Blockchain based Diagnostic Information Logger

Apnatva S. Rawat

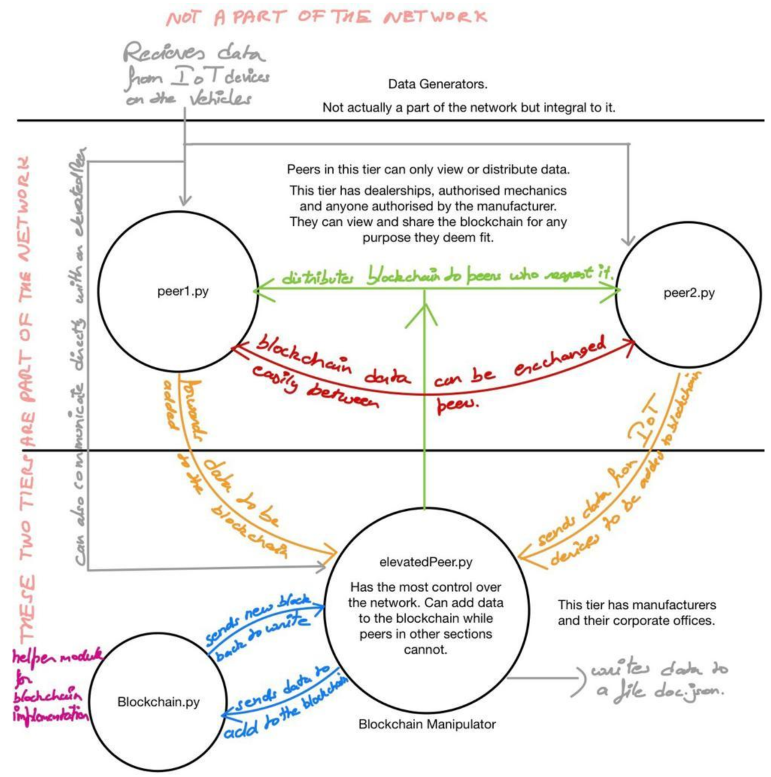
Use Case

With so much technology in our cars now, something is bound to fail. The more information you can have about a failure, the easier it can be to solve it. Sometimes you might need to recreate the situation exactly to obtain a similar error before you can even begin to solve it. To get past the lack of knowledge we have come up with a way to log information in a decentralized, immutable manner so that everyone can own their data without having to worry about the genuineness of the data.

This can help manufacturers and mechanics identify specific conditions or use cases where the onboard tech can fail and go into (un)recoverable states. It could log any diagnostic information that will help assess the internals of a car. This would provide a sort of a ‘traceback’ to being able to see the progress of events and determine if they were caused by internal/external matters.

With the advent of self-driving cars we can get even more insight into how the AI was formulating decisions which led to an undesirable situation. The continuous logging on a private decentralized server would mean that all of this data will be available to any authorized dealer/repairmen but not anyone else, increasing authentic repairs and fits. This provides manufacturers, mechanics, with more information than ever before which can be used for ML or Analytic purposes. The possibilities with this kind of information in a secure and synchronized manner are endless.

Logical Diagram


NOT A PART OF THE NETWORK
Recieves data from IoT devices on the vehicles
Data Generators.
Not actually a part of the network but integral to it.
THESE TWO TIERS ARE PART OF THE NETWORK
can also communicate directly with an elevatedPeer
Peers in this tier can only view or distribute data.
This tier has dealerships, authorised mechanics and anyone authorised by the manufacturer. They can view and share the blockchain for any purpose they deem fit.
peer1.py
peer2.py
distributes blockchain to peers who request it.
blockchain data can be exchanged easily between peers.
forwards data to be added to the blockchain
sends data from IoT devices to be added to blockchain
elevatedPeer.py
Has the most control over the network. Can add data to the blockchain while peers in other sections cannot.
This tier has manufacturers and their corporate offices.
Blockchain Manipulator
writes data to a file doc.json.
sends new block back to write
sends data to add to the blockchain
Blockchain.py
helper module for blockchain implementation

Functional Description

1. blockchain.py - this is the main module that implements the blockchain. It reads data from the requested file, parses it line by line into a list. Then it reads the previous block, gets the index and the hash to generate information for the current block. After calculating the proof for it it adds a timestamp and rides this information back to the "doc.json" file.
2. peer1.py - this is the script that any peer would be running. It enables every peer to open a port and broadcast its IP address so that any other peer can fulfill its requests and send data. It can receive the blockchain file or data directly from vehicles. Data transfer from the vehicles was not implemented as this project focuses on building a peer-to-peer blockchain network and here the vehicles are not a peer but rather an outside data generator.
3. peer2.py - exactly the same script as peer1. Made to demonstrate data transfer between two peers.
4. elevatedPeer.py - imports the blockchain module unlike peer1 and peer2. Can receive data from peers or directly from the vehicles. Converts this data into a block using the blockchain module. This updated blockchain is now ready to be transmitted to anyone on the network with an open port that is requesting an update.

