**EPISODE 1484**

[INTRODUCTION]

**[00:00:00] ANNOUNCER:** Breakthroughs from labs that are exclusively or mostly focused on research tend to stay buried as papers. Sometimes, the lag is natural. The research is far ahead of its broad applicability. But we find that there are hundreds of breakthroughs useful long before they reach users. It can take many years, even decades before breakthroughs are realized in products that improve people's lives.

This is massively inefficient. Research has to be coupled closely with development and deployment so that we can iterate through the cycle quickly and build good products that actually solve problems for people.

Protocol Labs is a research, development and deployment lab for network protocols. Their projects include IPFS, Filecoin, Libp2p and many more. Jeromy Johnson is a distributed system researcher focused on incentive mechanism design and trustless distributed system scalability. He joins us today to share how Protocol Labs is spearheading the innovation in Web3.

[INTERVIEW]

**[00:01:07] JM:** Jeromy, welcome to the show.

**[00:01:09] JJ:** Hey, thanks.

**[00:01:10] JM:** We've done a few shows on Filecoin, and Protocol Labs and IPFS. The thing I'd like to start by talking about is usage. The necessity for a totally decentralized storage system is pretty obvious when it comes to thinking about Web3. The basics of Filecoin and IPFS have been in place for a while. And I guess I'd like to get a sense for the abstractions as of 2022, the abstractions that are being built on top of that storage primitive and what you're seeing in the wild in terms of actual applications.

 

**[00:02:01] JJ:** Yeah, I think it's really interesting because a lot of people, the majority of people interacting with IPFS and Filecoin, really just want something that feels like what they're used to. There are a number of people who are really diving in. They're running their own infrastructure. They're doing their own peer-to-peer CDN, deploying things in different locations around the world. But most people just want to be able to say upload file, make a backup, and I can download it later.

And, like, towards that we've been building several different frontend type interfaces and API interfaces to Filecoin and IPFS. And notably, the one I've been working on is called Estuary. And the way these manifests is just a very simple HTTP upload API. We handle taking the file in from the user and then shuttling it off to Filecoin. And this does mean that for some period of time there is like a dependence on a centralized service for the data being uploaded. But once it's there and on Filecoin, it's not like owned by the service. It's stored into centralized network. And you can retrieve it yourself. You can maintain the data yourself if you so choose. And that seems to be you know getting a lot of people excited about it because it makes it feel just like throwing things onto S3.

**[00:03:23] JM:** Is it API compatible with S3?

**[00:03:26] JJ:** It's not right now. We've been thinking about adding a portion of the API that mimics the S3 surface area. But I think that it totally could.

**[00:03:37] JM:** Is there a potential for the IPFS storage layer to be cheaper bucket storage than S3?

**[00:03:45] JJ:** Oh, absolutely. I think that's the biggest thing we've noticed about Filecoin is that the bulk storage prices that we are getting with the storage providers are significantly cheaper than S3. Like, effectively free for most sizes of data at this point. I expect that to normalize a bit as the market expands. But given the block reward subsidy of Filecoin, miners, storage providers just want more user data. And they're willing to give like crazy discounts of 100x to 1000x cheaper than S3 just to get that data on board.

**[00:04:23] JM:** Walk me through the lifecycle of a right to IPFS in 2022.

**[00:04:30] JJ:** Yeah. To IPFS specifically, it's a right to IPFS. Is kind of an interesting concept. Because IPFS mostly weirdly doesn't describe a storage protocol. It describes a file transfer protocol. And so, the most common thing that people see when they are like adding content to IPFS is they're either using their own IPFS node, which is effectively just indexing it like you're making it available over the Internet and not actually uploading it to anywhere. Or you're using a pinning service, which is putting it into somebody else's IPFS node.

And this is different than the flow for Filecoin, where, with Filecoin, you're actually – If you want to do it manually, you are selecting which storage providers you would like to host your data probably by using a reputation system or by having some pre-selection, which providers you want to use. And negotiating a price and making an on-chain contract with them for the storage of your data. And that ends up with your data ending up being stored and secured on the blockchain. As opposed to IPFS is really – The default doesn't necessarily store your data. It makes your data available, if that makes sense.

