Blockchain and Space

(Bibliographic report)

Ziad Kheil*, Jonathan Detchart† *Institut Supérieur de l'Aéronautique et de l'Espace

(ISAE-SUPAERO), 31400 Toulouse, FRANCE
Email: ziad.kheil@student.isae-supaero.fr

***Abstract* - Blockchains have been around for a few years now, yet the vast majority of people only see them as speculation tools and are quick to undermine the innovative aspect they have shown. Ethereum, with their smart-contracts, is one example that holds the potential to change the way we interact with one another online. In view of creating a space debris related smart-contract allowing a pooling of information, this paper will deal with the deployment of a private blockchain for testing purposes, the control of this private chain and the interaction of a web based app with said smart-contract.**

1 Context

The specifics of the smart-contract is not the point of this report, but please feel free to check my colleague's complimentary report on the subject. Instead, I will focus on the behind the scenes of the project. Deploying smart-contracts and making them successful require a certain number of steps to be made beforehand. First of all an appropriate testing environment is strongly recommended, thus the reason for deploying a private ethereum blockchain, secondly to monitor, and simplify the management of this chain, we decided to create a gateway for our test chain. Finally, user-friendly interfaces to interact with our smart-contract were needed for two reasons,first to make it instinctive to use, but also because the security and basis of our smart contract relies on its web handler.

All you need to know about Ethereum networks is that they share many similarities with other blockchains in the way it functions, the specificities of it being mainly that it creates a *decentralized* virtual machine (Ethereum Virtual Machine) which executes scripts on its nodes, thus smart-contracts can be computed on all nodes. This provides a decentralized authority to execute contracts (*smart-contracts*) linking two or more users. Their yellow paper [14] provides further details about this.

2 Ethereum basics

When it comes to Ethereum, and blockchains as a whole, a lot is to be said, I will stick to the fundamentals necessary to understand what we worked on and why it is needed.

To begin with, Ethereum is a platform for decentralised applications called smart-contracts or DApps, it provides an immutable database through its consensus based decision taking. [2] [14] Decentralised and anonymous are keywords here, these features are especially interesting because they mean that no third party controls, meddles with, or referees transactions between individuals [8]. Instead, miner nodes (individuals connected to the blockchain whom compute what is required) all contribute to run and verify the results of computations, constitute what is known as the EVM. The computations of these nodes are stored in blocks, which are then hashed and linked to one another, although an individual may try to modify a block, the data is considered immutable because with each block, a *consensus algorithm* is applied, to verify everyone has the correct blocks and no information has been tampered with [14] [12].

The previous all adds up to a technology allowing for secure, anonymous data exchange and *promises* between individuals with no need for a liability such as a third party referee. In Ethereum specifically, smart-contracts that link 2 or more individuals can safely interact - provided that both have read the code of the contract they are subscribing to (beware of badly written contracts [3]).

3 State of The Art

Obviously, monitoring solutions and wallet handlers already exist and are widely used in the Ethereum (and more generally, blockchain) world. For example a Besu-compatible choice for monitoring various metrics of our network could be Prometheus [9] [10], which works perfectly well and allows to access various data, alongside tools such as Geth [5] which provides a solution to manipulate our network. But several reasons pushed us to code our own monitoring and manipulating tool and web interface: first and foremost to understand how everything works. Indeed, using a ready built tool is very handy, but since this was my first time dealing with blockchains and ethereum, starting from scratch was essential to understand how everything functions and fits in together, furthermore it greatly helped me understand how a network and a web interface could communicate and what was or was not possible. This knowledge was essential and helped smooth everything out for the rest of the project. Another reason was personalising what can be done with and the available metrics to fit our needs.

On the same page, tools to handle wallets, make transactions and subscribe to contracts are also widespread. A private network compatible example is MetaMask [7], which is even available as a Chrome browser extension. Again, for the same reasons as previously stated, I decided to include this in my project.

