Hello everyone,

I'd like to introduce the generalized state channels framework that L4 has been working on for the past couple of months. We released an introductory blog post at https://medium.com/statechannels/counterfactual-generalized-state-channels-on-ethereum-d38a36d25fc6 geared towards a more non-technical audience, as well as a technical paper at https://counterfactual.com/statechannels.

For existing readers of ethresear.ch, I think the best place to start would be to jump right in and read the paper

In the rest of this post I'll highlight what we think are the important innovations we've introduced, especially with respect to the existing discussion on channels we've seen on this forum. I hope this list provides a good comparison to existing research and will provide a good jumping-off point for discussions.

Latency

While many people discuss channels as a way to increase transaction throughput, we additionally highlight the latency-reduction benefits. Once a channel is set up, interacting within it only requires off-chain message exchange, hence the interaction finalizes instantly and we provide web-like response times. This is in contrast to non-channelized interactions (e.g., directly on ethereum, or on Plasma) where one normally has to wait for "confirmations" (or for some other finality metric, e.g., a finalized block in Casper). Channels are the only technology that provide this instant finality guarantee.

Application-specific channels, generalized state channels, counterfactual instantiation, and multisigs

We introduce the concept of "generalized" state channels. In our terminology, an application-specific state channel runs a specific application (e.g., payments or chess), and requires an on-chain transaction to deploy before the specific application can be used. In contrast, a generalized state channel is one where new applications and functionality can be installed into an existing channel without any on-chain transactions. In particular, a user installing a new application into an existing channel experiences instant finality for installing the application.

In a generalized state channel, parties are able to enter into a contract (i.e. be bound by the terms of the contract) just by exchanging messages. We call the process of being bound by the contract, "counterfactually instantiating" the contract.

See sections 4.1.3 and 5.3 for more details.

Multisigs

In our constructions, the only on-chain component of any individual state channel is a multisig wallet. This is possible due to counterfactual instantiation. This has privacy and upgradability benefits.

See section 5.4 for more details

Counterfactual terminology

Counterfactual terminology is not a new feature we provide, but a useful way of thinking about state channels and many other L2 techniques. For any on-chain operation X, we say that "counterfactual X" holds when

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

This terminology is already useful when talking about existing payment channels. In that case, X could be "4 ether is transferred from the payment channel smart contract to Alice's account, and 6 ether from the smart contract to Bob's account", and counterfactual X would be the state of affairs if both parties have the latest signed copy, which records Alice's balance as 4 and Bob's as 6.

We use this terminology to talk about counterfactual instantiation, counterfactual state, counterfactual state transitions, and counterfactual objects. This definition can also be adapted to talk about Plasma.

Object-Oriented Approach

A generalized state channel can have many applications or instances of the same application going on at the same time. We organize this in an “object-oriented approach” that combines state and functionality, much like how ethereum contracts combine state and functionality.

See sections 5.5 and 5.7 for more details.

Miscellaneous Features

One reason we think the way we think about state channels is fruitful is that it allows us to naturally support many features without any special effort:

- n-party channels for n > 2
- Instant closeout of a channel (i.e., releasing the funds to the owning parties without waiting for the dispute period; only possible if all parties agree)
- Partial withdrawal, where some amount of funds is taken out of a channel and moved elsewhere on-chain, without the channel being closed
- Top-ups, where some amount of funds is moved on-chain into the channel (e.g., in a payment channel, if your balance is running low, you can do a top-up into the channel to replenish your balance)
- Contracts that have some notion of time
- Contracts that depend on non-channelized on-chain state (e.g., betting with your friend within a channel on the outcome of some Augur market)

State Channel Networks and Metachannels

In a payment channel network, two people who do not have a payment channel with each other but who have channels with some path of intermediaries to each other can still do a channelized payment to each other. State channel networks are the analogous constructions for state channels. Some existing work on state channel networks include:

- Perun’s “virtual channels”, at Perun: virtual payment and state channel networks
- Celar Network’s mediated conditional payments
- For payment channel networks: Lightning, Sprites

These different constructions present different usage models to the end-user. For example, with lightning or sprites style payment channel networks, the intermediaries have to cooperate for every payment, even if the same route is reused, whereas virtual channels don’t have this problem. The flip side is that per-payment routing might tie up capital for lesser amounts of time.

Our “metachannels” construction is an object-oriented approach to constructing state channel networks. In terms of usage model, is most similar to Perun’s virtual channels, but with some differences. Note that since generalized state channels support counterfactual instantiation, you can also do HTL-style payments within a network of generalized state channels, if desired.

See section 5.9 and 4.3.9 for constructions and discussions.

Griefing

Analyses of griefing in state channels is under-explored in the existing literature. We explain different types of griefing possible, how it affects the threat model of channel users, and some mitigations in section 3, as well as mention some third-party services that can help in section 7.2.

Definition and Limitations of Channels

The definition of “state channels” is not entirely consistent among researchers, and has recently provoked a friendly twitter war

For our definitions, interactions within a channel must proceed only by unanimous consent, and these interactions must have instant finality (except in special cases like partial withdrawal).

In section 7.1, we give a representative example of something that cannot be channelized (a public bounty).