Engineering Software Architectures of Blockchain-Oriented Applications

Florian Wessling
Paluno, University of Duisburg-Essen
Schuetzenbahn 70, 45127, Essen, Germany
florian.wessling@paluno.uni-due.de

Volker Gruhn
Paluno, University of Duisburg-Essen
Schuetzenbahn 70, 45127, Essen, Germany
volker.gruhn@paluno.uni-due.de

***Abstract*—Building blockchain-oriented applications forces developers to rethink the architecture of their software from the ground up. The use of blockchain technology poses multiple challenges as the software is operated in a decentralized, trustless, transparent and tamper-proof environment. When building decentralized apps the developers need to deal with blockchain properties such as decentralization, a certain delay in the execution of function calls of distributed code contracts and particularly need to consider how users interact with their application. We surveyed several existing decentralized apps and examined their architecture to identify multiple reoccurring architectural patterns, each with different implications regarding the trust, user experience and security. As building blockchain-oriented applications is gaining importance, models, tools and methods for blockchain-oriented software engineering have to be developed. This paper gives a first hint towards architectural patterns for blockchain-based applications and motivates why it is important to consider how the user interacts with the decentralized apps.**

I. Introduction

Blockchain-Oriented Software Engineering (BOSE) [1] is an emerging area of research for building decentralized applications (abbreviated "DApps") based on blockchain technology (see [2], [3]). Currently, the Ethereum blockchain [4], [5] is the most prominent platform for building DApps and will therefore be the focus of this paper. DApps can be described as open-source software, storing its data and records of operation in a decentralized ledger that participants agree on using a consensus algorithm and employing a token mechanism (see [6], [7]). DApps generally consist of two parts: Business logic and an interface for users to interact with the DApp.

The business logic is represented by one or more Executable Distributed Code Contracts (abbreviated "EDCC", a more precise term for Smart Contracts[1]), which are deployed on the blockchain network. There are multiple ways how the front-end could be designed to allow the users the interaction with the business logic. As EDCCs are executed in a decentralized network that ensures the correct completion of DApp function calls, the user interface has to handle a certain delay in execution. In one variant, the users would interact with the deployed business logic directly (by running an own or using a public blockchain node). In another variant, a centralized server of the DApp provider is used, that interacts with the EDCC on behalf of the users. Nevertheless, decisions like this lead to the architectural design of the DApp that has a strong impact on attributes such as trust, user experience and security. In this paper we will examine the architectural design of existing DApps, identify possible architectural patterns and compare their advantages and disadvantages.

The European Union supported this work through the project CPS.HUB NRW, EFRE No. 0-4000-17.

[1] cf. https://www.ethnews.com/edcc

II. Architectural Patterns of DApps

Pattern A – Self-Generated Transactions

Figure 1 shows the basic architectural pattern of a DApp in which the users directly interact with the EDCC by generating and sending transactions themselves. Three variants exist: Users can (1) directly send a transaction to the blockchain, (2) use a web frontend such as MyEtherWallet [8] or (3) use a browser with built-in wallet such as Chrome with MetaMask [9] or a cryptobrowser such as Cipher [10] or Status [11] (each variant can be carried out with a private blockchain node running on the user's device or a public node operated by a third-party, e.g., Infura [12] or Etherscan [13]).

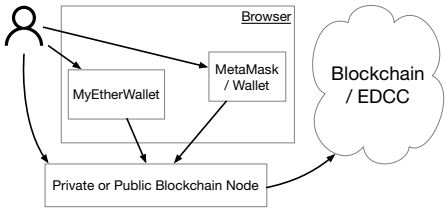

Fig. 1. DApp Pattern A – Self-Generated Transactions

For directly interacting with an EDCC it is necessary that the interface description (Application Binary Interface, ABI) of the EDCC is publicly available. Furthermore, for understanding the specific actions carried out by a contract, the source code should be published as well. DApps structured with this pattern have the security advantage that the users keep their private keys on their device and are able to generate, sign and send transactions on their own. Almost no trust is required in infrastructure or third-party providers as the signed transactions cannot be manipulated. The drawback of this approach is the low user experience as the process is error-prone and requires a strong technical understanding.

978-1-5386-6585-5/18/$31.00 ©2018 IEEE
DOI 10.1109/ICSA-C.2018.00019

This architectural pattern can be found in decentralized cryptocurrency exchanges such as EtherDelta [14], IDEX [15] or gambling DApps such as Ethorse [16].

Pattern B – Self-Confirmed Transactions

The pattern illustrated in Figure 2 offers more convenience for the users than Pattern A as the interaction with the DApp is primarily done using a cryptobrowser or MetaMask. Transactions are not generated by the user but are triggered by the DApp website, presented to the user for further verification and then manually sent to the blockchain node. Thus, this pattern offers a trade-off between convenience and trust that is required in the DApp website providing the transaction details.

