Synchronization Requirements of Token Smart Contracts

Giorgia Azzurra Marson
NEC Labs Europe

Based on joint work with:
Orestis Alpos, Christian Cachin, Luca Zanolini
University of Bern
Motivation

- Decentralized applications rely on a distributed protocol emulating a shared ledger (blockchain)
Motivation

- Decentralized applications rely on a distributed protocol emulating a shared ledger (blockchain).
- Distributed consensus (a.k.a. total-order broadcast) ensures consistency among ledgers.
Motivation

- Decentralized applications rely on a distributed protocol emulating a shared ledger (blockchain).
- Distributed consensus (a.k.a. total-order broadcast) ensures consistency among ledgers.
- However, consensus is the bottleneck of blockchain speed 😞.
Consensus is not necessary for decentralized cryptocurrencies (!)

Prior work [GKMPS’19]

Consensus is not necessary for decentralized cryptocurrencies (!)

Prior work [GKMPS’19]

Approach:
- Define AT abstraction as shared-memory object
- Analyze the synchronization power (consensus number) of AT

Main result:
The consensus number of AT is 1 (range: \([1, \infty])\).

“Basic cryptocurrency functionality” = Asset Transfer (AT) object

Consensus is not necessary for decentralized cryptocurrencies (!)

Prior work [GKMPS’19]

Approach:
- Define AT abstraction as shared-memory object
- Analyze the synchronization power (consensus number) of AT

```
AT.balance
AT.transfer
```

"Basic cryptocurrency functionality” = Asset Transfer (AT) object

```
AT
```

Main result:
The consensus number of AT is 1 (range: [1, ∞]).

Interpretation:
- AT has weakest synchronization power
- Consensus is an overkill for basic cryptocurrency applications
- Intuitive reason: total order is not necessary to prevent double spending (causal order is enough)

Consensus is not necessary for decentralized cryptocurrencies (!)

**Approach:**
- Define AT abstraction as shared-memory object
- Analyze the synchronization power (**consensus number**) of AT

**Main result:**
The consensus number of **AT** is 1 (range: \([1, \infty]\)).

**What about Smart Contracts?**

**Interpretation:**
- AT has weakest synchronization power
- Consensus is an overkill for basic cryptocurrency applications
- Intuitive reason: total order is not necessary to prevent double spending (causal order is enough)
Synchronization power of Token Smart Contracts

This work [ACMZ’21]

Motivating question:
What level of synchronization is required for “popular” smart contracts?

Synchronization power of Token Smart Contracts

Motivating question:
What level of synchronization is required for "popular" smart contracts?

Approach:
- Define ERC-20 abstraction as shared object $T$
- Analyze the consensus number of ERC-20 object

ERC-20 token standard
(most popular fungible token in Ethereum)

Synchronization power of Token Smart Contracts

Motivating question:
What level of synchronization is required for “popular” smart contracts?

Approach:
- Define ERC-20 abstraction as shared object $T$
- Analyze the consensus number of ERC-20 object

New features compared to AT:
- Account owners can delegate approved spenders to manage asset
- Approval of spenders is dynamic (any time, arbitrary amounts)

Synchronization power of Token Smart Contracts

This work [ACMZ’21]

Motivating question:
What level of synchronization is required for “popular” smart contracts?

Approach:
- Define ERC-20 abstraction as shared object $T$
- Analyze the consensus number of ERC-20 object

New features compared to AT:
- Account owners can delegate approved spenders to manage asset
- Approval of spenders is dynamic (any time, arbitrary amounts)

Main results:
The consensus number of ERC-20 token $T$ dynamically changes with the contract state ($q$):

$$CN(T_q) = 1 + \max_a \# \text{ approved spenders for account } a$$

Outlook

Prior work
Synchronization power of cryptocurrency:

\[ CN(AT) = 1 \]

⇒ transactions can be processed concurrently
⇒ total order is not necessary, causal broadcast can be used instead

Currently adopted, an overkill
Total-order broadcast (blockchain)

Sufficient (faster, asynchronous!)
Causal broadcast
Outlook

Prior work
Synchronization power of cryptocurrency:

\[ CN(AT) = 1 \]

⇒ transactions can be processed concurrently
⇒ total order is not necessary, causal broadcast can be used instead

Currently adopted, an overkill
Total-order broadcast (blockchain)

Sufficient (faster, asynchronous!)
Causal broadcast

This work
Synchronization power of Ethereum ERC-20 token T:

\[ CN(T_q) = 1 + \max_{a} \{\# \text{ approved spenders for account } a\} \]

⇒ transactions can be processed concurrently, if issued by spenders of different accounts
⇒ total order is needed only for resolving conflicts, causal broadcast could be used optimistically

Ideally: optimally-concurrent protocol for “useful” smart contracts

As-concurrent-as-possible broadcast
Outlook

Prior work
Synchronization power of cryptocurrency:
\[ CN(AT) = 1 \]

⇒ transactions can be processed concurrently
⇒ total order is not necessary, causal broadcast can be used instead

Currently adopted, an overkill

Total-order broadcast (blockchain)

Sufficient (faster, asynchronous!)

Causal broadcast

This work
Synchronization power of Ethereum ERC-20 token T:
\[ CN(T_q) = 1 + \max_a \{ \# \text{ approved spenders for account } a \} \]

⇒ transactions can be processed concurrently, if issued by spenders of different accounts
⇒ total order is needed only for resolving conflicts, causal broadcast could be used optimistically

Ideally: optimally-concurrent protocol for “useful” smart contracts

As-concurrent-as-possible broadcast

Thank you for your attention 😊
Backup slide
Synchronization power (consensus number) of shared objects

- Consensus is universal: any shared object has a wait-free implementation from consensus objects
- \( \Rightarrow \) consensus can serve as reference for the synchronization power of shared objects

**Consensus number** of object \( O \): \( CN(O) := \max n \mid \exists \) wait-free implementation of consensus object from objects of type \( O \) and registers, in a system with \( n \) processes.

- Intuitively: max # processes that can be synchronized “using \( O \)”
  \[ CN(O) = 1 \Rightarrow O \text{ useless for synchronization} \]
  \[ CN(O) = \infty \Rightarrow O \text{ can synchronize any number of processes} \]
- Metric to compare synchronization power of shared objects