**[00:05:53] JM:** Yeah. Maybe you could go into that in a little more detail. So, if I want to actually treat the storage medium that IPFS provides as actual usable storage for my application, maybe you can talk through the API that I'm actually going to be using.

**[00:06:13] JJ:** It really depends on what your goals are. So, the average case is going to be somebody is using a pinning service of some kind, whether this is Pinata, or Estuary, or Web3 storage. This is effectively outsourcing the actual persistence layer of your data to somebody else. And that really feels like a basic file upload. Or it looks like you are adding your data to IPFS locally and then asking one of these services to pin it for you.

The main thing there is you can entirely self-host it. You can run your own IPFS node on your server and you can add your data to it. But then if nobody else is hosting the data and your node goes offline, then it becomes unavailable until your node comes back online.

However, if you know you can run your own server and pin it somewhere else and then you gain the performance of, if either of them are online, your data is online. And you get the ability to set your own replication for the availability of your data, which is a really interesting concept. And

anybody – Like, say, the stuff you're uploading is interesting to other people, they can choose to also provide a copy to the network. And that makes it even more resilient.

**[00:07:36] JM:** So, the files that I would upload to IPFS, how many times are they replicated? Or can you describe the way that the replication works and explain – I mean, we've talked about the incentive design on previous episodes. But maybe you could talk about how the incentive design has changed over time.

**[00:07:57] JJ:** Yeah. So, IPFS itself doesn't actually provide replication. As I mentioned, like, IPFS, just speaking about IPFS a protocol, is a way of moving data from point A to point B efficiently, or moving data from somewhere to somewhere else efficiently, and referencing that data.

With IPFS, you can specify, like, you can ask other parties to replicate your data for you. And that's kind of on a one-off basis. You can pick a pinning service for that. But on the Filecoin case, you, as the user, are in charge of the replication. So, you can say, "Make deals with five storage providers, or 10 storage providers, or only three."

And in the default like bare bones, you're in control of your own destiny case. That looks like you uploading the file to each of the storage providers that you would want to use for replication of that data. You can also use an intermediary that manages that replication for you. And it really just depends on how in control of your own data do you want to be.

**[00:09:09] JM:** Okay. If I do want to have that storage replicated, let's say, 10 times, and I want to pay on an ongoing basis to have that replication factor, how am I doing that?

**[00:09:23] JJ:** Yeah. There's a number of different ways you can go through the default, like, lotus Filecoin client and select a storage provider or 10 storage providers. And then individually make storage contracts with them. And then you're effectively done. The contracts are for kind of a fixed term. So, at the end of the term, you have to renegotiate the deal with current pricing. And so, you'll pay for the storage effectively once per year, a year and a half depending on how long the initial contract you made.

Beyond that, one of the things we're excited about being able to do is we are going to be shipping a virtual machine in Filecoin soon. And you'll be able to have a smart contract on Filecoin that manages all of this for you. Whereas, right now, you kind of have to pay attention to the individual replication of files and worry about renegotiating deals over time. Once the virtual machine upgrade happens, you'll be able to have a single smart contract that can manage all the replication for all of your storage, which should dramatically simplify the process for the average user.

**[00:10:34] JM:** So, you said I'm paying for it. Who am I actually paying?

**[00:10:37] JJ:** You're paying the storage providers directly. When you find a storage provider on the network, you reach out to them through the peer-to-peer network and make a deal directly with them. It gets brokered by the blockchain and enforced by the blockchain. Because the actual storage proofs get periodically published to the chain by the miners as they're showing that they have the data. And as long as they are correctly showing they have your data, they continue to get paid. And if they lose your data, then the remainder of the funds get refunded to you and the miner is penalized for that.

**[00:11:18] JM:** How do my storage providers prove that they have my data?

**[00:11:20] JJ:** Yeah, this is actually kind of the most fundamentally fascinating piece, is they are proving that they have – Basically, on-chain, the hash of your data gets registered. And it's a very specific hash. It is a very SNARK-friendly hash. And so, once per day, miners are submitting a zK-SNARK proof that they have probabilistically all of the content for all of the hashes that they've committed to on-chain. And so, effectively, you have once per day the miner is submitting a proof to the chain that they could not have produced if they didn't have your data.

