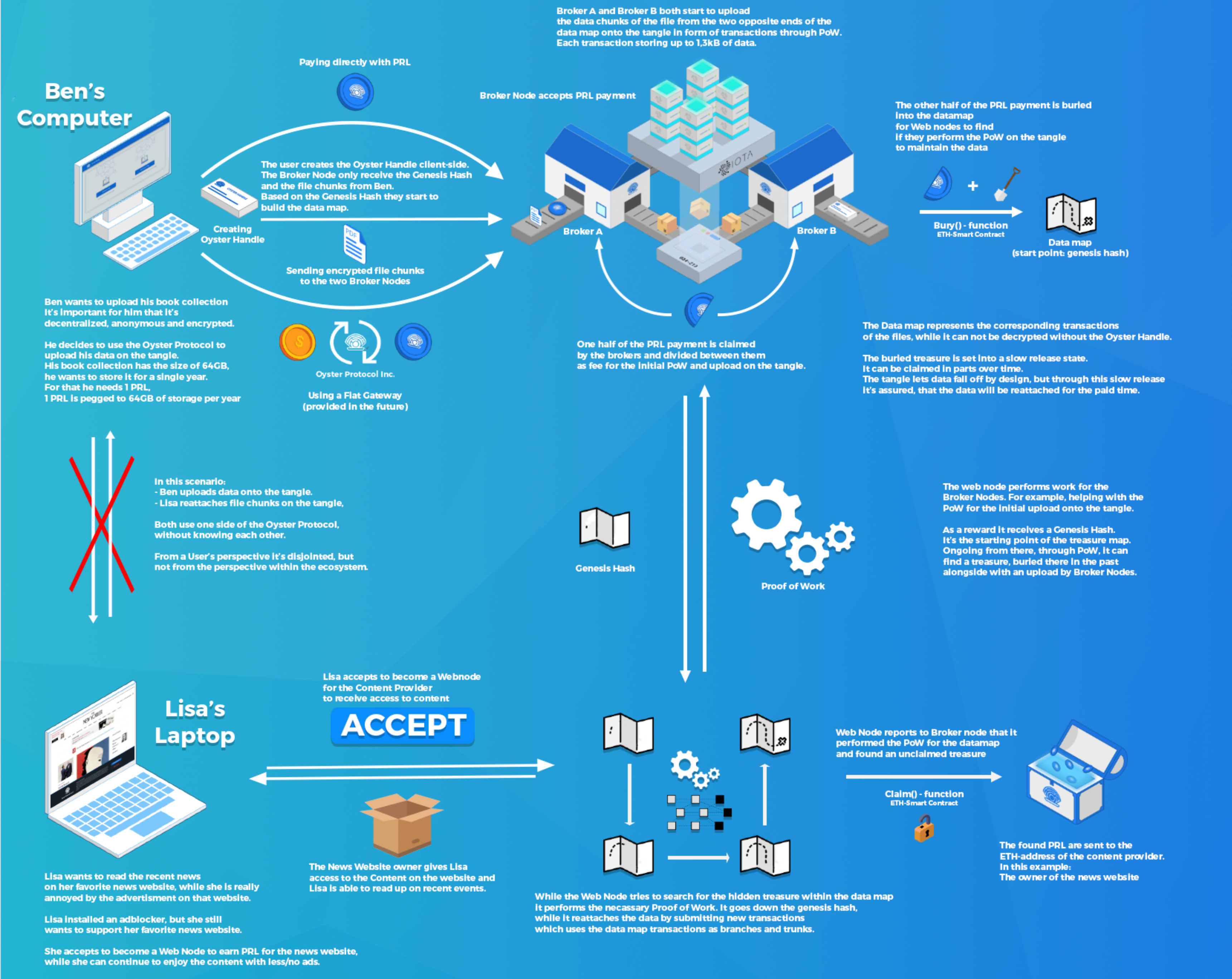
Oyster Protocol Simplified Ecosytem



Ben's Computer

Paying directly with PRL

Creating Oyster Handle

The user creates the Oyster Handle client-side.
The Broker Node only receive the Genesis Hash and the file chunks from Ben.
Based on the Genesis Hash they start to build the data map.

Sending encrypted file chunks to the two Broker Nodes

Ben wants to upload his book collection
It's important for him that it's decentralized, anonymous and encrypted.

He decides to use the Oyster Protocol to upload his data on the tangle.
His book collection has the size of 64GB,
he wants to store it for a single year.
For that he needs 1 PRL,
1 PRL is pegged to 64GB of storage per year

Oyster Protocol Inc.

Using a Fiat Gateway
(provided in the future)

Broker A and Broker B both start to upload the data chunks of the file from the two opposite ends of the data map onto the tangle in form of transactions through PoW.
Each transaction storing up to 1,3kB of data.

Broker Node accepts PRL payment

Broker A

Broker B

One half of the PRL payment is claimed by the brokers and divided between them as fee for the initial PoW and upload on the tangle.

The other half of the PRL payment is buried into the datamap
for Web nodes to find
if they perform the PoW on the tangle
to maintain the data

Bury() - function
ETH-Smart Contract

Data map
(start point: genesis hash)

The Data map represents the corresponding transactions of the files, while it can not be decrypted without the Oyster Handle.

The buried treasure is set into a slow release state.
It can be claimed in parts over time.
The tangle lets data fall off by design, but through this slow release it's assured, that the data will be reattached for the paid time.

In this scenario:
- Ben uploads data onto the tangle.
- Lisa reattaches file chunks on the tangle,

Both use one side of the Oyster Protocol, without knowing each other.

From a User's perspective it's disjointed, but not from the perspective within the ecosystem.

Genesis Hash

Proof of Work

The web node performs work for the Broker Nodes. For example, helping with the PoW for the initial upload onto the tangle.

As a reward it receives a Genesis Hash.
It's the starting point of the treasure map.
Ongoing from there, through PoW, it can find a treasure, buried there in the past alongside with an upload by Broker Nodes.

Lisa's Laptop

Lisa accepts to become a Webnode for the Content Provider to receive access to content

ACCEPT

Lisa wants to read the recent news on her favorite news website, while she is really annoyed by the advertisment on that website.

Lisa installed an adblocker, but she still wants to support her favorite news website.

She accepts to become a Web Node to earn PRL for the news website, while she can continue to enjoy the content with less/no ads.

The News Website owner gives Lisa access to the Content on the website and Lisa is able to read up on recent events.

While the Web Node tries to search for the hidden treasure within the data map it performs the necassary Proof of Work. It goes down the genesis hash, while it reattaches the data by submitting new transactions which uses the data map transactions as branches and trunks.

Web Node reports to Broker node that it performed the PoW for the datamap and found an unclaimed treasure

Claim() - function
ETH-Smart Contract

The found PRL are sent to the ETH-address of the content provider.
In this example:
The owner of the news website