We can actually scale asset transfer transactions on ethereum by a huge amount, without using layer 2's that introduce liveness assumptions (eg. channels, plasma), by using ZK-SNARKs to mass-validate transactions. Here is how we do it.

There are two classes of user: (i) transactor, and (ii) relayer. A relayer takes a set of operations from transactors, and combines them all into a transaction and makes a ZK-SNARK to prove the validity, and publishes the ZK-SNARK and the transaction data in a highly compressed form to the blockchain. A relayer gets rewarded for this by transaction fees from transactors.

The system is managed by a contract, whose state consists solely of two bytes32

values representing Merkle roots: address book (A) and balances+nonces (B). A starts off as a Merkle root of 2^{24}

zero entries, and B as a Merkle tree of 2^{24}

(0, 0) tuples.

There are three types of operations for transactors: (i) registration, (ii) deposit/withdraw and (iii) sending.

To register, a user needs to provide a Merkle branch showing some index i, where either i=0 and A[i]=0

, or i > 0 and A[i] = 0 and A[i-1] != 0

. The Merkle tree is updated so that A[i]

now equals the msg.sender's address, and the Merkle branch is logged so that a client reading logs can get all of the data they need to create their own Merkle branches.

To deposit or withdraw, a user needs to provide a Merkle branch showing some index i, where A[i]

equals the msg.sender's address, along with the corresponding branch for B[i]

, and the amount they want to deposit or withdraw, D

(negative for withdrawals). The contract checks that B[i][0] + D >= 0

. If D > 0

, it verifies that (if the system is for ETH) msg.value == D * 10**12

(ie. the base unit of the system is 10^{-6}

ETH) or otherwise calls transferFrom(msg.sender, self, D * 10**12)

to the appropriate ERC20 token contract. If D < 0

, it sends the ETH or token to msg.sender

. The contract then updates the Merkle root so that B[i][0] += D

. Note that for efficiency, we can combine together the registration and deposit steps for not-yet-registered transactors.

To send, a user constructs the data: from address index (3 bytes), to address index (3 bytes), amount (ETH's ~100m supply requires 47 bits, so we can say 6 bytes, but most of the time <=4), fee (one byte floating point, top 5 bits are exponent, bottom 3 are mantissa), nonce (2 bytes). The user broadcasts (from, to, amount, fee, nonce) plus a signature.

A relayer can gather together many of these operations, and create a ZK-SNARK proving that, when processing all of the operations in sequence, at that start of each operation B[from][0] >= amount + fee

, B[from][1] == nonce

and that a valid signature is known from A[from]

, and then updating the Merkle root with B[from][0] -= amount + fee

, B[to][0] += amount

, B[relayer][0] += fee

, B[from][1] += 1

. A log is issued to alert users that the transaction is a payment batch transaction and they would need to recompute their Merkle tree witnesses.

The cost of a ZK-SNARK verification with the latest protocols is ~600000 gas, and we can add ~50000 gas for overhead (base tx cost, log cost, storage slot modifications…). Otherwise, each transaction included costs 68 gas per byte, unless it is a zero byte in which case it goes down to 4 gas. Hence, for a regular transfer, we can expect the marginal cost to be 68 * 3 (from) + 68 * 3 (to) + 68 * 1 (fee) + 68 * 4 + 4 * 2 (amount) + 68 * 2 (nonce), or 892 gas. In the best case, a relay transaction would consume a block's full 8 million gas, so the marginal costs could take up ~91% of the total costs, so the total cost would be < 1000 gas per transaction, a gain of ~24x for ETH transactions and ~50x for ERC20 transfers. In practice, it would be more at first, as while the system has low volume users would prefer faster smaller batches even at higher cost per transaction over slower larger batches, but once the system gets to high volume, it should be able to get efficiency levels close to this.

Note that anyone can be a relayer; there is no assumption of even an untrusted special "operator" existing. Because all the data needed to update the Merkle tree goes on chain, there are no data availability issues.

We can optimize further by allowing a relayer to (with ZK-SNARKs proving the operation) rebalance the tree, moving more frequent users into lower indices, changing the encoding for the value so that common values (eg. round numbers of ETH) are represented, and changing the nonce scheme so that, for example, details in the signature change every 100 blocks and so the nonce can reset every 100 blocks.