**[00:11:59] JM:** And what are the mechanics behind that proof system?

**[00:12:03] JJ:** For the uninformed listeners who don't know what SNARKS are, a SNARK is a way of doing a zero-knowledge proof. Effectively, running some code over potentially private data and being able to hand somebody a succinct statement that proves the thing you did was

done correctly. So, you can use this to prove knowledge of something. You can prove that you know a secret without revealing what the secret is.

And the way we use this in Filecoin is we have all of – You can think of each proof as addressing all of the data that's there. And we use randomness from the chain to select pieces of that file of the corpus of data to the miner to show that they have. And the usual way you would do this is you take all of the data and you build a Merkle tree of it, which is kind of hashing each of the small components of the data together recursively until you get a single root hash that references the whole thing. And now, to prove that you have, the data you can provide a path of the hashes through this like Merkle tree.

And the reason this convinces the person has the data, is that they could instead store all of the hashes in order to respond to the proof. But then they would also, at that point, still have to have the very bottom leaf that actually contains the raw data. And so, doing so, you could potentially cheat this occasionally. But doing so requires storing significantly more data and is impractical.

So, the miner ends up creating this proof that is fairly large on its own, because it is a piece of this giant hunk of data that they're storing, terabytes, or petabytes, or larger of data. And then using these zero-knowledge proofs, they create a proof that verifies that this Merkle tree is correct.

And the cool thing here is the SNARK proves that they verified it correctly, which means that it is correct. And in doing so, we're not really using the SNARKs to hide information. We're using the SNARKs to compress the information that the proof is correct. I'm trying to think if I'm explaining this properly. But how does that sound?

**[00:14:33] JM:** It sounds good. Did the original version of proof of replication – Did the original version use SNARKs?

**[00:14:42] JJ:** SNARKs the original version of replication did use SNARKs. But things that came before it for earlier versions of Filecoin did not. Early versions of Filecoin, like, the versions that were not implemented, they wanted to use effectively passing of data around and just using basic retrievability proofs to prove that miners had the data.

And one of the big problems with the earlier constructions is that they required a lot of bandwidth consumption to be passed around. So, you had to move tons of data around to prove that you had it. And that proved to be fairly impractical, which is why we ended up having to go back to the drawing board a little bit to figure out how to build this system and gain the same level of assurance over the data without the bandwidth costs. And so, SNARKs came in here as like a really amazing tool to not only like hide the data, but to compress computation effectively.

**[00:15:41] JM:** The reason I asked this, I was curious about upgrading a decentralized system. Maybe you could talk through – We could get into some of the software development details of how a protocol actually gets updated. We've done lots of shows on continuous delivery. Can you talk through how an actual software update to a decentralized storage system works?

**[00:16:05] JJ:** I think there's two different cases here, which are both really interesting. The case of IPFS is that it's not strictly necessary that everybody runs the exact same version of the software. The way that we've designed IPFS and the underlying components of like Libp2p, IPLD and multiformats allow things to be self-describing and protocols to be self-describing so that we can continue to have very seamless backwards compatibility across versions while new things happen.

So, when we're shipping a new version of IPFS, say, one of the more recent versions where we switched the default networking protocol to QUIC from TCP, this was very seamless. Because by the time we actually shifted it over, a lot of the previous versions had support for QUIC. They might not have necessarily chosen it as their default, but they had support.

And so, even if not everybody in the network updated, you still had a lot of connectivity. The new nodes still supported the old protocols. The old nodes can talk the new protocols. And then the old, old nodes that only supported the TCP-based protocols, they are still able to communicate with the newest nodes because the newest nodes have the fallbacks. And since everything is negotiated through Libp2p, you have this nice seamless connection layer where it doesn't really matter if people are running old versions. They're obviously getting worse performance and they're missing out on bug fixes and so on. But the actual upgrading of the network is kind of a slow roll of people updating things as they need to and when they remember to.

