Keynote: [title redacted]

(0:08 - 0:45)

This is a proposed redesign of the consensus layer that incorporates all of the latest and greatest ideas from the research roadmap, and the goal is to try and transition in a safe and fast manner from the beacon chain that we have today to this beam chain which is much, much closer to the final design of Ethereum. Now, before I share more information, I have two disclosures, disclaimers. Disclaimer number one is that this is just a proposal.

(0:45 - 8:08)

This is my proposal, and the proposal will only go forward if there is rough consensus with it going forward. And the second disclaimer is that there is no new token, there is no new network, and we're reusing the same ticker, and Vitalik was very, very clear as to what this ticker is. Now, in the rest of this talk, I want to take what may sound like a totally crazy idea and convince you that actually it's maybe not so crazy, that it might be a reasonable proposal to put on the table to completely redesign the consensus layer.

And first, I want to tell you a little bit more about the beam chain and what the big picture vision is all about. So the scope of the beam chain is specifically the consensus layer, and I'm excluding the data layer with the blobs and the execution layer with the EVM. And the reason is that the blobs and the EVM are directly consumed by applications, and there is a need to be forward compatible.

And so the opportunity to modify these two layers is rather limited. On the other hand, the consensus layer is not directly consumed by applications, and so there is a big opportunity to shake things up a little bit there. Now, why am I proposing now to do this massive redesign of the consensus layer? And it basically boils down to the fact that the beacon chain is kind of old.

The spec was frozen five years ago, and in those five years so much has happened. In particular, we have a much better understanding of MEV. Five years ago, we were extremely naive when it comes to MEV, and since then we've seen the market really blossom and grow.

And we also have a much better understanding of mechanisms that can help us mitigate the negative externalities of MEV. Secondly, from an engineering standpoint, we have this wonderfully powerful technology called Snarks, where there's been plenty of breakthroughs. In particular, we've seen Snarks become orders of magnitude faster over the last five years, and we've seen the advent of ZKVMs.

ZKVMs are amazing technology that allows any programmer in the world to make use of

this very powerful technology without having to be an expert in cryptography or in Snarks. And then finally, with the benefit of hindsight, we now know what the mistakes we made with the beacon chain. And we have a bunch of technical debt which is extremely sticky and tends to pile on over time, and maybe now we have an opportunity to clear this technical debt.

So I'm suggesting putting the greatest and latest of the consensus layer roadmap in the beam chain, so I guess it might be worth spending a little bit of time looking at what exactly is in the consensus layer roadmap. There's basically nine different items, and I've categorized them in three different buckets. Block production, staking, and cryptography.

So the block production has to do with MEV. Right now, we have a lot of centralization at the builder and relay level, and one of the things that we want to do is have inclusion lists that dramatically improve censorship resistance. Once we have censorship resistance with inclusion lists, we'll be in a position where we can cleanly decouple the validators from the block production pipeline, and this is called a tester-proposer separation, and there's ideas like execution auctions.

And then the final item here in the block production bucket is faster slots. Maybe we can take the 12-second slots that we have right now and shrink them, all while preserving the invariance that if you are on a home internet connection, you can participate as a validator, as a first-class citizen, even if you are in Australia with a high-latency internet connection. Second bucket is staking.

The researchers, I think, have come to consensus, broadly speaking, that the current issuance curve is kind of broken, and that there is an opportunity to improve the health and long-term outcomes of Ethereum by changing it. The second item in the roadmap here under the staking bucket is this idea of dramatically reducing the total amount of ETH to become a validator, from 32 ETH down to just one ETH, and there are ideas like Orbit that have been circulating around recently. And then finally, this is an idea that has been talked about for many years, is single-slot finality.

Can we dramatically accelerate the process of Ethereum gaining finality? And then the final bucket has two big-ticket items. Number one, can we snarkify the whole of the consensus layer in real-time using reasonable hardware? And then finally, can we make the cryptography that is securing Ethereum sustainable for the next decades and centuries, and make it post-quantum secure? Now the coloring here is meant to suggest whether or not the item in the roadmap can be done easily, incrementally, or whether it can't. So the four green items here in the top left corner are items that I think can be done and should be done in incremental forks on the beacon chain.

