

Multi-client Partially Non-Interactive and Instantaneous One-way Payment Channel for Ethereum

August 8, 2018 – DRAFT

Thomas Shababi¹, Joël Gugger¹, and Daniel Lebrecht¹

TrueLevel SA, Neuchâtel, Switzerland
{tom, joel, d}@truelevel.io

Abstract. Ethereum is a distributed computing platform and operating system featuring smart contract functionality. Transactions are faster than other blockchain but not instant and each of them cost some “gas”. This gas is used to quantify the amount of fee to pay for computation.

Keywords: Crypto-currencies, Ethereum, Payment channels

1 Introduction

Requirements Language The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [1].

2 Multi-clients payment channels

Partially non-interactive multi-clients payment channels are composed of clients $c \in \mathcal{C}$ and one provider \mathcal{P} . \mathcal{C} is the set of all clients registered in the multi-clients payment channel. Clients can send money to the provider through the channel $c \rightarrow \mathcal{P}$ but cannot receive through the channel $\mathcal{P} \nrightarrow c$.

A payment channel is by definition a structure composed of two layers of states. The first layer of states is registered to the blockchain, i.e. “on-chain” or contract states $\mu \in M$, and the second layer of states is kept “off-chain” between the participants, i.e. channel states $\sigma \in \Sigma$.

It is possible to transition from a state σ to another state σ' . We denote state transition with \rightarrow , i.e. $\sigma \rightarrow \sigma'$. Modifiers $\omega \in \Omega$ are used to create transition on channel states $\sigma \in \Sigma$. A transition in Σ depends on modifiers and contract states. We denote a transition from $\sigma, \sigma' \in \Sigma$ that depends on $\mu \in M$ and $\omega \in \Omega$ as

$$\sigma \xrightarrow{\mu+\omega} \sigma'$$

Contract state transitions are triggered by external events $e \in \mathcal{E}$ or messages $m \in \mathcal{M}$. Transitions between two states $\mu, \mu' \in M$ are denoted $\mu \rightarrow \mu'$. If the

transition is due to an external event e we write $\mu \xrightarrow{e} \mu'$. If the transition is due to a message m we write $\mu \xrightarrow{m} \mu'$. Generic transitions are noted as $\mu \rightarrow \mu'$.

A message m can be “applied” to a state μ , we denote this operation with $m(\mu)$, i.e. $\mu \xrightarrow{m} \mu'$. Messages are created based on a channel states. Each channel state $\sigma \in \Sigma$ can derive its corresponding message $m \in \mathcal{M}$

$$\forall \sigma \in \Sigma, \exists m \mid m \in \mathcal{M}$$

In reality it is not necessary to derive all messages. External events (top ups) affect the channel state but no messages are created because the previous generated message is still valid and usable to trigger a valid state transition in M .

2.1 Contract state variables

We define variables to access the the contract state $\mu \in M$ for a client $c \in \mathcal{C}$.

Current on-chain index, $I(c)$ For each client the current index i must be retrievable, $\forall c \in \mathcal{C}, \exists I(c) \in \mu \geq 0$. The index for a client must start from 0.

Current on-chain balance, $B_\mu(c)$ For each client the current balance amount must be retrievable, $\forall c \in \mathcal{C}, \exists B_\mu(c) \in \mu \geq 0$.

2.2 Contract states

Each client $c \in \mathcal{C}$ in the contract is defined by an on-chain state $\mu \in M$. States $\mu \in M$ are a tuple

$$(I(c), B_\mu(c))$$

2.3 Channel state variables

We define variables to represent: (i) the lifetime of a single channel in the multi-clients channel architecture, (ii) the total amount deposited for a client over the lifetime, (iii) the total amount own by the provider over the lifetime, and (iv) the minimal and full available amount for the client.

Channel lifetime, $L(c \rightarrow \mathcal{P})$ It exists one lifetime and only one per element in $\mathcal{C} \times \mathcal{P}$ and begins when the first deposit is made, i.e. one lifetime per client $c \in \mathcal{C}$.

Total deposit, $D(c)$ Total deposit of a client c represents the total amount recieved from the begining of the lifetime. Each top up increases the total deposit.

Total sent, $\sum c \rightarrow \mathcal{P}$ Total amount sent to the provider by a client c represents the sum of all transaction since the begining of the lifetime. Each transaction increases the total sent.