So, it's kind of like both easy and hard to manage updates and network like IPFS because, on one hand, you can just ship new updates whenever you feel like it. People might not update to it. But you can do it. And it doesn't actually cause problems. The downside of that is if there's bugs or if there's like any critical network problems, getting the whole network to upgrade is extremely difficult. Because most people run their software once and forget about it. And convincing them to update is nearly impossible.

I think last time I looked we still had people running like a five-year-old version of IPFS. And I don't know what to do about that. But they're there. They're running it. And they seem to be enjoying it for some reason.

Then on the Filecoin side of things, you do need everybody to be running the same version of the software. Otherwise, you can't – With the blockchain protocols, you have to run the same version of the code in order to come to consensus on the state of the chain. And this makes things very tricky. Because you can't just ship updates quickly because you have people all over the world with this software running and with lots of money depending on the software.

So, when we're deploying updates for Filecoin, it's usually communicated months in advance, a specific date and specific time that everybody has to be updated by. If you don't update, you will actually risk losing money, especially as a miner. And so, these updates are planned months in advance. They have release candidates that are you know ready to go significantly ahead of time. And there's tons of communication with community, broadcasting on Twitter and through all the communication channels we have to alert people that, "Hey, this upgrade is coming. You need to update." And if they don't, they can be left behind.

But for the most part, you have this nice feature where since there is money on the line for a lot of the participants, people pay attention enough that they see this and they update in time. And it mostly goes well. You just have to be careful to not do it too often. Otherwise, it's pretty annoying to people to have to update like once a month, because there's new improvements or whatever. Most people don't actually care about improvements. They want the software to just do the thing and work.

**[00:20:05] JM:** S, in that discussion you had some mention of the relationship between miners and the actual protocol developers. Because the miners have to make a decision to update their nodes with the most recent version of the protocol. And it makes me think a little bit more about just the economics of being a miner.

And I'd love to know, if I'm a miner and I'm choosing between mining file coin and mining some other currency, presumably, you have to make it – Somehow you have to make it competitive with the other currencies that they could be spending their bandwidth mining. What happens when it becomes non-profitable or less profitable to mine Filecoin than some other system? Do you have some self-corrective mechanism?

**[00:21:01] JJ:** Yeah, it is interesting, because mining Filecoin is a very different task than most other cryptocurrencies. It feels most similar to a proof of stake. But even then, it's quite different. And the main reason for that is miners and Filecoin are storing users' data. And they're doing so on like very long-term contracts, a year, year plus.

Like, right now, we have the maximum length of an individual contract fixed to a maximum of a year and a half. But even then, if a miner decides they want to stop mining, if they don't want to be heavily penalized, they have to wait until all of their contracts expire. And this means that if there's a temporary reduction in profitability, they have to wait it out.

And this does change the calculus for a lot of people of how they think about mining Filecoin. Where with mining other currencies, like, Bitcoin or Ethereum, on a moment-to-moment basis, you can decide to mine or not based on the profitability at that moment. When you're thinking about Filecoin, you're really thinking about running a data storage business. Like, running the entirety of the business. And you're thinking about things on year time scales instead of day time scales. And so when you're looking at, "Should I mine Filecoin?" You're really thinking about the broader picture and deciding whether or not this longer-term venture makes sense for you.

And from what I've seen, the numbers for like profitability of Filecoin are actually quite good, which helps to balance out the additional commitment and additional cost that such an operation takes.

**[00:22:39] JM:** Are there any particular node specifications that make for a good Filecoin miner?

**[00:22:50] JJ:** Yeah. There's a lot going on with the hardware requirements for mining Filecoin. The proofs in particular require you have GPUs, which isn't abnormal for a crypto miner. But you also need lots of storage. You need lots of RAM and very fast single core processors. Because there's a particular algorithm that's used to encode data to make it provable, this proof of replication. And this relies on what is effectively a VDF, a verifiable delay function, which is a mostly single-threaded process that encodes the data. You can't speed it up by parallelizing it. The only way you can speed it up is by going fast.

