EPISODE 549

[INTRODUCTION]

[0:00:00.3] JM: Ethereum is a system for running decentralized smart contracts. In the current implementation of Ethereum, every smart contract gets deployed to every full node. Whenever a user wants to call a smart contract, that smart contract gets executed on each of those full modes across the entire network. The current model for smart contract execution could be made more scalable.

In today's episode, Christian Reitwiessner joins the show to describe Plasma; a system for scaling smart contracts. Christian is a developer who has worked extensively on Solidity; the most popular smart contract programming language in Ethereum.

For the last month we focused en blockchain related topics and we will soon be shifting to other topics. Some of the listeners have not enjoyed the blockchain focus. Other people have loved it. So for everyone listening, we would love if you filled out the listener survey at softwareengineeringdaily.com/survey. Of course, you can send me an email at any time, jeff@softwaredaily.com and let me know what you think. I'm happy to afield individual emails. You can also join our Slack channel if you want to hang out with our community at softwareengineering.com/slack. If you're completely sick of cryptocurrencies, you can check out our back catalog of episodes at softwaredaily.com or by downloading our apps, which have all of our episodes, including a greatest hits list, which is a curated set of most popular shows. The apps will soon have off-line downloads and bookmarking. I know a lot of people have been asking for those things, and we've actually got them in testing right now. They will be out very, very soon and your software daily app will be more like an actual podcast player.

So with that, let's get to the episode.

[SPONSOR MESSAGE]

[0:01:57.1] JM: Software Engineering Daily is brought to you by Consensys. Do you think blockchain technology is only used for cryptocurrency? Think again. Consensys develops tools

and infrastructure to enable a decentralized future built on Ethereum; the most advanced blockchain development platform.

Consensys has hundreds of Web3 developers that are building decentralized applications focusing on world changing ideas, like creating a system for self-sovereign identity, managing supply chains, developing a more efficient electricity provider, and much more. So, listeners, why continue to build the internet of today when you can build the internet of the future on the blockchain?

Consensys is actively hiring talented software developers to help build the decentralized web. Learn more about Consensys projects and open-source jobs at consensys.net/sedaily. That's C-O-N-S-E-N-S-Y-S.net/sedaily. Consensys.net/sedaily.

Thanks again, Consensys.

[INTERVIEW]

[0:03:12.1] JM: Christian Reitwiessner, welcome to Software Engineering Daily. It's great to have you.

[0:03:16.6] CR: Yeah, thanks for having me.

[0:03:18.5] JM: So today we're going to talk about smart contracts and scaling those smart contracts. I think that most of the listeners know what a smart contract is. It's a program that executes across multiple computers so that the computers can have a consensus on the order of computation, and it's mostly widely used in terms of Ethereum, Ethereum smart contracts. So today smart contracts are most widely used for token sales and multi-sig transactions, and perhaps CryptoKitties. In order to get more widespread usage, more widespread applications and higher throughput on smart contract networks, we need to be able to scale smart contracts. What are the different ways that we need to scale smart contract execution?

[0:04:13.4] CR: This question I always give the — So there are basically three dimensions where smart contracts or block chains in general can scale. One dimension is transaction

throughput and one solution to that is take the transaction off chain or basically create state channels or payment channels for the transactions. Then the second dimension is scaling the computation itself. So if you have a single transaction that requires a lot of processing power to be processed, then this is scaling computations, and solutions to that are, yeah, SNARKs, STARKs and interactive solutions like TrueBit. What's the third? Yeah, scaling state size. So if you have gigantic smart contracts that were single executions of the smart contract don't do much, but they have to access a gigantic state. So think about a token for example. A very popular token where everyone has some of these, but where a single token transfer is quite easy to do, but you have a gigantic state you have to modify. The solution to that is seems to be sharding or Plasma. I mean, Plasma goes a bit — It doesn't fit well into this dimension theme because it scales both state and transaction throughput.

[0:05:43.6] JM: Okay. I think the three ways that you're were talking about that smart contracts have difficulty scaling on. One; it's transactions. The same kind of throughput problems that Bitcoin has for financial transactions, but in the context of Ethereum we're talking about computational transactions. The other two areas, one is scaling the amount of computation that can take place in a transaction. This would be something like if you wanted to execute a huge map reduce function across the blockchain, for example, maybe if I understand correctly.

[0:06:24.1] CR: Yeah. I mean, things that are typically done there are typical examples for things like that is verifying one blockchain inside of another blockchain.

[0:06:32.4] JM: So this is a highly — It's a CPU-intensive, computationally-intensive process. Then the other one is the ability to process something that takes a ton of space. It takes a lot of storage or memory in order to accurately process it. So maybe you need to do a linear search through a large amount of space, and that is difficult to scale, because in order to do that you would have to have all of the nodes that are processing that transaction to also iterate through that entire state space.

