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

August 9, 2018 – DRAFT

Thomas Shababi[1], Joël Gugger[1], and Daniel Lebrecht[1]

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 registred in the multi-clients payment channel. Clients can send money to the provider through the channel $c \rightarrow \mathcal{P}$ but cannot recieve through the channel $\mathcal{P} \not\rightarrow c$.

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) afect 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 acces the the contract state $\mu \in M$ for a client $c \in \mathcal{C}$.

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

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

Refund parameters, $\boldsymbol{R(c)}$ For each client the current refund parameters must be retreivable, $\forall c \in \mathcal{C}, \quad \exists R(c) \in \mu$ such that $R(c)$ is the provider owned amount at the moment of the requested refund. The remaing money must go to the client.

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

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

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, $\boldsymbol{L(c \rightarrow \mathcal{P})}$ 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, $\sum c \to \mathcal{P}$ 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

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

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

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

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 $\sigma \in \varSigma$. States $\sigma \in \varSigma$ 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 $\sigma \in \varSigma$ as

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

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

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

Message derivation Messages are derived from channel states $\varSigma$ as

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

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, $\boldsymbol{\omega(a)}$ 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, $\boldsymbol{\mu \rightarrow \mu'}$

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

Settlement $(I(c), B_\mu(c), R(c)) \xrightarrow{m} (I(c)+1, B_\mu(c)\downarrow, R(c))$ where

$$m = (i, B_\sigma(c), D(c), \sum c \rightarrow \mathcal{P}) \mid i = I(c)+1$$

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.

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

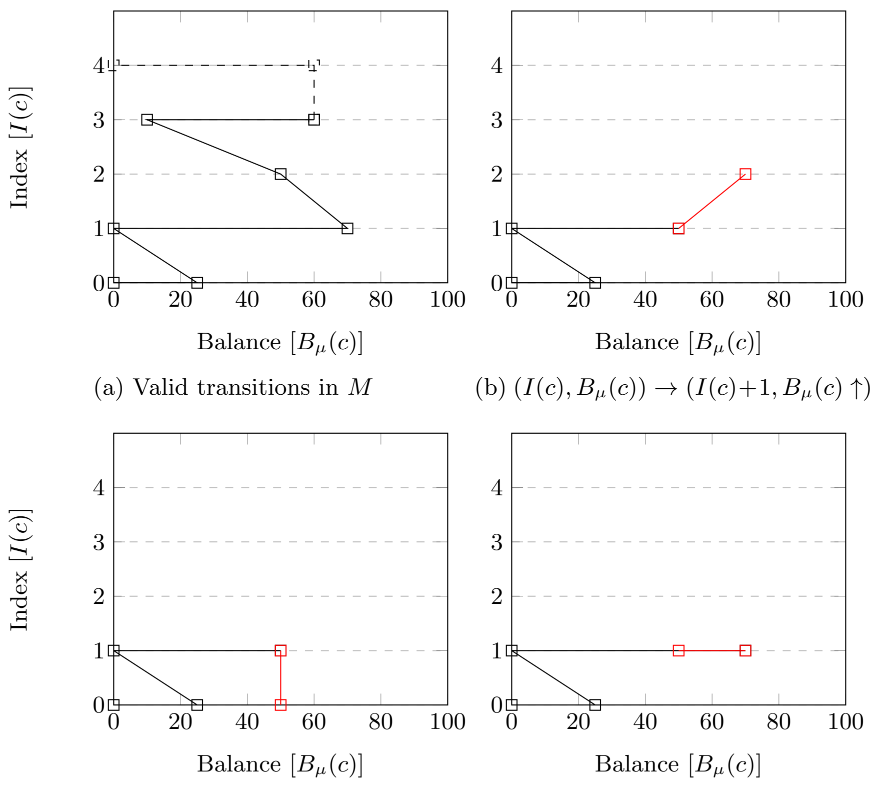
The new client balance is the current contract balance minus the the amount due to the provider.

Refund $(I(c), B_\mu(c), R(c)) \xrightarrow{m} (I(c)+1, B_\mu(c), R(c)')$ where

$$m = (i, B_\sigma(c), D(c), \sum c \rightarrow \mathcal{P}) \mid i = I(c)+1$$

It is a full refund if the client available funds in the channel for the client are equal to the current contract balance and a partial refund if the available funds of the client are smaller than the balance

$$R(c)' = \begin{cases} B_\mu(c) - A(c) & \text{if } B_\mu(c) > A(c) \\ 0 & \text{if } B_\mu(c) = A(c) \end{cases}$$



(a) Valid transitions in M

(b) $(I(c), B_\mu(c)) \to (I(c){+}1, B_\mu(c)\uparrow)$

(c) $(I(c), B_\mu(c)) \to (I(c)\downarrow, \_)$

(d) $(I(c), B_\mu(c)) \to (I(c), B_\mu(c)\downarrow)$

Fig. 1. State transitions in M, transitions in red are not allowed. Dashed transition are during refund process. The first step is the refund request, the second can be either claim, resolve, or dispute.