And this does significantly impact the actual configuration of the hardware. Because now you're not looking at a motherboard with a bunch of GPUs strung all across it with a bunch of fans trying to cool it down. You're looking at a very balanced setup of infrastructure where you're balancing a whole bunch of hard drives. You're balancing your flash storage. You're balancing the processors and the cooling of that. And then you also have the GPUs for the portions of the processes that that has. So, running a Filecoin mining operation really looks a lot more like standard data center operations than what you might be used to seeing with the whole crypto mining shelves of GPUs idea.

**[00:24:15] JM:** I'd like to zoom out and talk about the broader crypto ecosystem. We did a show recently on Arweave. And Arweave is one of the more promising storage protocols I've seen since Filecoin came out. There are other storage options as well. I think Ethereum has like a native – I don't remember what it is. But Ethereum has one.

**[00:24:38] JJ:** They had Swarm.

**[00:24:40] JM:** Swarm. Did they deprecate it?

**[00:24:42] JJ:** I think it's been deprecated. I haven't seen anything in years.

**[00:24:44] JM:** Okay. Anyway. So, yeah, there's a variety of storage things. What's the spectrum of adoption like and use cases? And is it a competitive market? Is it a cooperative market? Can you use Arweave with IPFS somehow? Talk through the spectrum of storage options.

**[00:25:01] JJ:** Yeah. I don't follow a lot of the other decentralized storage protocols as much. They're not that big. Like last time I looked at Arweave, it had 20 terabytes of total usage, which is about what Filecoin onboards every 30 seconds, which, I don't know. To me, that's just not super interesting.

I really tend to pay attention to other storage options like S3 or Cloudflare's – What do they call it? D2 or something? Where they have an S3-like storage offering that is significantly cheaper on the bandwidth. The space for the centralized storage providers, it's not super flourishing, I would say. There's really not a ton there, especially when you look at like how much data is actually being handled.

You have other – If you're just looking at blanket numbers, you see other things like Chia. Chia has a ton of storage attached to their network. But it's not actually used for storing anybody's files. So, it's like a storage network in that it uses hard drives for its mining process. But it doesn't actually provide any data storage for end users.

And aside from that, you have Arweave, which has a cool mechanism of you can send your data out in a transaction and put it literally on the blockchain. But it's kind of shown that it doesn't scale. It's very small in terms of actual raw storage capacity. And it serves some of these use cases of, "If you want to put your JPEG on the blockchain, it seems like a reasonable way to do that." It's pretty expensive. But if you're paying a million dollars for a JPEG, then you might as well pay a lot of money for that. But it doesn't actually hit scale.

And I have seen people using IPFS and Arweave together where they address their data with IPFS and then store it on Arweave with the hashing specified inside of the transaction somehow. And then there was an Arweave node that made data available over IPFS out of Arweave. And that's interesting, because it shows you that when you store your data with IPFS, you're really just showing a way of addressing it and not specifying how it's persisted.

And the cool thing that means is you can persist your data however you like. As long as it's addressed by its CID through IPFS, you'll be able to use IPFS to fetch it regardless of what blockchain you end up putting it on. Or however else you end up storing it. I've seen people use S3 to store IPFS things, which is kind of weird. But it also makes sense where your IPFS is just shifting how the addressing happens. And then the data can go wherever it wants. But that's kind of the breadth of that as I see it.

**[00:27:50] JM:** What do you think is going to be the delineation between who uses centralized versus decentralized storage providers in the limit? I mean, for many applications, like, censorship is not really an issue. It's like why would I even want to use decentralized storage? But obviously not universally the case. Yeah. So, what's the delineation?

**[00:28:15] JJ:** Yeah. I mean, I think the biggest thing is, decentralized or not, people just care about what am I getting and for how much money? And if decentralized storage networks can provide a better experience, whether in terms of price, or performance, or what have you, then people will use it. And I think they don't even need to know or care that it's decentralized.

I do believe that, with decentralized protocols, we can provide a strictly better experience than the centralized cloud providers can, especially on terms of when you look at costs of using S3 or most centralized cloud providers for storage, the storage itself, it is expensive when you look at like the cost of the underlying hard drives and how much it would actually cost to store that. But the real expense of storing on cloud architectures like this is the bandwidth.

When you store your data on S3, you pay a little bit of money to store it. But then you pay a lot of money to actually get the data out. And egress costs on Amazon are like very, very high. This is how Cloudflare's been able to move in with their storage option and use their CDN layer to actually cut costs way below what Amazon has been charging, which is super cool.

