On the Synchronization Power of Token Smart Contracts

Giorgia Azzurra Marson

NEC Laboratories Europe

Heidelberg, Germany

giorgia.marson@neclab.eu

Abstract

Modern blockchain systems support a variety of distributed applications beyond cryptocurrencies. In particular, smart contracts allow users to execute arbitrary code in a distributed and decentralized fashion. Regardless of their intended application, current blockchain platforms implicitly assume consensus for the correct execution of a smart contract, thus requiring that all transactions are totally ordered. Contrary to common belief, however, consensus is not a necessary requirement to prevent double-spending in a cryptocurrency (Guerraoui et al., PODC'19). On the other hand, the decentralized execution of arbitrary smart contracts does require agreement on the blockchain state, hinting that consensus is needed in this case. Based on our recent findings (Alpos et al., ICDCS'21), in this presentation we will take a closer look at the synchronization requirements of smart-contract token standards defined by Ethereum's Request for Comment (ERC).

Overview

The increasing popularity of decentralized applications has motivated prominent efforts towards improving the scalability and performance of blockchain protocols. Research on distributed protocols has led to many new proposals to scale the throughput of blockchain platforms, giving rise to a plethora of different distributed ledgers today. A common objective of these proposals is to ensure that blockchain nodes execute all transactions in the same order, using the replicated state-machine approach where a broadcast protocol allows blockchain users to agree on a sequence of transactions. This process is often referred to as "consensus", which is equivalent to total-order broadcast. Since reaching consensus is expensive, it is important to understand where it may be avoided without losing consistency.

Concurrent objects for synchronization For investigating concurrency, one usually considers two distinct models: *message-passing* and *shared memory*. In the message-passing model, processes do not have any shared state and communicate with each other via messages. Processes in the shared-memory model, however, operate on the same data, which they access concurrently. Despite these differences, results in one model can be transferred to the other one. The gist of a synchronization problem in a distributed system using message passing is often more clearly expressed by the corresponding shared-memory formalization. Shared-memory abstractions are objects providing operations to processes. The simplest such object is a *register*, which offers operations for reading and writing a value. Another important object is *consensus*, which implements agreement on a value.

The prominent result by Fischer, Lynch, and Paterson [1] establishes the impossibility of implementing consensus from atomic registers in a wait-free manner. In other words, implementing a consensus object with only atomic registers cannot ensure that every invocation to consensus operations terminates. This means that consensus requires a higher level of synchronization than atomic registers. In fact, the consensus object is universal, in the sense that any shared object described by a sequential specification can be wait-free implemented from consensus objects and atomic registers. Therefore, consensus can be used to reason about the synchronization power of all shared objects among a number of processes. This leads to the central concept of *consensus number* to express the synchronization power of shared objects. Formally, the consensus number associated with a shared object O is the largest number n such that it is possible to realize a consensus object from atomic registers and objects of type O, in a system of n

processes. Consensus numbers establish a hierarchy among concurrent objects and allow for comparing them based on their synchronization power as well as their synchronization requirements [2].

Cryptocurrencies do not need consensus! Guerraoui et al. [3] propose a shared-memory abstraction for asset transfer—the basic functionality of a cryptocurrency as implemented in Bitcoin [4]—and show that this requires only a minimal level of synchronization. Specifically, asset transfer has consensus number 1 in the wait-free hierarchy. Their result suggests that current implementations may be sacrificing efficiency and scalability because they synchronize transactions much more tightly than actually needed. For cryptocurrencies that support shared accounts with up to k owners, Guerraoui et al. introduce a k-shared asset transfer object and show that it has consensus number k, which is as powerful as consensus among its k owners. Going beyond their theoretical elegance, these results are of great practical interest because they pave the way to consensus-free implementations of cryptocurrencies.

Our results Modern blockchains support a variety of distributed applications realized by smart contracts. These applications go far beyond cryptocurrencies and decentralized payments, as initially envisioned by Bitcoin. Smart contracts enable blockchain users to execute arbitrary programs in a fully decentralized fashion, akin to a world computer. Introduced by Ethereum, smart contracts come in many different flavors and are the key element in most decentralized finance (DeFi) projects today. Typically, smart contracts represent value using tokens. These are blockchain-based assets which can be exchanged across users of a blockchain platform. Ethereum's Request for Comment (ERC) 20 defines a blueprint for the creation of a specific type, dubbed ERC20 token, one of the most widely adopted tokens on Ethereum. The ERC20 standard provides functions for handling tokens over Ethereum, allowing users to own and exchange goods such as digital and physical assets. It formulates a common interface for fungible tokens and has become the most widely-deployed API for implementing a token functionality.

To investigate the synchronization power of token smart contracts on Ethereum, in our work [5] we propose a shared-memory abstraction for a token object that captures and generalizes the functionality of an ERC20 contract. ERC20 is considerably more flexible than the transaction model in Bitcoin. The additional features of ERC20 make it possible, for example, that account owners allow other users to interact with their accounts and issue transfers on their own. Likewise, the ERC20 token object is strictly more powerful than k-shared asset transfer, because account owners may approve other spenders to transfer tokens. Moreover, an owner may approve new spenders flexibly, at any time, and for arbitrary amounts. This results in a dynamicity that has no counterpart with the standard asset transfer object. This increased level of dynamicity is reflected in the token synchronization requirements. Namely, the consensus number of ERC20 tokens depends on the number of approved spenders for the same account, which may change as the account owner enables more spenders.

Our findings suggest that synchronization among a dedicated subset of blockchain nodes is enough for building prominent applications. Namely, ERC20 require consensus only among the largest set of enabled spenders for an account. Importantly, the exact synchronization requirements can be readily deduced from the current object's state by reading the blockchain state. This opens up the possibility to deploy realistic smart contracts, such as ERC tokens, on more robust and performant protocols than consensus-based blockchains.

References

[1] M. J. Fischer, N. A. Lynch, and M. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[2] M. Herlihy, "Wait-free synchronization," *ACM Trans. Program. Lang. Syst.*, vol. 13, no. 1, pp. 124–149, 1991.

[3] R. Guerraoui, P. Kuznetsov, M. Monti, M. Pavlovic, and D. Seredinschi, "The consensus number of a cryptocurrency," in *PODC*. ACM, 2019, pp. 307–316.

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," https://bitcoin.org/bitcoin.pdf, 2009.

[5] O. Alpos, C. Cachin, G. A. Marson, and L. Zanolini, "On the synchronization power of token smart contracts," in *ICDCS*. IEEE, 2021, pp. 640–651.