The following is adapted from a talk given on Friday, April 9th

This is the road to Ethereum. Ethereum

today

is what eth2 developers are calling “eth1”, the current Proof-of-Work (PoW) chain that we all know and love. Ethereum

tomorrow

isn’t eth1, nor eth2, it’s just… Ethereum: a synthesis of eth2 efforts at deploying Proof-of-Stake (PoS) and sharding on top of today’s execution layer.

Word soup. Let’s backtrack. This is today.

Users want to do things (“transactions”), so they send their transactions to miners, who package them nicely into blocks and add their blocks into an ever-growing chain. Miners run the PoW consensus mechanism

[1]

to decide who’s next to add a block, and also execute transactions contained in the block to make sure they are valid.

PoW requires a large mass of hardware running intensive computation, incurring exorbitant energy consumption. We’re not going to talk about waste in the sense that most cryptocurrency critics intend (“why would we need a payment system that uses ’s energy??”) but really waste in the sense that if we can do something with a strictly smaller absolute number of kWh, we should.

So enter Proof-of-Stake, replacing large swathes of mining rigs with one smol server, and replacing miners with validators.

So what is 2.? What are validators validating exactly?

Consensus layer ensures everyone agrees on the correct data. Execution layer actually interprets this data, makes it meaningful. “Data” is any kind of interaction with the blockchain, e.g., deploying a contract, doing trades on an exchange, sending a payment etc…

The heartbeat of a blockchain is the introduction of a new block in the chain. When that happens, there is a

state transition

between the state before the new block and the state after. The

state

summarises the whole bunch of data that came previous, between blocks 0 and N.

For instance, if the state keeps a ledger with Alice and Bob’s balances, and the new block contains a payment from Bob to Alice for 10 coins, the state after the block will have the new balances. The dashed blob on the right of the “State after block N+1” bubble shows that a new block might either modify existing state (e.g., Alice and Bob’s balances) or create new state (deploy a new contract, or adding Carol to the ledger…)

The general gist of rollups and eth2 efforts tackle point 1. of our plan, decorrelating consensus and execution. So how’s that going?

Since December 2020, we have two chains running in parallel:

- At the top, the PoS-consensus chain (a.k.a., beacon chain)
- At the bottom, our ol’ reliable PoW-consensus+execution chain (a.k.a., “eth1”).

At the top, the PoS-consensus chain (a.k.a., beacon chain)

At the bottom, our ol’ reliable PoW-consensus+execution chain (a.k.a., “eth1”).

They run parallel, but they talk to each other. Well, it’s pretty one-sided at the moment…

To become a validator on the PoS-consensus chain, a user on the PoW-consensus+execution chain locks up 32 ETH in the deposit contract, which is automagically transposed to the PoS-consensus chain. Once the validator is activated, they can start validating and receiving rewards.

This one-sided conversation hopefully won’t last long. The Merge bridges the gap between the PoS-consensus chain and the PoW-consensus+execution chain permanently, allowing validators to produce blocks for the execution layer. After the merge, we therefore have two chains:

- The same PoS-consensus chain
- A PoS-backed-execution chain (f.k.a, “eth1”)

The same PoS-consensus chain

A PoS-backed-execution chain (f.k.a, “eth1”)

When this happens, validators of the PoS-consensus chain will finally be able to redeem and withdraw their earnings and deposits, sending them back onto the PoS-backed-execution chain.

2

All that in itself gives us the move from a PoW environment to a fully fledged PoS environment. But notice that it doesn’t quite increase the

bandwidth

of the chain—how much execution a block is packing—, which is the goal of

sharding

. Before we get to that, let’s talk about other ways to scale execution if we only have one execution chain.

Enter rollups, one of the myriad scaling solutions available, but perhaps the one offering the most optimal trade-offs from a protocol design perspective. The idea is simple: synthesise state transitions by committing

on-chain

to data necessary to

recreate

the execution of the state transition, and push execution off-chain. If anyone has a problem with the execution result, or if someone forgets to execute in the first place, the data is there for all to process. Permissionless!

To be more precise, the data necessary for execution (transaction inputs) is separated from its containers (transactions) and “bundled” in a space-saving manner. Meanwhile, rollups run besides the execution chain, committing to the data and executing it.

To “get on a rollup”, users deposit assets in the rollup’s contract on the execution chain (“eth1”), then they do stuff on the rollup, such as trade them with other assets also on the rollup. Once they are done, users withdraw assets back from the rollup onto the execution chain.

Voilà

.

What is the alternative to rollups? Well, what if instead of having these perpendicular yellow chains, we had a bunch of parallel red chains? As in, what if the eth1 chain was “duplicated” and several instances were running side-by-side?

