The Alaskan Pipeline (Inspired by Nervos' inspirational [NC-Max]) is a proposal for decoupling block propagation from witness propagation. Once these two things have been broken apart witness sizes have a drastically lessened impact on chain security. Paired with a mechanism for incentivizing witness propagation this proposal aims to solve two large outstanding difficulties with Stateless Ethereum.

Currently, transactions are immediately processed. By the time a block has been sealed all the transactions inside that block have been processed and the results of their execution are reflected in the state root. Alaska breaks transaction processing into a two-step process: inclusion and execution:

- Inclusion

: Each block commits to a list of transactions which are propagated along with the block. However, those transactions are not yet processed: their execution is not reflected in the state root and no receipts for them are included.[1] * There is a limit on how many transactions each block can include, but how that limit is determined is an open question (more on this below)

- There is a limit on how many transactions each block can include, but how that limit is determined is an open question (more on this below)

- Execution

: Each block commits to a list of transactions which are executed in that block. * It must execute transactions in the order they were included without skipping any transactions, starting with the oldest included transaction which has not yet been executed.

- After execution a transaction has been fully applied: that execution is reflected in the block's state root and list of receipts
- There is a cooldown window: transactions may only be executed if they were included at least $\eta$

blocks ago, where $\eta$

is a configurable parameter.

- There are limits on how few/many transactions blocks may execute, but those limits are also an open question.

- It must execute transactions in the order they were included without skipping any transactions, starting with the oldest included transaction which has not yet been executed.
- After execution a transaction has been fully applied: that execution is reflected in the block's state root and list of receipts
- There is a cooldown window: transactions may only be executed if they were included at least $\eta$

blocks ago, where $\eta$

is a configurable parameter.

- There are limits on how few/many transactions blocks may execute, but those limits are also an open question.

This is a fair amount of additional complication, why is it worth doing? Separating inclusion from execution introduces a delay which affords us the time to build and propagate witnesses for each transaction before it is executed.

Because included transactions have a total ordering and are never skipped, executed transactions have completely deterministic witnesses and network participants have $\eta$

blocks to disseminate those witnesses. If everybody is honest then block propagation will not need to include witness data because by the time a block has been sealed everybody already has the witness.

This decoupling does not add any delay to transaction latency. The result of transaction execution is completely determined once the transaction has been included, and participants such as miners will know exactly how the transaction will resolve. Receipts and state roots are delayed, which means there is a delay before light clients and stateless clients can have transaction execution proven to them, but Alaska will not slow down transactions for network participants who are willing to fully process blocks.

For Alaska to work we will need an additional network primitive peers can use to gossip block witnesses to each other, honest miners will gossip the witness for a block as soon as they know what the witness will be. Honest nodes will encourage miners share block witnesses by introducing a delay to block propagation: any node which receives a block for which they have not yet received a witness will wait for one second before gossipping that block to their peers

.

This rule ensures that any miner which does not pre-propagate witnesses risks having their blocks delayed which puts them at significant risk of having their blocks uncled which probabilistically loses them a significant amount of revenue. If a miner does pre-propagate witnesses they pay an additional cost in terms of bandwidth but for any reasonable witness size this is outweighed by the revenue they earn from having their blocks become canonical blocks.

This proposal does not change how much bandwidth Stateless Ethereum consumes, witnesses must still be propagated. However, Alaska propagates those witnesses during the time we are not bandwidth constrained: between block propagation. As a result, larger witnesses do not slow block propagation

.

That's the Great Alaskan Transaction Pipeline!

If we suddenly care much less about witness sizes then the only remaining blocker to a workable Stateless Ethereum is a solution to gas cost accounting. We don't even need to come to agreement on a witness format: Alaska keeps the specifics of witness outside of consensus meaning we can easily evolve the format over time.

#### Open questions:

- The obvious: What should \eta

be? How long should the propagation delay be?

- What should the rules be around the queue of unexecuted transactions? How can we prevent the queue from growing unboundedly? Any rule we use should ensure the next few block witnesses are determinstic.
- Miners are providing two separate services: including transactions and executing transactions. How should the fee be split between those two operations?
- How does Alaska interact with ReGenesis?
- How does Alaska interact with 1559?
- For Alaska to work witnesses need to be deterministic. This means opcodes like BLOCKHASH must act as if the current block is the block in which the transaction was included, not the current block during transaction execution. Is there anything else which might cause witnesses to no longer be deterministic?
- What does the transition strategy look like, how do we switch over to these new rules?

#### A proposed rule for bounding the size of the transaction queue:

This solution isn't very clean but it seems like a decent starting point: the transaction queue is allowed to grow until the total number of transactions is (\eta * blockGasLimit) / 21000

. Blocks must execute as many transactions from the queue as they can without going over the block gas limit (and without breaking the rule that transactions must have been included at least \eta

blocks ago. This rule means that witnesses for future blocks are completely deterministic: miners do not have control over how many transactions they process. It also means some transactions will be executed/proven twice: once to show that completely processing this transaction would cause the block to go over the limit; once more to execute the transaction in the following block.

#### Footnote:

[1]: This description is a simplification: Each block commits not to the set of transactions it includes but to the full queue of unexecuted transactions. Each block appends some transactions to the end of the queue of unprocessed transactions and commits to the new queue. This allows network participants to statelessly process blocks.