AIP[1]-001: Numbered, Ordered, and Unordered Nonces[2]

platfowner 2019-07-01

Problem Definition

Using RLP encoding, Ethereum allocates some bytes of the serialized transaction for nonce[3]:

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

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

Proposed Solution

We propose the following:

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

Depending on the characteristics of the application, transaction creators can choose one of the three types of nonce: 1) **numbered nonce**, 2) **ordered nonce**, and 3) **unordered nonce**.

---

[1] AI Network Improvement Proposal. Visit https://docs.ainetwork.ai for the full list.
[2] Original title: Strictly ordered, loosely ordered, and unordered transactions (link).
[3] Bitcoin doesn't use nonce as its protocol is not based on accounts.

Numbered nonce is identical to the Ethereum nonce. For the ordered nonce, the following rules are applied:

- 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 the transaction's timestamp is used to order the transactions, i.e., timestamp takes 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
Numbered	Single client	Human-generated txs	1.0
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

Conclusion

In addition to the popular *numbered* nonce type, we introduced two more nonce types, *ordered* and *unordered*, for flexible use cases.

Document History

Date	Who	Change	Notes
2019-07-01	platfowner	Initial draft	
2019-07-02	liayoo, kmh4500, platfowner, (chris)	Internal review	

2021-03-15	platfowner	Published	
2021-05-12	platfowner, liayoo	Introduced simplified naming: - *strictly ordered -> numbered* - *loosely ordered -> ordered*	
2021-05-12	platfowner	Github IDs Link to full list	
2021-05-12	platfowner	Repubished	