Counterfactual: Generalized State Channels

Ethereum Sharding Research

Jeongho Jeon <maczniak@gmail.com>

July 23, 2018

Whitepaper Foundation, Nonce
(for internal discussion purposes only)

Whitepaper
Foundation

State/Payment Channel

places as little on-chain as possible while still remaining secure.

Payment Channel Lightning Network, Raiden, Teechain, eltoo, Sprites

State Channel Counterfactual, Perun, Pisa

Application-specific Channel Fate Channel (FunFair), Celer Network, Gnosis

2/3 sessions of Master workship: off the chain[1] (Jun 30-Jul 1) are related to state/payment channels.

[1] https://github.com/state-of-blockchain-scalability/blockchain-conference-videos/blob/master/master-workshop-off-the-chain.md

Whitepaper
Foundation

State/Payment Channel: Pros

- the cheapest among scalability solutions
- instant finalization upon participants' consent
- fast withdrawal (e.g., cooperative withdrawal)
- state channel network
- (optional) partial choseout

State/Payment Channel: Cons

- participant availability (i.e., participants or delegates need to keep online.)
- participants need to agree about currently finalized state quickly.
- implement application-specific channels from scratch, without standard libiraries and reference implementations

Comparison

	Lightning	Raiden	Sprites	Perun	Counterfactual
per payment path	✓	✓	✓	✓	✓
rent-a-path			✓		✓
coop instant withdrawal	✓				✓
partial closeout	✓	?	?	?	✓
network setup	$O(n)$	$O(n)$	$O(1)$	$O(\log n)$	$O(1)$

Whitepaper
Foundation

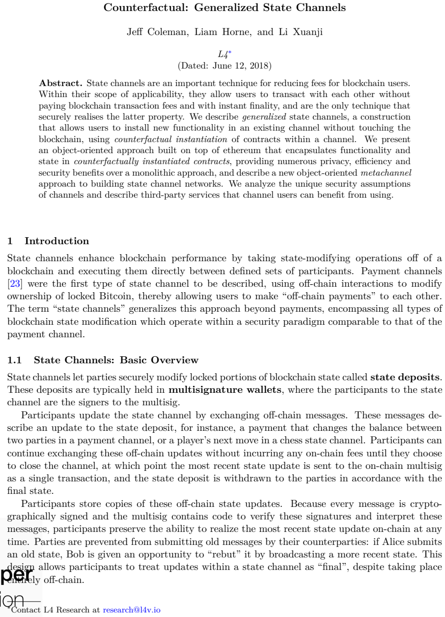
Counterfactual: Generalized State Channels

released on Jun 12[a] by L4[b], and updated on GitHub repository[c]. “generalized” means that participants can have only one state channel, and run a lot of applications parallelly on the channel. Developers need not to know state channel details. It is designed with modular (code reuse) and open source style (no codes yet).

[a] https://twitter.com/statechannels/status/1006536423086977024a

[b] https://l4.ventures/

[c] https://github.com/counterfactual



Counterfactual: Generalized State Channels

Jeff Coleman, Liam Horne, and Li Xuanji

L4[*]

(Dated: June 12, 2018)

Abstract. State channels are an important technique for reducing fees for blockchain users. Within their scope of applicability, they allow users to transact with each other without paying blockchain transaction fees and with instant finality, and are the only technique that securely realises the latter property. We describe *generalized* state channels, a construction that allows users to install new functionality in an existing channel without touching the blockchain, using *counterfactual instantiation* of contracts within a channel. We present an object-oriented approach built on top of ethereum that encapsulates functionality and state in *counterfactually instantiated contracts*, providing numerous privacy, efficiency and security benefits over a monolithic approach, and describe a new object-oriented *metachannel* approach to building state channel networks. We analyze the unique security assumptions of channels and describe third-party services that channel users can benefit from using.

1 Introduction

State channels enhance blockchain performance by taking state-modifying operations off of a blockchain and executing them directly between defined sets of participants. Payment channels [23] were the first type of state channel to be described, using off-chain interactions to modify ownership of locked Bitcoin, thereby allowing users to make “off-chain payments” to each other. The term “state channels” generalizes this approach beyond payments, encompassing all types of blockchain state modification which operate within a security paradigm comparable to that of the payment channel.

1.1 State Channels: Basic Overview

State channels let parties securely modify locked portions of blockchain state called **state deposits**. These deposits are typically held in **multisignature wallets**, where the participants to the state channel are the signers to the multisig.

Participants update the state channel by exchanging off-chain messages. These messages describe an update to the state deposit, for instance, a payment that changes the balance between two parties in a payment channel, or a player’s next move in a chess state channel. Participants can continue exchanging these off-chain updates without incurring any on-chain fees until they choose to close the channel, at which point the most recent state update is sent to the on-chain multisig as a single transaction, and the state deposit is withdrawn to the parties in accordance with the final state.

Participants store copies of these off-chain state updates. Because every message is cryptographically signed and the multisig contains code to verify these signatures and interpret these messages, participants preserve the ability to realize the most recent state update on-chain at any time. Parties are prevented from submitting old messages by their counterparties: if Alice submits an old state, Bob is given an opportunity to “rebut” it by broadcasting a more recent state. This design allows participants to treat updates within a state channel as “final”, despite taking place entirely off-chain.

[*] Contact L4 Research at research@l4v.io

Whitepaper
Foundation

counterfactual

1. X could happen on-chain, but doesn’t
2. The enforcement mechanism allows any participant to unilaterally make X happen
3. Participants can act as though X has happened

counterfactual instantiation

Counterfactual instantiation is the act of signing these commitments, not the act of actually calling deploy on-chain. Counterfactual instantiation is the process by which members of a channel agree to be bound by an off-chain smart contract. You can install and upgrade channel-based (channelized) applications without any on-chain operations (no fees).

Nonce-Dependent Conditional Finalization: The existence of state channel objects depends on nonce object's nonce. All other objects depend on "root nonce".

Whitepaper
Foundation

counterfactual address

current contract address `sha3(rlp.encode([normalize_address(sender), nonce]))[12:]`

counterfactual address of a contract need to be a deterministic function of the contract’s initialization code. Ideas:

- ENS (Ethereum Name Service)
- account abstract (depends on the code and a chosen salt)
- skinny CREATE2 (endowment, memory_start, memory_length, salt) `sha3(msg.sender ++ salt ++ init_code)[12:]`

Whitepaper
Foundation

state

Multisig wallets take control of state deposits.

Ethereum account state `(nonce, balance, contract code, storage)`

Counterfactual state is divided into nonce and application-specific state.

Counterfactual Pros: Privacy

Third party cannot tell state channels from multisig wallets. Transaction details such as amount are visible only to participants.

> *A sufficiently powerful multisignature wallet is sufficient to act as a state deposit holder, and is the only on-chain component that must be deployed for each additional state channel.*

Counterfactual metachannel

- state channel network
- not “per payment path” like Lightning $O(n)$ setup, but rent-a-path for a certain amount of time like Sprites $O(1)$ setup (that does not require cooperation from intermediaries for every payment)
- onion routing - intermediaries cannot know the whole path

Whitepaper
Foundation

Ideas

- implement a (generalized) state channel
- make a game on the top of the state channel