AIP-001: Strictly Ordered, Loosely Ordered, and Unordered Transactions

seo@ 2019-07-01

Problem Definition

Using RLP encoding, Ethereum allocates some bytes of the serialized transaction for nonce[1]:

> *The nonce is the number of transactions sent from a given address.*

Each time you send a transaction, the nonce value increases by 1. In handling transactions, the following rules are applied:

- Transactions must be in order. You cannot have a transaction with a nonce of 1 mined before a transaction with a nonce of 0.
- No skipping! You cannot have a transaction with a nonce of 2 mined if you have not already sent transactions with a nonce of 1 and 0.

We call this type of nonce strictly ordered nonce. It's a good solution for preventing common blockchain issues such as double-spending. However, nonces do have the following drawbacks:

- Noncing makes it difficult to set up payment systems where multiple clients share the same account.
- Nonce handling is often a common bottleneck in many system

Proposed Solution

We propose the following:

> *Make numbered nonce optional, i.e., allow other types of nonce: loosely ordered and unordered*

Depending on the characteristics of the application, the transaction creators can choose one of the three types of nonce: 1) strictly ordered nonce, 2) loosely ordered nonce, and 3) unordered nonce. Strictly ordered nonce is identical to the Ethereum nonce. For loosely ordered nonce, the following rules are applied:

---

[1] Bitcoin doesn't use nonce as its protocol is not based on accounts.

- If a new transaction has a nonce greater than the nonce of the lastest transaction processed, it's accepted
- Otherwise, it's rejected

In this case, the nonce field of the transaction can be set with a constant value (e.g. -2) and timestamp is used to order the transactions, i.e., timestamp does the role of nonce.

For unordered nonce, the following rules are applied:

- If there is no conflict in the transaction hash, a new transaction is always accepted
- Otherwise, it's rejected

In this case, the nonce field of the transaction can be set with a constant value (e.g. -1) and timestamp can be used to distinguish transactions with the same operations.

Types of nonces can be compared as follows:

Type	When to use	Typical cases	API version
Strictly ordered	Single client	Human-generated txs	1.0
Loosely ordered	Multiple clients, transactions are aligned with time	Machine-generated txs in multiple places	TBD
Unordered	Single or multiple clients, each transaction is processed independently.	Machine-generated txs, Tx generation time can be different from tx submission time	1.0

Document History

Date	Who	Change	Notes
2019-07-01	seo@	Initial draft	
2019-07-02	lia@, kimminhyun@, chris@, seo@	Internal review	
2021-03-15	seo@	Published	