But then when we get rid of those, we're left with big-ticket items, the red items, that I would argue are best done in a more holistic way. So take chain snarkification, for

example. In order to achieve real-time proving of the beacon chain with reasonable hardware, we're going to need to change the hash functions.

We're going to need to serialize and mercolize the states. And this is a massive change to the beacon chain. And so maybe there's an opportunity, if we were to make such a change, to change other things at the same time.

And then a similar thing can be said for the bottom two red boxes, faster slots and faster finality. The truth is that five years ago, our mindset was security first when we designed the beacon chain. And performance was not really a main consideration.

And with the benefit of hindsight, we found that there are designs that preserve all of the security that we want and at the same time pick some of the low-hanging fruit that is available to us to improve performance. Okay. So this slide here is trying to show the mapping from the consensus layer roadmap that I showed you and Vitalik's broader roadmap.

(8:09 - 9:55)

We have some items that are classified under the merge, some that are under the scourge, and then a couple that are under the verge and the splurge. And the whole point of this slide is to communicate the fact that the beam chain is not about changing the roadmap more so than it is about identifying a specific subset of that roadmap and accelerating it and putting a mimetic wrapper around it. Now, one thing that is new in the consensus layer roadmap here is the faster slots.

And the reason is that the discussion around faster slots has happened this year in 2024, but Vitalik's roadmap diagram was only last updated in 2023. Now, in addition to being able to potentially accelerate these big ticket items, there's a lot of technical debt that I mentioned that can be cleared out. If we have single slot finality, we no longer need epochs.

We can just have slots. The deposit contract that we have right now is kind of crazy, and it's a remnant of the merge. And things like the sync committee is infrastructure that we won't need going forward if we have real-time snarkification of the beacon chain.

And the list goes on and on and on. And this is an opportunity, as I mentioned, to just clean it up all in one go. And if you're interested in learning more about some of these beacon chain mistakes, last year I did a whole talk where I talked about 20 or so different mistakes that we made in the beacon chain design.

(9:57 - 17:55)

So this is the big picture view of how we have done upgrades to the consensus layer since Genesys. So in the bottom left here, you can see we did Genesys in 2020, and

since then, it's been an extremely regular pattern. Every single year, we've had a new fork, and every time we had a fork, we made one incremental change to the consensus layer.

So in 2021, we added sync committees. In 2022, we did the merge. In 2023, we added withdrawals.

And then proto-dangsharding. And soon, in 2025, we're going to increase the max effective balance. Now, what I expect will happen is that over the coming years, we will continue doing these incremental forks, and we will be picking up the low-hanging fruit that was marked as green bubbles in the road map, in the top left corner of the road map.

But then we're going to kind of hit a wall. And the reason is that once we've picked all of the low-hanging fruit, we'll be left with all of the big-ticket items that are much more difficult to do in an incremental fashion. And this is where the beam fork happens.

The beam fork is an opportunity to have a quantum leap forward in terms of upgrading the consensus layer all in one go. And one way to think about the beam fork is as a batching opportunity. We're batching multiple upgrades into one single fork, and this has benefits both technically and from a governance standpoint.

And one way to think about this batching opportunity is as ossification accelerationism. This might sound like an oxymoron, but the basic idea is that we want Ethereum to go in maintenance mode as soon as possible. And right now, there's this tension because we know that there's these big-ticket items that require a fundamental re-architecturing of Ethereum, and the more we drag it along, the further Ethereum will be in a position where it can comfortably ossify.

Okay, part two. I'm going to try and highlight some of the technology that is going into the proposed beam chain. And the way that I would frame this is basically in eras of Ethereum consensus.

Initially, we had the proof-of-work era of Ethereum consensus, and then we moved to proof-of-stake. And now, we're potentially entering this ZK era of Ethereum consensus. And in the ZK era, we would be making heavy, heavy use of snarks as a technology.

So one place where we'd be making use of snarks is snarkifying the entire beam chain, the entire consensus layer. And this is where the ZKVMs become extremely, extremely handy. So imagine that you have implementations of the beam chain in various high-level languages, in Rust, in Go, for example.

What you can do is compile these high-level languages down to bytecode that the ZKVMs will understand, and get snarkification without having to worry about the details of the snarkification process. Now, one thing that I do want to highlight is that the only

part that needs to be snarkified is what's called the state transition function, which is basically the crystalline core of what it means to be a consensus client. All of the infrastructure surrounding the state transition function, for example, the networking or the syncing or the caching optimizations or the fork choice rule, none of that has to be snarkified.

