AIP-002: Micropayment Protocol for Low-Level Services

seo@ 2019-07-07

Problem Definition

Many blockchain applications are based on high-level, person to person payments. However, there are also many blockchain applications based on low-level services where payment needs to be made from person to machine or machine to machine. Examples such payments include transportation card, check card and public FaaS calls. In contrast to high-level person-to-person payments, low-level payments have the following characteristics:

- The payment is one-directional, i.e., from the user to the service provider
- The amount is relatively small (e.g. less than 0.01 USD)
- The frequency is relatively high (e.g. 100 times per second)

Naive use of the solutions currently available (e.g. Ethereum Smart-contract) for such low-level payment have the following issues:

- High transaction fee
    - e.g. 0.177 USD in Ethereum, 0.07 - 0.63 USD in Bitcoin (as of 2019-07-06)
- Long confirmation time
    - e.g. minutes in Ethereum, tens of minutes in Bitcoin (as of 2019-07-06)

Bitcoin Lightning Network [1] is one of the most widely known solutions addressing these problems. However this solution requires a separate P2P network, which has many limitations [2].

In this document, we propose a new solution called Micropayment Protocol for Low-level Services (MPLS).

Requirements

The solution has the following requirements:

- Support for one-directional payments
- Instant payment processing
- Reasonable transaction fees

Proposed Solution

The solution proposed in the document is facilitated through a *Reservation Account* provided by the blockchain. The Reservation Account will be covered by a separate document, however the key ideas behind this Reservation Account can be summarized as follows:

- Users can create a reservation account for holding tokens from his/her account
- Once a reservation has been made, a unique ID (i.e. reservation ID) is assigned
- Each reservation is pegged to a target account which will eventually receive payment for services provided
- The creator of the reservation account becomes its owner
- When a reservation is made, the user specifies an **expiration period**[1] for the account. Once the reservation expires, the user can reclaim the remaining balance of the account at any time.

The key idea of the proposed solution is to make use of off-chain transactions:

- Users attach signed payment transactions to each API call
- Resource providers send payment transactions in a bundle to the blockchain. When this bundle of transactions is processed through the blockchain, payment takes effect.
- To make sure that users have sufficient funds for payment, resource providers require users to make reservation accounts

The proposed solution will be implemented through the four separate steps described in the following sections.

Step 1: Open Service

The service provider registers a service (e.g. FaaS) sending the following parameters to the blockchain:

- Service provider's account (i.e. public address)
- Service address (e.g. https url)
- Pricing policy and price (e.g. credit per call)
- Service specifications
    - Input parameters
    - Output format
- **User reservation requirements:**
    - Minimum reservation amount (depending on the price)
    - Minimum expiration period (e.g. 24 hours)

Once the service is registered successfully, a unique ID (i.e. service ID) is assigned.

[1] For security reasons, it can be specified by a blockchain block number instead of actual timestamp.

Step 2: Reserve Credit

A user who wants to use a service needs to make a reservation, which should meet the reservation requirements from the service provider. The actual amount or the actual expiration period can be chosen depending on the user's plan for his/her use of the service. For example, if the user plans to use the service long term, the user can opt to create a reservation account with a relatively large balance and long expiration period. Alternatively, for trial services, a small balance and reservation period would suffice.

Step 3: Use & Pay for Service

Whenever the user calls the service (e.g. public FaaS call), he/she provides the following to the service:

- Input parameters
- Payment transaction

The payment transaction consists of the following information:

- Reservation ID
- User’s account (i.e. public address)
- Service provider’s account (i.e. public address)
- Transaction amount
- Signature

The service provider verifies the given payment transaction by checking the following:

- Check if the transaction signature is valid
- Check if the reservation account has enough balance

If the transaction is valid, the service provider provides the service (e.g. returns FaaS call results). In this step, all transaction-related work is done off-chain, i.e., independent of the blockchain transaction handling performance (TPS).

Step 4: Charge

Finally, the service provider charges the user the cost. It’s not done directly to the user’s account but to the reserve account. To do so, the service provider sends a package of user payment transactions to the blockchain. The blockchain verifies the request and, if it’s valid, the requested amount is actually transferred from the reservation account to the service provider’s account. This request is handled by the blockchain via one API call for a reasonable, once-off fee.

Note that it’s the service provider’s responsibility to charge the cost before the expiration of the reservation account.

Conclusion

Using the proposed solution, we can set low-level services up where payment is done person to machine or machine to machine with low latency and for a reasonable transaction fee.

It is notable that the proposed solution is

- Not application specific, i.e., it can be applied to a wide range of low-level services.
- Independent of the underlying blockchain protocol. For example, it can be implemented on Ethereum Network using a Smartcontract. Of course a blockchain that natively supports the protocol would maximize performance.

Future Work

We have the following future work:

- Reservation Account API
- Convenient auto-signing ODMP transactions from the user side

Links

[1] Bitcoin Lightning Network, https://lightning.network/lightning-network-summary.pdf
[2] Mathematical Proof That the Lightning Network Cannot Be a Decentralized Bitcoin Scaling Solution, https://medium.com/@jonaldfyookball/mathematical-proof-that-the-lightning-network-cannot-be-a-decentralized-bitcoin-scaling-solution-1b8147650800

Document History

Date	Who	Change	Notes
2019-07-07	seo@	Initial draft	
2019-07-08	lia@, kimminhyun@, chris@, seo@	Internal review	
2021-03-15	seo@	Published	