Abra-Ka-Dabra

- (A.K.D) A revamped NutBerry Protocol

To even better support most (rollup) system designs and requirements,

I propose a more abstract version of NutBerry that introduces the concept of ETH 2.0 Execution Environments (EE).

This makes it possible to allow for domain-specific rollups and arbitrary Layer-1 interactions inside EE's - on eth 1.0.

Let's start with the core protocol that provides the following functionality:

- Submitting Blocks, Blocks contain the address of the target EE

and arbitrary data.

- Submitting a Solution for a Block, a Solution is a 32 byte value type and usually represents the state of the EE.
- Finalizing a Solution, committing to the 32 byte value type and moving the chain forward.
- Disputing a Solution, marks it as invalid and requires that the corresponding block has to be validated on Layer-1.
- Validating a block, validates the block on Layer-1 and moves the chain forward.
- Token deposits, provides a interface for Token Deposits to a specific EE, this creates a new Deposit-Block

in the system. * This is required for state arrangement but not necessary the only way to get tokens into the system. Aka convenience.

- This is required for state arrangement but not necessary the only way to get tokens into the system. Aka convenience.

Note, Layer-1 is the Ethereum mainnet- and Layer-2 the rollup -chain.

Block

A Block contains arbitrary data and is designated to a given [#ee

](Execution Environment).

Each Block has a deadline

parameter, this is useful to give Blocks an expiration time, i.e the Block is skipped if it is not finalized until the deadline

.

- Anyone can submit, validate, challenge and finalize a Block.
- Submitting a Block requires a bond. The bond is used to repay finalization / validation gas costs.
- There can be any number of Block submissions inside a Layer-1 (root-)Block.
- Blocks are sequenced and are being processed in order.
- A dispute and/or challenge does not put Block submissions on hold.
- The deadline

parameter can not be larger than TBD

to avoid arbitrary Layer-2 chain congestion.

Block Solution

A Solution is a representation - or compressed view of the state from a Execution Environment. Any EE can use it as it sees fit, but it also doesn't need to support that, for example, if a EE does mint NFT's as course of action of a Block, it may disallow finalization and thus requiring that the Block needs to be validated on Layer-1 in order to mint the NFT and calculating the new state.

- Solutions for Blocks can be submitted in advance. This is limited for up to 256 Blocks to limit congestion attacks.

- Solutions can be disputed, disputed Blocks needs to verified on Layer-1 once all Blocks before the disputed Block are finalized.
- A Execution Environment must not support the quick finalization-interface for a number of reasons.

EE

A Execution Environment is a smart contract on Layer-1, it must support the interface for finalization and validating(state transition) of Blocks.

It may do arbitrary interactions on Layer-1 but must finish any given Block in a fixed time window and/or a maximus gas budget.

This allows for great flexibility and domain specific optimizations for data structure and application logic.

A EE can interact with Layer-1 contracts and other EE's, thus allowing great interoperability.

Consensus Mechanism

Consenus ought to be simple and assumes that honest participants are usually poor, thus requiring that Block Producers have to deposit bonds on Block submission.

- A Block submitter has to always supply a bond, this is used re-imburse validation and finalization costs for network participants.
- The left-over of the Block submitter's bond is returned after the Block is finalized.
- Requires a smaller bond on challenge

that gets burned or returned depending if the Solution was indeed wrong or not.

- Execution Environments are just smart contracts on Layer-1 and must implement the challenge

interface.

They can contain any arbitrary logic and can also interact with any contract on Layer-1.

They must complete transitions in a fixed time-window and/or within a maximum gas budget.

However, state transitions / challenges can span across multiple Layer-1 transactions.

Calls for finalization and validations of Blocks can also include arbitrary witness data, witnesses can also be included in the Block-data itself, if it makes sense for the application.

The detailed specification of the interface shall be determined in the first development stage.

Application Example

EE-1

We have EE1

that acts as a custodian smart wallet for ERC-20 tokens.

Users can deposit and exit tokens on Layer-1 and transfer tokens on Layer-2 via signed Transactions.

Other EE's can interact with user's funds if they are allowed to do so.

Note, this EE does only allow state changes inside a Block validation phase - from any EE, it can query this information from the Core Protocol contract.

EE-2

EE-2 is a simple board game that takes custody of users funds in each round of the game and re-distributes that value depending on the game logic.

Instead of implementing it's own custodian vault, this EE depends on the functionality on EE-1. That also means that any

User who is interacting with EE-1 and the Layer-2 Protocol has a single point of entry for exits and deposits of tokens (thanks Core Protocol and EE-1), and a streamlined user experience without having to sign three different transactions to move tokens around the system for each round in the game.

Pro

- Domain specific optimizations possible.
- Developers…
- … have more flexibility on functionality, UX and application performance.
- … only need to take care of the application and the state transition functionality. The NutBerry Protocol handles Consensus and provides a rollup foundation. Both on-chain and via client-side libraries.
- … getting familiar with eth 2.0 Execution Environment / State Transitions patterns.
- … have more flexibility on functionality, UX and application performance.
- … only need to take care of the application and the state transition functionality. The NutBerry Protocol handles Consensus and provides a rollup foundation. Both on-chain and via client-side libraries.
- … getting familiar with eth 2.0 Execution Environment / State Transitions patterns.
- Users can benefit from a more streamlined UX if the Protocol gets used in the wild.
- The Core Protocol has, in comparison, low complexity, thus it becomes easier to audit.

Contra

- This is not a magic tool to get rollup functionality in one click

Versus other proposals

Other projects additionaly provide a fixed application environment or functionality, while this is not the case here, the only interesting comparison are Consensus & Protocol Incentives. This post gives a good overview: https://medium.com/molochdao/the-state-of-optimistic-rollup-8ade537a2d0f .

Graph

Roadmap estimates - This is a Grant Request

This section is for the EF folks. However, don’t hesitate to comment on this as well.

First Stage - 4 Weeks

- Building a MVP and finding a good fit of parameters for the system constraints (Consensus), by simulation- and test-driven development.

Second Stage - 4 Weeks

- After settling on the concrete parameters of the System, pure development phase for contracts and client-side utilities.

Third Stage - 4 Weeks

- Quality Assurance and documentation phase, finishing the Core Protocol to get it audit ready.