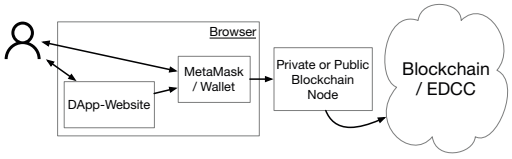

Fig. 2. DApp Pattern B – Self-Confirmed Transactions

Manually generating transactions can be hard or is not always feasible, especially when it is necessary to query the current state of the blockchain or to gather additional data, e.g., like the marketplace of CryptoKitties [17]. This requires a certain trust in the DApp provider as transactions are generated but the implication of their execution is not always completely transparent (e.g., the `geneScience` contract of CryptoKitties without published source code or ABI). Similar to Pattern A, the private key resides with the user and still some technical knowledge is required regarding transactions and gas prices.

Pattern C – Delegated Transactions

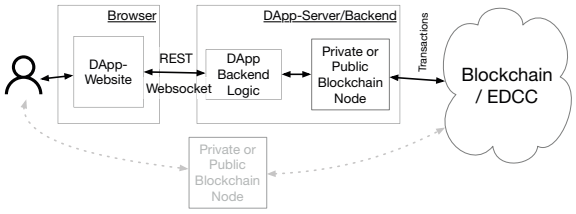

Fig. 3. DApp Pattern C – Delegated Transactions

Even more convenience and a high ease of use is achieved with Pattern C as illustrated in Figure 3. In this case the DApp provider offers a website the users can interact with, not requiring a cryptobrowser or MetaMask plugin. The website communicates with the DApp backend logic via REST calls and encapsulates all blockchain-specific actions. This means the backend is responsible for interacting with the blockchain and is sending transactions on behalf of the user, who is not able to validate them. Pattern C therefore offers the maximum convenience but also requires complete trust in the DApp provider who is handling user logins and manages the private keys. In case the EDCCs used by the provider are known, it would be possible for the user to examine the execution of transactions triggered by the backend. The user can watch the state of the EDCCs or receive feedback via websockets from the provider itself. Cryptocurrency exchanges such as Kraken [18] or Binance [19] are a common example for this pattern as they usually interact with the blockchain on behalf of the user.

III. Conclusion and Future Work

In this paper we surveyed the architecture of several existing DApps in order to identify three common patterns and compare their impact on trust, user experience and security. We motivated why it is important to consider these aspects when engineering software architectures of blockchain-oriented applications. Furthermore we highlight that additional research is required to examine patterns in terms of maintainability, infrastructure costs, transaction costs and complexity.

Examining existing DApps serves as a first quantitative approach for identifying architectural patterns. For our future work we plan to complement this with (semi-)structured questionnaires or interviews to gain insights about reasoning and best practices for creating software architectures.

References

[1] S. Porru, A. Pinna, M. Marchesi, and R. Tonelli, "Blockchain-oriented software engineering: Challenges and new directions," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, 2017, pp. 169–171.

[2] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Crypto-Currencies*, 1st ed. O'Reilly Media, Inc., 2014.

[3] M. Swan, *Blockchain: Blueprint for a New Economy*, 2015.

[4] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," 2013. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[5] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger." [Online]. Available: http://gavwood.com/paper.pdf

[6] D. Johnston, S. O. Yilmaz, J. Kandah, N. Bentenitis, F. Hashemi, R. Gross, S. Wilkinson, and S. Mason. DecentralizedApplications: Decentralized applications white paper and spec. (Accessed 2018-03-15). [Online]. Available: https://github.com/DavidJohnstonCEO/DecentralizedApplications

[7] S. Raval, *Decentralized applications: harnessing Bitcoin's Blockchain technology*, 2016, ISBN 978-1-4919-2452-5.

[8] MyEtherWallet LLC. Myetherwallet. (Accessed 2018-03-15). [Online]. Available: https://www.myetherwallet.com

[9] ConsenSys. Metamask. (Accessed 2018-03-15). [Online]. Available: https://metamask.io

[10] HardFork Inc. Cipher. (Accessed 2018-03-15). [Online]. Available: https://www.cipherbrowser.com

[11] Status Research & Development GmbH. Status – a mobile ethereum os. (Accessed 2018-03-15). [Online]. Available: https://status.im

[12] ConsenSys AG. Infura. (Accessed 2018-03-15). [Online]. Available: https://infura.io

[13] Etherscan.io. Etherscan – the ethereum blockchain explorer. (Accessed 2018-03-15). [Online]. Available: https://etherscan.io

[14] EtherDelta. Etherdelta. (Accessed 2018-03-15). [Online]. Available: https://etherdelta.com

[15] Aurora Labs S.A. Idex – decentralized ethereum asset exchange. (Accessed 2018-03-15). [Online]. Available: https://idex.market

[16] Ethorse. (Accessed 2018-03-15). [Online]. Available: https://ethorse.com

[17] AxiomZen. Cryptokitties. (Accessed 2018-03-15). [Online]. Available: https://www.cryptokitties.co

[18] Payward, Inc. Kraken. (Accessed 2018-03-15). [Online]. Available: https://www.kraken.com

[19] Binance.com. Binance. (Accessed 2018-03-15). [Online]. Available: https://www.binance.com