Minimal available amount, $A_m(c)$ Minimal amount available for a client c is computed with the latest channel state σ . Without quering the contract state μ it is impossible to know if a bigger amount is now available. The minimal available amount is computed with

$$A_m(c) = D(c) - \sum c \rightarrow \mathcal{P}$$

Client available amount, $A(c)$ The full amount available into a single channel for a client $c \in \mathcal{C}$ is computed with

$$A(c) = D(c) - \sum c \rightarrow \mathcal{P} + (B_\mu(c) - B_\sigma(c)), \quad B_\mu(c) \geq A(c)$$

It is worth noting that $B_\mu(c) - B_\sigma(c)$ is added to the difference of total deposit and total sent in case of on-chain changements, like top up.

2.4 Channel states

Each client $c \in \mathcal{C}$ is defined by their channel state $\sigma \in \Sigma$. States $\sigma \in \Sigma$ are composed of: (i) a validity index, (ii) the latest observed on-chain balance of the client, (iii) the total deposit of the client, and (iv) the total owned by the provider. We denote a state $\sigma \in \Sigma$ as

$$(i, B_\sigma(c), D(c), \sum c \rightarrow \mathcal{P})$$

2.5 Messages

Messages $m \in \mathcal{M}$ are exchanged between clients \mathcal{C} and provider \mathcal{P} . A message is related to one and only one element in $\mathcal{C} \times \mathcal{P}$.

Minimal message A minimal message $m \in \mathcal{M}$ between a client c and the provider \mathcal{P} is composed of four components: (i) a validity index, (ii) the lastest observed balance, (iii) the total of deposit, and (iv) the total owned by the provider. We denote a message $m \in \mathcal{M}$ as

$$(i, B_\sigma(c), D(c), \sum c \rightarrow \mathcal{P})$$

Message derivation Messages are derived from channel states Σ as

$$\begin{aligned} \sigma &\implies m \\ (i, B_\sigma(c), D(c), \sum c \rightarrow \mathcal{P}) &\implies (i+1, B_\sigma(c), D(c), \sum c \rightarrow \mathcal{P}) \end{aligned}$$

2.6 Modifiers

Modifiers $\omega \in \Omega$ are input parameters that modify channel states σ . They encapsulate the channel capabilities. In this multi-clients payment channel the capability is sending money through the channel from clients to provider, i.e. $c \rightarrow \mathcal{P}$.

Payment, $\omega(a)$ This modifier is used to increase the balance $O(\mathcal{P})$ of the provider of a amount from a client c .

3 State transitions

3.1 Contract state, $\mu \rightarrow \mu'$

Contract states $\mu \in M$ can transition $\mu \rightarrow \mu'$ because of a message $m \in \mathcal{M}$ or external events.

Settlement $(i, B_\mu(c)) \xrightarrow{m} (i+1, B_\mu(c) \downarrow) \mid m = (i+1, B_\sigma(c), D(c), \sum c \rightarrow \mathcal{P})$. It is worth noting that with this rule it is not possible to settle a zero amount because the balance amount $B_\mu(c)$ must go down!

$$\begin{aligned} O(\mathcal{P}) &= B_\mu(c) - A(c) \\ &= B_\mu(c) - (D(c) - \sum c \rightarrow \mathcal{P} + (B_\mu(c) - B_\sigma(c))) \end{aligned}$$

$O(\mathcal{P})$ is the amount owned by the provider at the settlement time. The provider can settle the amount in the balance minus the remaining funds of the client c .

$$\begin{aligned} B_\mu(c) \downarrow &= B_\mu(c) - O(\mathcal{P}) \\ &= B_\mu(c) - (B_\mu(c) - (D(c) - \sum c \rightarrow \mathcal{P} + (B_\mu(c) - B_\sigma(c)))) \end{aligned}$$

The new client balance is the current contract balance minus the the amount due to the provider.