The issue here is how we deal with several execution chains running in parallel.

What if something happens on a chain that another needs to know about?This is pretty much the thorny problem of

sharding

, or dividing execution between multiple chains.

“Rollups aren’t so different!” you will say, and in essence it’s true. When you want to do something on another rollup from one rollup, the same thorny questions arise. The point however is that several rollup designs exist and the space of solutions is wide and mostly unexplored. So why not let rollups bootstrap experimentation before enshrining one approach at protocol-level?

This leads us to……

So you heard about block space scarcity did you? Rollups do need to publish their data, and eth1 block space is rare! Plus as we discussed, cross-shard stuff is hard. So to make lemonade with our lemons, why not use the shards to hold the data that rollups critically need to publish? With 64 shards, that’s 64 times the bandwidth available now, and possibly more as one shard block would likely hold a much higher amount of data than the eth1 chain blocks currently can.

It should be emphasised that this doesn’t mean execution at the shard level is forever excluded. The

rollup-centric Ethereumis a short-to-mid-term way forward, until (for instance) better cryptographic primitives are found to guarantee correct execution sharded across multiple chains. It's all very fascinating and should keep a lot of people busy for a long amount of time. In the meantime, rollups are the way.

There's plenty more work on that side! First let's not forget the Merge and data shards are extremely complex endeavours with multiple teams working on either or both at the moment. But even on the rollup side, there are still some extremely juicy questions that are left to explore, and these are but just a few.

- Mass migration of users to and from rollups is a very cool conceptWhy should you take your own car to go between Layer 1 (eth1) and Layer 2 (rollups) if you have frequent enough public transport shuttling you to and from? What are the economics of this?
- What if you'd like to do something on a neighbouring rollup, because they have something cool that yours doesn't have? Should you necessarily move from your rollup to L1, and then from L1 to the neighbouring rollup? This seems rather wasteful.
- Rollups

are

a massive improvement in bandwidth for current operations, there is no question about this, yet rollups aren't the infinite highways users seem to expect. There is still a lot of people wanting to do a lot of stuff, sometimes at the same instant! So rollups will naturally have to deal with congestion, but contrary to the L1 protocol enshrining a particular congestion market (soon,

eip1559), rollups have a much larger design space to explore.

- Speaking of congestion, and this is more protocol-specific, but we'll also see

eip1559 playing traffic copto regulate how much data is published on every data shard, ensuring validators can cope up with the amount. If you thought one eip1559 mechanism was cool, wait until there are 64 running. So where

should

rollups publish their data? On one shard only to make it accessible without shard-hopping for whoever is listening to it? On multiple shards to take advantage of the planned

shard staggering, where shards publish a new block in turn, such that you are never more than a couple hundreds of milliseconds away from a new block to publish your data, ideal for quick-finality applications?

Mass migration of users to and from rollups is a very cool conceptWhy should you take your own car to go between Layer 1 (eth1) and Layer 2 (rollups) if you have frequent enough public transport shuttling you to and from? What are the economics of this?

What if you'd like to do something on a neighbouring rollup, because they have something cool that yours doesn't have? Should you necessarily move from your rollup to L1, and then from L1 to the neighbouring rollup? This seems rather wasteful.

Rollups

are

a massive improvement in bandwidth for current operations, there is no question about this, yet rollups aren't the infinite highways users seem to expect. There is still a lot of people wanting to do a lot of stuff, sometimes at the same instant! So rollups will naturally have to deal with congestion, but contrary to the L1 protocol enshrining a particular congestion market (soon,

eip1559), rollups have a much larger design space to explore.

Speaking of congestion, and this is more protocol-specific, but we'll also see

eip1559 playing traffic copto regulate how much data is published on every data shard, ensuring validators can cope up with the amount. If you thought one eip1559 mechanism was cool, wait until there are 64 running. So where

should

rollups publish their data? On one shard only to make it accessible without shard-hopping for whoever is listening to it? On multiple shards to take advantage of the planned

shard staggering, where shards publish a new block in turn, such that you are never more than a couple hundreds of milliseconds away from a new block to publish your data, ideal for quick-finality applications?

Many thanks to Danny Ryan and Sacha Saint-Leger for their suggestions.

I hear that PoW is not a consensus

algorithm

, but I think it’s fine to describe it as a consensus

mechanism

if we overload its definition a bit.

PhiGonoted on

Twitterthat deposits won’t necessarily be withdrawable

right

after the Merge (when PoW is turned off). It’s true that the current Merge plans focus more exclusively on the turning off part, with withdrawals a separate but related issue.