And ultimately, the state transition function is a small subset of what it takes to build a client. And what we've seen recently in the last couple years is RISC-V become the de facto industry standard for these ZKVMs. So RISC-V is an instruction set, and basically you can take high-level code and compile it down to RISC-V.

And we've seen seven different companies provide these RISC-V ZKVMs. You may have heard of RISC-0 and SP1, and the list goes on and on. Now, one little side note here is that this exact technology, which is extremely powerful, can also be used at the execution layer for the EVM.

But that is a completely different story to the beam chain story. It's one which is extremely exciting because it means that we can dramatically increase the gas limit, as well as vertically scale if you're in layer one. But this is going to have to be for a different talk.

The other place where we make heavy use of snarks in the beam chain is with aggregatable signatures. We want to have post-quantum aggregatable signatures, and the proposal here is to use hash functions. Hash functions are post-quantum secure, and you can use that as a basic building block to build your cryptography.

So we would have hash-based signatures that are produced by the validators, by the attesters, and we would also have hash-based snarks where you can take many, many thousands of signatures and compress them down to just one proof. And with these two combined, you get a hash-based post-quantum aggregatable scheme that could be used for Ethereum. And one little nice detail is that this aggregatable scheme is infinitely recursively aggregatable, so you can take aggregates of aggregates of aggregates, which is something that we can't really do today with BLS.

So it's much more flexible. And the reason why I have this proposal here today is because in the last few months, the progress in performance of snarkifying hash functions has gone through the absolute roof. So for those who are in the know, we're now able to prove on a laptop, so this benchmark here was done on a laptop CPU, a MacBook Pro, we're now able to prove two million hashes per second, which is an astounding amount of hashes per second.

And this means that this hash-based proposal has the potential of being extremely performant for the beam chain. Now, in addition to the very powerful ZKVMs and snarks that we would be using, I also want to highlight that to a large extent we would also be

reusing existing infrastructure. So the networking libraries, libp2p, the serialization libraries, simple serialize, all of that can be reused as is today.

Same thing for the PySpec. PySpec is the framework that we use to write the formal specification, sorry, the Python specification, and the unit tests, and we can also reuse protocol guilds, all of which is infrastructure that didn't really exist when we started with the beacon chain. And the same thing can be said about teams.

When we started the process of the beacon chain, there was no team, there was no consensus client teams. So the five consensus client teams that we have today, this is manpower that can be reused and doesn't have to be rebuilt. And in addition to that, we have dedicated teams that were put in place for the merge, for example, EVE Panda Ops, which does DevOps, and the security team within the EVE Foundation.

(17:56 - 20:49)

And I also want to give a shout out to the incentives team and to the applied research group, which are also teams that all of this infrastructure didn't exist, and we can just reuse it for free. Okay, final part. So here I want to try and highlight some of the next steps and how I see the future.

So one possible outcome here is that, starting from 2025, we start the specing process. So this would be something that a small group of researchers would do, and it would take maybe a whole year. And then in 2026, the building process can happen where clients would start writing production-grade code, and then in 2026, EVE 7 would start an extremely thorough testing process to make sure that this is all production-grade and safe to deploy on mainnet.

Now, the next step for me as a researcher would be to start writing the executable spec, which I call the executable roadmap. And the idea is to take the pixels that we have of the roadmap combined with the hundreds of thousands of words that have been written on the EVE research and in academic papers, as well as all of the ideas that lie in the minds of researchers, combine all of this, and extract the core essence, which would be this executable spec. And ultimately, it would be a very small document, roughly 1,000 lines of Python code.

Now, one exciting thing for me here is that the BeamChain, assuming that there is rough consensus to go in this new direction, would be a fantastic onboarding opportunity to bring fresh blood into the space, especially for the consensus clients. So today, we have consensus clients in North America, in Europe, in Oceania, and I'm very pleased to be able to announce today that we have already new consensus client teams that are willing to build the Beam clients. So we have one which is based in India, which is called the Zim Team.