Libraries Used:
os[1] - Python library to provide an interface for OS actions.
socket[2] - Python library to establish connections between hosts on specified ports.
threading[3] - Python library to support parallelism between tasks.
hashlib[4] - Python library to use SHA256 hashing functions.
getpass[5] - Python library to securely input password from the user.

Algorithms

Consensus Algorithm - Each blockchain requires a consensus algorithm and for this we have used Proof of Work[6] combined with Proof of Authority[7]. First, the scripts distributed among peers and elevated peers are different to prevent having access to the blockchain module. If anyone does acquire the script through unfair means, Proof of Work is an intensive algorithm

[1] "os — Miscellaneous operating system interfaces — Python 3.10.2 ...." https://docs.python.org/3/library/os.html. Accessed 11 Mar. 2022.
[2] "socket — Low-level networking interface — Python 3.10.2 ...." socket — Low-level networking interface — Python 3.10.2 documentation. Accessed 5 Mar. 2022.
[3] "threading — Thread-based parallelism — Python 3.10.2 ...." https://docs.python.org/3/library/threading.html. Accessed 5 Mar. 2022.
[4] "hashlib — Secure hashes and message digests — Python 3.10.2 ...." https://docs.python.org/3/library/hashlib.html. Accessed 5 Mar. 2022.
[5] "getpass — Portable password input — Python 3.10.2 documentation." https://docs.python.org/3/library/getpass.html. Accessed 11 Mar. 2022.
[6] "Proof of Work (PoW) Definition - Investopedia." https://www.investopedia.com/terms/p/proof-work.asp. Accessed 11 Mar. 2022.
[7] "Proof of authority - Wikipedia." https://en.wikipedia.org/wiki/Proof_of_authority. Accessed 11 Mar. 2022.

that requires a lot of computation power. This means that anyone trying to manipulate the blockchain will also have to invest in building a validating farm which should be faster at validating than the set of 'good' servers in the blockchain.

Communication Model

This is a peer-to-peer model where each peer can request and provide information at any point. It has been divided. Data is produced and passed on to a dedicated server (peer on the network) which can redistribute it among the entire network after validation. The 'master' copy is just any copy of the blockchain that is held by the validators. Multiple validators can exist on the network, these are also peers in the same network only with an elevated authority to validate blocks. Certain sockets have been assigned inside the code to get it to work, which can be expanded and handled better based on the size of the project. I have used a TCP/IP connection. The receiver opens a port to receive data and the data can be sent. Since there is only the blockchain file that is to be distributed among peers, one does have to specifically request for certain files. A more real version of this code should be able to look for a peer from a certain category based on the peers it has already connected to and then scan. For the purpose of this project the peer that opens a port displays the relevant socket address so that one can manually add it to the code and establish a connection on the local host.

For a simpler distinguishing factor we have divided peers into two categories.

1. *Peers* - IoT Devices can directly communicate with these peers. They are spread across the country as dealerships and authorized mechanics are. They do not have the privilege to add to the blockchain, but can receive it from their peers/elevated peers.
2. *Elevated Peers* - They act the same as normal peers except that they can add data to the blockchain and distribute it again. Limited in number but with huge computing power to perform calculations faster.

Important Considerations

Network is focussed on providing secure immutable data from generator to user. To have secure hashes, SHA256 was used, which is publicly available and highly secure. Each data is hashed and the unique hash is stored on the block itself.

Since there are so many uploads to the network, a time and energy intensive consensus algorithm like Proof of Work may not be very sustainable forcing to move to other algorithms such as Proof of Stake[8].

Implementation Details

Since the modules are written in Python it can be used on an OS as long as it supports Python. Connections are made using TCP/IP connections. An authentication system is also in place with a hashed password that is not visible at all in plaintext. A menu driven program which allows the peer to share the blockchain between each other and data with the main server.

[8] "Proof-of-Stake (PoS) Definition - Investopedia." 17 Dec. 2021, https://www.investopedia.com/terms/p/proof-stake-pos.asp. Accessed 11 Mar. 2022.

USPs

The unique selling point of this project would be the fact that it is based on the blockchain, with security built in, making the data secure and immutable while being public. With the way the consensus algorithm is implemented we can be sure each update can track to a previous one providing the data in a sequential manner. The functionality of the peers is specifically limited so that they have minimal interaction with the network and the project is more data focussed.

Advancements on Project 1

The similarities in the project implementation were obvious. We have combined both of our projects, using Apnatva's idea to develop a Vehicle Information Logger. The original consensus algorithm was changed to combine Proof of Authority and Proof of Work.

References

To code the functionality of this project some of the code was adapted from various websites. They are

1. File Transfer using Socket Programming[9].
2. Blockchain[10].

[9] "Build Your Own File-Transfer App Using Python Within 5 Minutes." 5 Jun. 2021, Build Your Own File-Transfer App Using Python Within 5 Minutes | by c0d3x27 | Geek Culture | Medium. Accessed 11 Mar. 2022.
[10] "Create simple Blockchain using Python - GeeksforGeeks." 19 Feb. 2022, https://www.geeksforgeeks.org/create-simple-blockchain-using-python/. Accessed 11 Mar. 2022.