Full refund $(i, B_\mu(c)) \xrightarrow{m} (i+1, 0) \mid m = (i+1, B_\sigma(c), D(c), \sum c \rightarrow \mathcal{P})$. It is a full refund if the client available funds in the channel for the client are equal to the current contract balance

$$\begin{aligned} B_\mu(c) &\stackrel{?}{=} A(c) \\ &\stackrel{?}{=} D(c) - \sum c \rightarrow \mathcal{P} + (B_\mu(c) - B_\sigma(c)) \end{aligned}$$

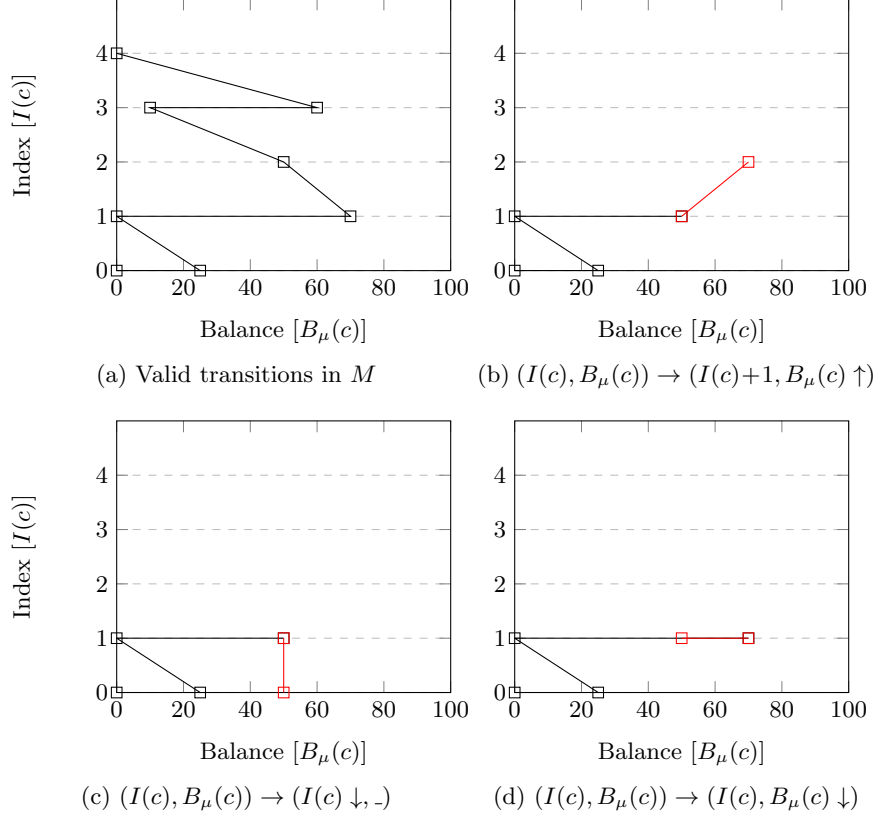


Fig. 1. State transitions in M

Partial refund $(i, B_μ(c)) \xrightarrow{m} (i+1, 0) \mid m = (i+1, B_σ(c), D(c), \sum c \rightarrow \mathcal{P})$. It is a partial refund if the client available funds in the channel are smaller than the current contract balance

$$B(c) \stackrel{?}{>} A(c)$$

In that case the provider amount is

$$O(\mathcal{P}) = B_μ(c) - A(c)$$

Top up $(i, B_μ(c)) \xrightarrow{e} (i, B_μ(c) \uparrow)$. Top up increases the contract balance for a client $c \in \mathcal{C}$. Validity index i must not be incremented during the top up.

Invalid settlement $(i, B_μ(c)) \rightarrow (i, B_μ(c) \downarrow)$. This transition is invalid because the balance is decreased without incrementing the validity index i . The current set of transactions must be invalidate after the settlement.

	ω	μ	m	σ
	$\omega(a)$	$(I(c), B_\mu(c))$	$(i, B_\sigma(c), D(c), \Sigma)$	$(i, B_\sigma(c), D(c), \Sigma)$
init		(0, 10)		\emptyset
transact	+1		(1, 10, 10, 1)	(0, 10, 10, 1)
transact	+1		(1, 10, 10, 2)	(0, 10, 10, 2)
settle		(1, 8)		
transact	+2		(2, 8, 10, 4)	(1, 8, 10, 4)
transact	+2		(2, 8, 10, 6)	(1, 8, 10, 6)
top up		(1, 18)		(1, 18, 20, 6)
transact	+1		(2, 18, 20, 7)	(1, 18, 20, 7)
settle		(2, 13)		

Table 1. State transitions during channel lifetime

Invalid top up $(i, B_\mu(c)) \rightarrow (i + 1, B_\mu(c) \uparrow)$ Increasing validity index i while increasing the contract balance invalidates the set of transaction i . The current set of transactions must be invalidate only during a settlement.