This is a Beam client written in Zig. And then we have Lambda Class in South America that has signaled an interest to also write a Beam client. And so if you will also get involved, and we're going to need a lot of excellent talent, we're going to need speckers and networking experts and coordinators and cryptographic experts and client devs, please reach out at this email address and Beam up with us on this new adventure.

(20:49 - 21:38)

Thank you so much. All right, let's hear it one more time. Come on, a big round of applause.

That was a fantastic presentation. Really, really good. Really appreciate it.

And I'm sure the audience appreciated it as well. Now is the time to ask your questions, or if you see an interesting question, please upvote it. Now, for those of you in the hall, don't leave yet, okay? Just give us some time because the volunteers need to clear a bit of the space.

There are still a lot of people outside, so at the end of Q&A, don't move. Stay where you are because our volunteers need time to clear the hall later, okay? So just stay where you are, and I'll instruct you at the end of Q&A. All right, Justin, let's look at the first question with the five votes right on top.

(21:38 - 24:09)

Deploying many changes at once is risky in that if any one change gets delayed or has a bug, they all get stuck or rolled back. How hard have you tried to break them down in incremental batches? Okay, great question. So there's many things that I've tried to do to try and de-risk the bean chain.

The first one is to encourage all of the incremental upgrades, remember the four green items in the roadmap, that could be done ahead of time. And by going through the process of the incremental upgrades, we're going to go through a lot of the lessons and a lot of the hard work, and discover the challenges there and then. The other thing that I'll mention is that the bean chain is really only about changing the core of the client, which is the state transition function.

And a lot of the potential opportunity for bugs, whether it's a networking layer or the localization libraries, or the fork choice rule, a lot of that infrastructure can, to a large extent, be reused. Another thing that I want to do to de-risk the bean chain is make sure that all of the existing clients are on board, because ultimately these are 50 men and women that have many, many years of experience building consensus clients, and they know what a lot of these pitfalls are. And then another thing that I would say in terms of de-risking the bean chain is that we will have an especially intensive testing period, which could last, for example, a couple of years.

This is what I had in the Strawman timeline. And during that time, we would do many, many rehearsals on death nets and test nets so that we could grow confidence that there isn't a bug in this big upgrade. Excellent.

Thank you for the answer, indeed. Testing is very essential when it comes to the blockchain and updates. Now, the question that I've seen has gotten the most votes of any of the rooms so far.

20 is the number to beat. Are you OK, Justin, with people calling this Ethereum 3.0, the meme layer of Ethereum, wants to do this? OK, great question. So in my opinion, the moniker Ethereum 3.0 is not appropriate.

And the reason is that the bean chain is only about the consensus layer. It's not about all of Ethereum layer one. It's just a consensus layer.

It excludes the data layer, Dankshadeng, and it also excludes the EVM. And there's lots of exciting stuff happening at these layers of the stack as well. So this is why I've tried to avoid the Ethereum 3.0 moniker.

(24:09 - 25:21)

All right. I mean, it's a fair, fair, fair answer as well. But people will do what people do best and they'll call it Ethereum 3.0. And we'll get a few new coins and people ask all the exchanges, will my Ethereum change and all of that.

But let's go to the next highest voted question. Which ZKVM will be used for beam chain as there's no standardization of ZKVMs? Okay, fantastic question. So the amazing thing about the proposed design, and actually this is an idea from Vitalik, is that the snarkification would happen off-chain and would not be enshrined in consensus.

So what that means is that every independent validator can choose which ZKVM they prefer. So they can use any of the seven RISC-V ZKVMs, but they can also use non-RISC-V ZKVMs. And ultimately, by having this off-chain snarkification, it means that if there's a bug in one of them, it's very easy to fix.

If there is a performance optimization in one of them, you just update your client and you get this performance optimization. And also, it means that we don't have to pick winners and we don't have to add a lot of complexity on-chain. All of this is off-chain complexity at the social layer.

(25:22 - 25:45)

All right, thank you. We got 30 seconds. You want to take the last question at the top? Any way to accelerate that timeline? We moved.

Okay. Okay, well, let's take the first one. Okay, they voted for 18, 19.

Okay, let's do that one. Why are ETH researchers obsessed with lowering issuance? It's... Come on, guys. We need the DAO to agree.

(25:46 - 25:53)

Where is the consensus? Okay, Justin, take whichever question you want. Okay, I'm happy to answer all of the questions later on.