Folks may be interested in this recent post by a State Channels team member, George Knee discussing what “instant finality” means and whether or not state channels exhibit this property.

TLDR: no: but for many practical purposes, yes

Feature Photo

Not Instant: but much, much faster than the alternative (Photo by Markus Spiske from Pexels)

Terminology is a slippery old thing.

To lubricate technical discussions, good terminology must strike a balance between precision and brevity. Oftentimes this balance needs to shift depending on the nature of the speaker (be they a dev, or a marketer) and the listener (be they an expert or a layperson).

“Instant Finality” is one such vernacular snippet, and is the focus of this post. It is meant to convey something of the advantages that the addition of a state channel layer can bring to a blockchain. The concept of finality dates back to Satoshi Nakamoto’s original paper introducing blockchains and bitcoin. It is a property of blockchain transactions, and can be defined as the state of affairs when that transaction has been included in a block, and that block has had a sufficient number of subsequent blocks mined on top of it so that observers can be confident that the transaction will never be rejected by the network. It implies that the transaction, and therefore its effects (such as changes to token balances) is here to stay.

Finality in proof-of-work blockchains is not binary: there is no magical moment when a transaction becomes final. Instead, as Nakamoto sketched in his whitepaper, the probability of the transaction being rejected decreases exponentially over time, as the network establishes consensus about a longer and longer chain of blocks.

Of course, at any moment the chain could fork, leading to the possibility of a transaction falling by the wayside: but this becomes less and less likely, as long as confirmatory new blocks are observed, pointing back to the block which includes that transaction. Observing a fork, or rival chain that does not

include our transaction, would increase

the probability that our transaction will be rejected. It’s quite possible that finality never arrives, and the transaction actually becomes “anti-final”, which we could take to mean that we are very confident that it has been rejected. This could happen, for example, if the network is undergoing a so-called 51% attack. Again we could never be entirely certain either way: but our best estimate of the probability of rejection will, over time, approach extremely close to 0 or 1. By waiting long enough, we would have certainty for all practical purposes.

So how do state channels impact on this situation? Well, the truth is, they do not. They are known as a layer 2 (L2) technology, meaning that they do not change the underlying protocol of the blockchain by definition, and cannot alter its finality properties: although there are plenty of efforts in the layer 1 (L1) space for doing just that.

Moreover, state channel applications must involve at least one “lock” and one “unlock” L1 transaction to successfully bootstrap assets from the underlying blockchain. These transactions are subject to the exact same finality concerns as any other. Users will typically wait for the best part of a minute (a veritable infinity from a UX point of view) for those L1 transactions to be considered final.

State channels work by exploiting the opportunity that exists between the locking and unlocking: by providing an execution environment, between those steps, for blockchain-like applications to run securely. These applications involve state updates, which we might call “L2 transactions”, which are strongly analogous to blockchain transactions, with a few differences. They are cryptographically signed, contain arbitrary data, and are interpreted by a virtual machine defining programmatic (app-developer-defined) rules for the execution. The fate of L2 transactions, however, does not depend on a complex and probabilistic consensus protocol. The state of a state channel is updated (for example) unilaterally or unanimously. Each update is either finalizable or it isn’t.

What does finalizability mean? As is the case for L1 finality, it requires that the state channel state would be accepted by the L1 chain, in the sense of not causing the chain’s virtual machine to revert, were it to be submitted as a part of the L1 unlock transaction. This implies that the L2 transaction is properly formatted and correctly signed, and will pass any extra checks imposed by the state channel protocol designers and application developers. Once these conditions are validated by a participant, that participant has confidence that they will be able to cause the unlock to happen in the prescribed way at some point in the future. The L2 transaction resulted in a finalizable state.

These relatively loose requirements make the time to L2 finalizability potentially extremely fast in comparison to L1 finality.

This is where “Instant” comes from: it’s an imprecise (even inaccurate) term, but reflects the qualitative step-change in user experience from regular Dapps to State Channel Apps: fast enough for exciting applications such as micropayments and real-time games that are beyond the reach of L1 chains.

The finalizability of all of the L2 transactions between the lock and unlock L1 transactions is instant in this sense, conditional on the finality of the lock and unlock transactions themselves. If the lock transaction is rejected, all of the state channel updates become meaningless in the same way. But since we could wait for as long as desired after the lock transaction,

before starting the state channel, we can make the finality conditional on something almost certain, or as close to certain as you would like.

Similarly for the unlock transaction, with one caveat: state channels rely on a challenge and response (or dispute) mode which protects against participant inactivity. This mode affords a malicious counterparty the ability to try and unlock the state channel funds in an unfavourable way, by submitting an L1 transaction. Honest parties are protected as long as they can refute with their own L1 transaction within a timeout period. If this timeout is too short, the concept of finalizability begins to break down. If it is much longer than the time taken for the refute transaction to become final, then the finalizability of the L2 transactions becomes effectively unconditional.

It is worth remembering that asset transfer means something slightly different in L2: because consensus about a transaction is achieved only among a fixed (and typically small) set of participants, state updates are only ever meaningful to that set. If I pay a counterparty in a state channel, they cannot yet use that money to pay someone outside the channel, until we unlock. Although we have something like instant finality, we do not have instant liquidity:

In summary, those looking for a deeper understanding may wish to substitute “rapid conditional finalizability” for “instant finality” when describing state channels. For everyone else, the latter term is most likely here to stay: for the same reason that snippets such as “bitcoin is decentralized”, “public-key cryptography is secure”, “hash functions are not invertible” are also here to stay. The wider community accepts the imprecision because it allows conversations to continue more fluently, and for the main ideas to be communicated more easily. In those situations where precision is paramount, however, we must not be afraid to drill down and unpack a more comprehensive understanding than a two-word phrase can hope to convey — as this 1000+ word post has attempted to do!

Copied from

Medium – 14 May 20

Do state channels exhibit instant finality?

TLDR: No: but for many practical purposes, yes

Reading time: 5 min read