I think that decentralized storage networks can actually take that even further, where since we're addressing data – We're using content addressing on the data. It doesn't matter where the content is served from. You can always verify that it's the data that you expect it to be, which means any edge node can cache it and serve it successfully. So, the opportunity here for you saving on edge bandwidth and like just saving on the egress cost in general is pretty massive. I

think the short answer to question is people don't really care that it's decentralized. They just want a better product for less money. And I do think we can provide that.

**[00:30:06] JM:** Do you have similar feelings between decentralized computation as you do about decentralized storage usage?

**[00:30:14] JJ:** Yeah. I think nobody has really done decentralized computation at scale. I think things like Ethereum, I would really not describe them as decentralized computation. Like you are doing computation in a decentralized network. But the key interesting thing there is that you're actually getting consensus on the result of running something. The main thing like Ethereum provides is trustless computation, which is a pretty different thing than like generalized decentralized computation.

I think there's been several groups that have tried to build actual decentralized computation. I think there was one called Truebit for a while, which was doing actual large amounts of computation and then certifying it on-chain and doing all the right things to make sure that the computation was correct. I don't know what actually ended up happening to them.

But being able to do like large data center scale operations in a decentralized network is what I would actually call decentralized computation. And that, we don't really have right now. And I think there's definitely room for that to exist. With things like zk-SNARKs and a lot of these things. Like, optimistic roll ups are a very interesting way to do more higher scale computation in a decentralized manner. But yeah, I do think it kind of – Once we even get there, it will come back to, "Well, is the cost benefit worth it? Is the price per performance there for people to, at scale, adopt this? And that's a whole part of the world that I don't actually know as much – I don't have the same level of understanding of how it's going to work and how we can hit the same cost benefit tradeoff for that.

**[00:31:58] JM:** Are there – I mean, you mentioned you don't take that much away from – You don't follow other protocols as closely. But are there ways that you look at other protocols for, for example, scalability concerns? Or, I mean, I guess IPFS doesn't seem to have the same scalability concerns as Ethereum does.

**[00:32:17] JJ:** I mean, yeah, we definitely look at and talk to and work with people in the Ethereum community and other communities quite a bit for the general scalability of Filecoin as a blockchain. The scalability of the storage is a very different story than the scalability of the underlying transaction processing network. And yeah, we definitely are paying attention to a lot of the space for how all of that's working.

I think what Ethereum is doing with their scaling efforts and focusing on L2s, and there's a ton of different projects out there that are working on scaling, and I think they're making a lot of really good progress. And we ourselves are just kind of – We have some efforts that are looking at how to scale a Filecoin blockchain. But I think, yeah, we're going to end up collaborating or working closely with other groups who are going that far already to figure that out.

I think a lot of the work around layer twos and like zK roll-ups in particular, I think that's the really fascinating direction that all this should be going. Because it really ends up with fairly unlimited scalability, especially when you get into the land of having recursive SNARks. The only question ends up being like, "How do you move between L2s?" And that's where bottlenecks happen. But in terms of like raw ability to process transactions on the network, you can get it very high with a lot of these L2-type solutions.

**[00:33:43] JM:** There are so many different currencies today. And I sometimes have a hard time parsing which ones actually have the most novel breakthroughs. Look at Avalanche or any of the other newer ones. And I guess I'd like to get a sense from your perspective, are there any breakthroughs in cryptocurrency development, like, actual engineering breakthroughs that have stood out to you as being novel or actually being key to scalability, or latency, or – Yeah, where are the most novel engineering breakthroughs that you've seen in other systems?

**[00:34:24] JJ:** Yeah, I think definitely coming back to different zero-knowledge proving systems and their ability to make very fast like zK roll-up type systems, that's in my mind one of the most interesting things there.

Also, there's been a bunch of work around very interesting ways of batching transactions. I remember a while ago I read about this network called Grin. And I didn't really get too much into it. But I did learn about how the actual scalability of it works. And they had a way of combining

transactions. I believe combining transactions before even getting into the block. That allows it to run multiple transactions in the same cost as running one. So, that by the time you actually are processing the block, you have this like bulk execution of many, many state transitions all at the same time, which I find like that sort of thing is really cool.

