Thanks to @Mikerah for help revising this.

Merged Consensus

The high-level goal of merged consensus is to provide an objective deterministic permissionless consensus protocol that can be used for side chains, and runs entirely on-chain. It is an alternative to designs likeProof-of-Stake consensus for side chains, but unlike stake-limited consensus protocols it does not require a separate token, a separate set of block producers, a bunch of strong assumptions, etc.

Preambles

1. This spec assumes full data availability. This can be done by posting ordered data either directly on-chain, or somewhere the chain has access to (such as accessing Eth 2.0 shard chains within the Eth 1.0 EVM). Future work will be on relaxing this assumption, or making the assumption optimistically, to allow for greater scalability.
2. Merged consensus can be used with multiple side chains simultaneously, with each side chain running its own instance of merged consensus, but we'll concern ourselves with a single side chain for this post.
3. Merged consensus is not merged mining and can be used with any base-layer consensus protocol, including ones that are stake-limited rather than work-limited.
4. There are no restrictions on the underlying data model or execution framework of the side chain, so long as compact fraud proofs can be produced.

Assumptions

Adversarial Model

We assume an adversary is anything weaker than "majority of main chain block producers attacking the main chain," including "a majority of main chain block producers attacking the side chain." As the system presented here relies on fraud proofs, we can do no better than this, so our adversarial model is optimal.

Network Model

Merged consensus has the same network model assumption as the blockchain it uses to relay consensus messages (as transactions).

Minimal Merged Consensus Spec

Anyone may commit to a block on the side chain by including

1. block header (previous block header hash, state root, transactions root) and
2. block transaction data and
3. bond B

as a transaction to a contract on-chain (a commitment transaction

). No authentication, registration, stake, or work is required. The contract only accepts the first valid

commitment transaction (defined below) in each main chain block; subsequent commitment transactions simply revert and return the bond to the sender. Accepted commitment transactions lock the bond until the committed side chain block is finalized.

Valid commitment transactions must include all side chain transaction data as calldata

(similar to rollup, but witness data also needs to be included). The contract will Merkleize the transactions and verify the resulting root against the transactions root in the block header supplied by the commitment transaction. Invalid commitment transactions result in the bond being burned.

In addition, the contract checks that the commitment transaction links to the previous side chain tip, or includes a fraud proof and builds upon the most recent non-fraudulent side chain block. If not, the commitment transaction simply reverts and the bond is returned to the sender.

Side chain blocks are considered finalized

if they (or one of their ancestors) have not been subjected to a successful fraud proof within a finality period of F

main chain blocks. After this time, the bond B

may be withdrawn.

Side chain blocks that are not yet finalized may be fraudulent, as transaction data is only posted on-chain but not validated. Within F

blocks of a side chain block being committed on-chain, anyone may submit a fraud proof that attests to the block's invalidity, invalidating it and all its descendants. A successful fraud proof burns half the bond of the invalid block and sends the other half to the prover. The bonds for descendants are burned entirely. Fraud proofs can also be posted simultaneously as a new side chain block is committed.

Valid side chain blocks have the same finality guarantees as the main chain (i.e.

, they are as irreversible).

Fork Choice Rule

We show here that a non-trivial fork choice rule is not needed for this spec of merged consensus, as forks are not possible.

By the protocol definition, new blocks are added to the side chain through commitment transactions, which can only be accepted by the contract if they either build upon the current tip of the side chain, or roll back the side chain with a fraud proof then build upon the new tip. All these actions happen only through messages that are recorded on the main chain—as transactions, which happen atomically and for which there are no race conditions. Therefore from the point of view of the main chain only a single fork of the side chain can possibly exist.

A key intuition here is that, unlike classical side chains which may commit side chain block hashes to the main chain so that they can't be reverted but otherwise exist independently, side chains with merged consensus are inextricably linked to their main chain. Under this scheme, a side chain block cannot exist without being committed to on the main chain. This gives rise to the powerful property of seamlessly supporting main chain re-organizations: for each fork of the main chain, a single fork of the side chain exists (which is only valid for that fork of the main chain). Should the network's view of the canonical main chain change, its view of the canonical side chain will change as well.

Exit Procedure

A user who wishes to exit their state from the side chain can make a special withdrawal

transaction on the side chain, which simply burns their state on the side chain. Once a side chain block that includes this withdrawal is finalized, the state can be withdrawn from the contract on-chain with an inclusion proof.

Availability

Side chain blocks can only be orphaned if they are invalid or build on an invalid block; valid blocks are guaranteed to progress the canonical side chain by construction. Therefore a side chain with merged consensus is available with respect to the main chain.

Consistency

Merged consensus is deterministic, objective, and uses no off-chain information, so the side chain is consistent with respect to the main chain.

Trustlessness Analysis

A layer-2 system is trustless

if state elements that interact with it are live

(cannot be locked forever/burned) and safe

(cannot be stolen). When using fraud proofs, the optimal limit is that state is live and safe unless a majority of main chain block producers attack the main chain to censor the transactions that prove fraud (which is identically our threat model).

State Liveness

Anyone may exit the side chain at any time, subject to a finite delay F

, therefore state is optimally live.

State Safety

There are two ways an invalid side chain block can be finalized and cause a state safety violation:

1. The main chain undergoes a majority-block-producer censorship attack for F

blocks, which is not a valid adversary under our model.

1. If no one

validates side chain blocks to produce fraud proofs. However, note that side chain block producers are incentivized to fully validate side chain blocks, lest they build upon an invalid block and have their bond burned. A single rational block producer assumption is sufficient to prevent this case. Importantly, any user that thinks this assumption is too strong and wishes to ensure the safety of their funds may run a fully-validating side chain node, as the system is permissionless.

Cost Analysis

The cost of including ~200-byte transactions (200 \text{ bytes} * 68 \text{ gas per byte}=13600 \text{ gas}

) may seem to severely prohibit scalability, but this is largely illusory. Early measurements found that storage operations are underpriced by at least two orders of magnitude, while more recent measurements found that storage operations are underpriced by at least three orders of magnitude compared to calldata

. EIP 2028 aims to bring this cost more in line with real costs, with suggestions of at least a 30-fold decrease in calldata

costs (giving 200 \text{ bytes} * 2 \text{ gas per byte}=400 \text{ gas}

per transaction). This would easily allow several thousands of transactions per second to be posted as calldata

at the current gas limit, on Eth 1.0.

Future Work

1. Investigate how to align incentives so that we don't need to post all transaction data all the time, but instead do things optimistically.
2. Research possible compression mechanisms so we don't need to post raw transaction data.

Intuitions

This scheme can be thought of as an optimistic execution engine on top of ordered data. The main chain (Ethereum) block producers are only responsible for consensus on ordering transaction data and processing proofs of incorrect execution. The actual execution is done separately, so can be parallelized, sharded (if multiple side chains are used), or use a completely different execution engine (such as eWASM, or a UTXO data model) than the EVM.