Dispute $(I(c), B_\mu(c), R(c)) \xrightarrow{m} (I(c), 0, 0)$ where

$$m = (i, B_\sigma(c), D(c), \sum c \to \mathcal{P}) \mid i = I(c)$$

If $O(\mathcal{P}) > R(c)$ the provider won the dispute and $O(\mathcal{P})$ is transfer to the provider plus some penalities, the rest is sent to the client.

Resolve $(I(c), B_\mu(c), R(c)) \xrightarrow{m} (I(c), 0, 0)$ where

$$m = (i, B_\sigma(c), D(c), \sum c \to \mathcal{P}) \mid i = I(c)$$

If $O(\mathcal{P}) = R(c)$ the provider accept the refund. $O(\mathcal{P})$ is transfer to the provider and the rest is sent to the client imediately.

	ω	μ	m	σ
	$\omega(a)$	$(I(c), B_\mu(c), R(c))$	$(i, B_\sigma(c), D(c), \sum)$	$(i, B_\sigma(c), D(c), \sum)$
init		$(0, 10, 0)$		$\emptyset$
transact	$+1$		$(1, 10, 10, 1)$	$(0, 10, 10, 1)$
transact	$+1$		$(1, 10, 10, 2)$	$(0, 10, 10, 2)$
settle		$(1, 8, 0)$		
transact	$+2$		$(2, 8, 10, 4)$	$(1, 8, 10, 4)$
transact	$+2$		$(2, 8, 10, 6)$	$(1, 8, 10, 6)$
top up		$(1, 18, 0)$		$(1, 18, 20, 6)$
transact	$+1$		$(2, 18, 20, 7)$	$(1, 18, 20, 7)$
settle		$(2, 13, 0)$	$(3, 13, 20, 7)$	$(2, 13, 20, 7)$
refund		$(3, 13, 0)$		
claim		$(3, 0, 0)$		

Table 1. State transitions during channel lifetime

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

Invalid settlement $(I(c), B_\mu(c), R(c)) \to (I(c), B_\mu(c)\downarrow, R(c))$. This transition is invalid because the balance is decreased without incrementing the validity index $I(c)$. The current set of transactions must be invalidate after the settlement.

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

Invalid validity index $(I(c), B_\mu(c), R(c)) \to (I(c)\downarrow, \_, \_)$. Decreasing the validity index $I(c)$ is always invalid. Validity index must not be decreased.

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

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.

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

Table 2. Settlement after top up and refund resolved

The base channel state is

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

Payment, $\boldsymbol{\mu(I(c), B_\mu(c), R(c)) + \omega(a)}$ 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

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

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

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

4 Protocol

The protocol describes interactions between clients and the provider and messages negociation. 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 submited the right message m corresponding to the lastest channel state σ.

Provider actions The provider can settle channels with his lastest message corresponding to the latest channel state. If a client requested a refund, the provider has the choice to dispute it or resolve it.

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

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

Payment messages $m \in \mathcal{M}$ are composed of one directive for the provider signed by the client.

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

$$\begin{aligned} h &= \texttt{keccak256}(\texttt{contract}, c, I(c), B_\sigma(c), D(c), \Sigma c \to \mathcal{P}) \\ d_r &= \texttt{sign}(\texttt{prefix}, h) \quad \text{with } c \text{ private key} \end{aligned}$$

4.2 Refund

Refund messages $m \in \mathcal{M}$ are composed of one directive generated by the client and signed by the client. The provider must not be able to create directives d_r.

Refund directive, d_r Refund directive d_r allow the client to claim all the remaing funds $A(c \to \mathcal{P})$ 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.

$$\begin{aligned} h &= \texttt{keccak256}(\texttt{contract}, c, I(c), O(\mathcal{P})) \\ d_r &= \texttt{sign}(\texttt{prefix}, h) \quad \text{with } c \text{ private key} \end{aligned}$$

A refund request can be closed in tree different maners

$$\text{close a refund by} \begin{cases} \text{dispute} & \text{if the provider is able to prove a lie} \\ \text{resolve} & \text{if the provider agrees} \\ \text{claim} & \text{otherwise by the client after delay} \end{cases}$$

4.3 Dispute refund

Disputes are won if the provider can prove that is owns a payment directive d_s where

$$O(\mathcal{P})' > O(\mathcal{P}), \quad O(\mathcal{P})' \in d_s \land O(\mathcal{P}) \in d_r$$

with

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

4.4 Resolve refund

Provider can choose to agree with the request and resolve it directly. By resolving the request the provider will get his remaing funds and send the client available funds $A(c)$ directly to the client. It is worth noting that the provider has no incentive to resolve request where $\sum c \to \mathcal{P} = 0$.

4.5 Claim refund

The client can claim his funds after the delay if the provider has not disputed nor resolved the request.

References

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