Using different networks that are able to use like VLS signature aggregation. So, changing the way that you sign your transactions such that the signatures can all be aggregated and verified at once also saves a lot of time and block validation. Because, really, what you're trying to do when you're scaling the throughput of a blockchain is you're trying to decrease validation time. And the distinction there is you have the time it takes to create a block and then the time it takes to validate a block. And the validation ends up being the bottleneck. Because in order for the network to progress, everybody in the network has to receive the new block and they have to agree that it's correct.

And you can do some really fun tradeoffs where you put more work on the person creating the block to save time on the person who is verifying the block. And with this, for example, I think Mina is doing a lot of cool stuff here, where the block producer is producing a zero-knowledge proof of the correctness of the state transitions. And then the verifier is really just checking, I believe, it's just a single SNARK. And so, that shifts the way that the scalability of the system can happen pretty significantly.

And that moves it into the scalability of that system is based on how many transactions can somebody produce a SNARK proof for in the allotted time that they have for making the block, which is a much larger time frame than they have for verifying the block.

Yeah, the systems that are looking into the different ways of making these tradeoffs and also looking into how to make layer twos more efficient on their networks I think are definitely a lot of the most fascinating directions and research into scalability.

**[00:37:12] JM:** So, as we begin to wind down, the basics of Filecoin have been in place for quite a few years at this point.

**[00:37:21] JJ:** Well, kind of.

**[00:37:22] JM:** What has taken the longest? And I guess what's the focus right now?

**[00:37:26] JJ:** Yeah. I mean, what has taken the longest? I think there's a lot of things that seem easier than they end up being. And I think the proof system is one that has been in progress for a very long time. And have a set of a proof system that works. But it's still optimal for a number of ways. It's more expensive to compute than it could be. It is slower to verify than it could be.

And so, we have a lot of effort looking into how do we make these proofs better, more efficient? And then there's a number of different characteristics that you can have on these proofs. And that's one of the things I find really interesting.

The other thing that I mentioned earlier is the VM work that we're doing. We're going to be shipping a full user-programmable smart contracts platform on Filecoin that will be, as I understand it, allow both WASM and EVM native code to run on it. And that's something we did intend to ship with the original launch of the network. But the complexity of building and deploying that was a lot more than we wanted to try and fit into the launch. And so, that's been taking – Basically, since launch, we've been focused on working on that and researching it. And I think that's slated to launch sometime this year, at least early versions of it. So, really excited about that.

And there's a number of other things that have been in progress, like, scalability of the chain and different ways of potentially doing consensus. But it's an evolving, growing system, which is kind of a lot of fun to work on.

**[00:39:00] JM:** What is the distinction between the research side of the company and the core company itself?

**[00:39:05] JJ:** I don't think there's a strict distinction. We have a fairly large research team. And we do a lot of exploration in the space through different things that aren't strictly the core – What we're focusing on Filecoin right now. There's a lot of exploratory work around consensus.

Different types of consensus. Different types of proving systems. There's a lot of research done on different zK-SNARKs systems.

Notably, we've been working on a Turing Complete zero-knowledge proof language called Lurk that will allow you to just write code and run it and then. At the end of running it, you get a zK-SNARK proof that it was run correctly, which is a really fascinating concept that you could plug into a smart contract system. A lot of our research is kind of things that will be useful or could be useful to the system in general. And we try to be pretty open about what the space of things that we look into.

And that generally feeds back into we research a thing, we find that it's really good and compelling, and we should start adopting it. Then the research team will start working closely with the engineering team. And then they'll start to form more of a concrete plan and specification for what should be built. And then that group ends up taking that the engineers who are on the ground and maintaining a software and helping them integrate that into the system and get it ready for deployment.

And so, we have a pretty good research to development pipeline that we've cultivated over the years.

**[00:40:40] JM:** Cool. Well, it's been great talking to you. And thanks for giving us an update on what's going on at Protocol Labs.

**[00:40:46] JJ:** Yeah, thank you.

[END]