Finally, since the ultimate goal of the project was the space debris smart-contract, and this is a novel aspect, the web handlers to communicate with this specific contract are unique, thus completely new, at the time of writing this report, the project is not over yet thus I can not get into more details, but many tools I evoque in the next section, and pre-existing algorithms are obviously used (i.e cryptography algorithms to guarantee the safety of the information that is to be sold).

4 Software choices

As we pointed out, the use of a private local Ethereum Network was vital for us, and we decided to implement our own monitoring and development tools, with this in mind, several open-source possibilities arise. The one that we went forward is Hyperleger's Besu [1]. Besu is an open source ethereum client, it is easily used for developing entreprise applications which need extensive before-hand testing, and especially transaction computing through a private network. The main advantages, and the reason we chose Besu was not only its ease of use, the fact it is a java based code, but especially all the doors it opens: specifically, the option to choose a consensus algorithm and the support of private transactions, on private or even public networks[13].

In what concerns developing web interfaces, whether for monitoring and using the network or communicating with our smart-contract, three existing frameworks were vital. The core of my code is built in *node.js*. Node is a javascript server-side framework, its ease of use, tolerance to module based coding(*require, module, exports*), asynchronous handling and event handlers made it an easy choice [11][6]. Nodejs alone is not sufficient for handling a web page, a middleware and client-side handler was necessary, and we used *Expressjs*. Express is a framework directly intended for use with NodeJs, it provides many useful API's for handling my different requests and information [4], in a general way it made my life so much easier.

Finally, interaction with my Ethereum network. This is made possible by *Web3Js* which allows us to interact with our local blockchain (note that it can also be used with a public networks). The documentation of Web3 is clear about its usage, and it was our only way of communicating with the besu network through nodejs to gather information and control our nodes.

These four frameworks are the basis of my work and what allowed me to quickly overcome creating and manipulating a private Ethereum network as a novice, I highly recommend reading these tools' documentation for anyone new to this field. Furthermore, to understand how I used these tools together and have a more detailed view of my project and results, please have a look the code and the final paper.

References

[1] *Besu Hyperlger*. URL: https://besu.hyperledger.org/en/stable/.

[2] *Ethereum, How Does it Work Anyway?* URL: http://www.easygoing.pflog.eu/32_blockchain_P2P/ethereum_blockchain.pdf.

[3] *Exploring Smart Contract Vulnerabilities*. URL: https://blockgeeks.com/guides/smart-contract-vulnerabilities/.

[4] *Express Framework*. URL: https://expressjs.com/.

[5] *Go Ethereum docs*. URL: https://geth.ethereum.org/.

[6] Robert Ryan McCune. "Node.js Paradigms and Benchmarks". In: 2011.

[7] *MetaMask docs*. URL: https://metamask.io/.

[8] Hossein Nabilou. "How to regulate bitcoin? Decentralized regulation for a decentralized cryptocurrency". In: *International Journal of Law and Information Technology* 27.3 (Sept. 2019), pp. 266–291. ISSN: 0967-0769. DOI: 10.1093/ijlit/eaz008. eprint: https://academic.oup.com/ijlit/article-pdf/27/3/266/30251307/eaz008.pdf. URL: https://doi.org/10.1093/ijlit/eaz008.

[9] *Prometheus docs*. URL: https://prometheus.io/.

[10] *Prometheus ethereum exporter*. URL: https://github.com/31z4/ethereum-prometheus-exporter.

[11] Guillermo Rauch. *Smashing node. js: Javascript everywhere*. John Wiley & Sons, 2012.

[12] Sajana P. TIFAC-CORE. "On Blockchain Applications : Hyperledger Fabric And Ethereum". In: 2018.

[13] Dávid URBANCOK. "Blockchain open-source software comparison [online]". Master's thesis. Masaryk University, Faculty of Informatics, Brno, 2020 [cit. 2020-06-01]. URL: Available%20from%20WWW%20%3Chttps://is.muni.cz/th/qr98z/%3E.

[14] Gavin Wood et al. "Ethereum: A secure decentralised generalised transaction ledger". In: *Ethereum project yellow paper* 151.2014 (2014), pp. 1–32.