Invalid validity index $(i, B_\mu(c)) \rightarrow (i \downarrow, -)$ Decreasing the validity index i is always invalid. Validity index must not be decreased.

3.2 Channel state, $\sigma \rightarrow \sigma'$

Channel state transitions $\sigma \xrightarrow{\mu+\omega} \sigma'$ for $\sigma, \sigma' \in \Sigma$, i.e. applying a modifier ω on the current state μ to result on σ' . The contract state μ must be query on-chain. The base channel state is

$$\sigma = \begin{cases} \emptyset & \text{if no previous state exists} \\ (i, B_\sigma(c), D(c), \Sigma c \rightarrow \mathcal{P}) & \text{otherwise} \end{cases}$$

Payment, $\mu(I(c), B_\mu(c)) + \omega(a)$ Payments trigger transition in channel states from a base state σ to a destination state σ' noted $\sigma \xrightarrow{\mu+\omega(a)} \sigma'$. The destination state is computed as

$$\sigma' = \begin{cases} (I(c), B_\mu(c), B_\mu(c), a) & \text{if } \sigma = \emptyset \\ (I(c), B_\mu(c), D(c), \Sigma c \rightarrow \mathcal{P} + a) & \text{otherwise} \end{cases}$$

	ω	μ	m	σ
	$\omega(a)$	$(I(c), B_\mu(c))$	$(i, B_\sigma(c), D(c), \sum)$	$(i, B_\sigma(c), D(c), \sum)$
init		(0, 3)		\emptyset
top up		(0, 5)		(0, 5, 5, 0)
transact	+3		(1, 5, 5, 3)	(0, 5, 5, 3)
settle		(1, 2)		
transact	+1		(2, 2, 5, 4)	(1, 2, 5, 4)
top up		(1, 4)		(1, 4, 7, 4)
settle		(2, 3)		
transact	+2		(3, 3, 7, 6)	(2, 3, 7, 6)

Table 2. Settlement after top up

Top up, $\mu(I(c), B_\mu(c)) \uparrow$ Each top up event modify the contract state μ , then the channel state σ must be updated to keep track of the real balance.

$$\sigma' = \begin{cases} (I(c), B_\mu(c), B_\mu(c), 0) & \text{if } \sigma = \emptyset \\ (I(c), B_\mu(c), D(c) + (B_\mu(c) - B_\sigma(c)), \sum c \rightarrow \mathcal{P}) & \text{otherwise} \end{cases}$$

4 Protocol

The protocol describes interactions between clients and the provider and messages negotiation. Protocol introduce a new component in the channel architecture, directives. Directives are the concrete implementation of messages and are distributed between client and provider.

Client actions A client can send money through the channel and refund his money after some time if he submitted the right message m corresponding to the latest channel state σ .

Provider actions The provider can settle channels with his latest message corresponding to the latest channel state.

Anyone actions Anyone can top up a channel for himself or for another client.

4.1 Payment, $c \rightarrow \mathcal{P}$

Message $m \in \mathcal{M}$ is composed of two directives: (i) a refund directive for the client, and (ii) a settlement directive for the provider.

Refund directive, d_r Refund directive d_r allow the client to claim his remaining funds $A(c \rightarrow \mathcal{P})$ and transfer them after some delay if the provider is not able to dispute the claim. The provider is not able to dispute the claim if the directive come from the message generated by the latest channel state in Σ .

$$h = \text{keccak256}(\text{contract}, c, I(c), B_\sigma(c), D(c), \Sigma c \rightarrow \mathcal{P})$$

$$d_r = \text{sign}(\text{prefix}, h) \quad \text{with } \mathcal{P} \text{ private key}$$

Settlement directive, d_s Settlement directive d_s allow the provider to claim immediately, i.e. without any delay, his owned amount for a specific channel.

$$h = \text{keccak256}(\text{contract}, c, I(c), B_\sigma(c), D(c), \Sigma c \rightarrow \mathcal{P})$$

$$d_r = \text{sign}(\text{prefix}, h) \quad \text{with } c \text{ private key}$$

References

- [1] S. Bradner. *IETF RFC 2119: Key words for use in RFCs to Indicate Requirement Levels*. Tech. rep. 1997.