[0:07:09.2] CR: Exactly.

[0:07:10.3] JM: Okay. So now that we've talked about those three scalability issues, let's just talk about a quick example of something that encountered scalability issue on Ethereu, which is

CryptoKitties. Just to give people a small example, what were the scalability issues that CryptoKitties encountered on the Ethereum blockchain?

[0:07:29.5] CR: CryptoKitties, I would say, is a combination between the token contract and on-chain option. I think it had some computation-intensive tasks, I think, or perhaps not. I don't know. I mean, at least I think the genetic crossover. So you're able to take two CryptoKitties and let them breed and then you get a new CryptoKitty. Each of these CryptoKitties has a genetic code. Of course, the genetic code of the parents combines into the genetic code of the child. That might be computationally intensive, but I'm actually not sure about that. The other main issue is that it's just — I mean, it wasn't very popular in December 2017, and because of that it had very large estate and also many transactions going into the same smart contract.

[0:08:27.9] JM: So this would probably be problems one and three that you enumerated in the scalability challenges. There're lots of transactions. All of the full nodes on the network have to process all of those transactions. The third problem, which is that there was a large state space for CryptoKitties. So you've got two out of three out of those scalability problems that you mentioned just with CryptoKitties. So it sounds like CryptoKitties had problems one and three. You had lots of transactions and you had a large state space of those transactions. Is that accurate?

[0:09:04.1] CR: Yeah, exactly.

[0:09:05.2] JM: Okay. So we've got some solutions on the table for improving some aspects of scalability. My understanding is that we've got proof of stake, which will help with the transaction throughput. We've got plasma, which could help with — Well, no. Why don't you just go ahead and explain to me the solutions that are on the table to improving scalability issues?

[0:09:32.4] CR: So one quite easy solution, which is not too nice. One quite easy solution for this specific case would be to just take this single smart contract into its own blockchain that is still connected to the main Ethereum chain. That would make it easier to some degree, but we of course — So this single [inaudible 0:09:55.0] blockchain will be managed by a few number of special nodes, which of course is a centralization risk. Also, it's much harder for this specific smart contract to interact with other smart contracts on the Ethereum chain.

So one of the solution is sharding. This is often combined with proof of stake. So often when people say proof of stake, this sometimes means sharding. So you said proof of stake would improve the transaction throughput. I'm not too sure about that. Yeah, it increases transaction throughput a little because you don't have to do mining. But you won't get a big improvement. The big improvement you only get when you also add sharding, Sharding means that — The main problem with scalability in blockchains is that every single full node has to process and verify every single transaction. With sharding, transactions or rather state space or accounts are split into a number of different areas and each validator. So that's the name of the miners in proof of stake. It's only processing one of these shards, one of these areas. To keep the same security level, you rotate the assignment between validator and shard regularly.

[0:11:20.1] JM: Right. So what you said about proof of stake, not necessarily helping with scalability of transaction volume. What you're saying there is the first phase of rolling out proof of steak is just adding these validators on top of the proof of work mechanism. It's only later on when we get to full proof of stake that we can actually implement sharding and we can have transactions being processed in parallel.

[0:11:50.0] CR: Yes. So as far as I know, the current plan for Ethereum is to not take the Ethereum main chain and split into chards, but instead takes shards and add them to the Ethereum chain. So the problem with that is — I mean, of course, sharding is not fully specified yet and it depends on how sharding is done. But I am not sure whether, in the case of CryptoKitties, sharding would actually be helpful, because usually — So the easy way to do sharding is to just split the address space. So — I don't know. If your address starts with a one, it's in shard one. If it starts with a two, it's in shard two, and so on. That would cause the CryptoKitties smart contract to be in one specific shard and you still have — On that specific shard, you still have the problem with high transaction volume. So it does not really help.

[0:12:47.6] JM: Yes. Okay. So that could bring us to Plasma, because Plasma — If I understand Plasma correctly, it moves every smart contract into its own side chain. Is that it good way of presenting it?

[0:13:03.3] CR: At first, Plasma is more an idea than a specific system. So it's a way to scale transactions on a blockchain or smart contracts on a blockchain, but it's implemented by multiple different teams and they might also do it in a different way.

The main idea there is that you don't actually modify your blockchain in any way, but instead you add blockchains to it. Earlier I said that moving CryptoKitties into its own blockchain would be a way to scale it. This is more or less what is done with Plasma. You create a new blockchain and you connect it to the main blockchain by means of posting the current block hash of the new blockchain into the old blockchain. The new blockchain is called a child chain and the old block chain is called a parent chain, because they have this parent-child relationship.

So this is an idea that is already quite old. So you have a special purpose blockchain, which is perhaps controlled by a consortium and you connect it to the main chain by posting these hashes, and this ensures that you cannot just arbitrarily modify the child chain, because it's always bound to these hashes that were posted to the main chain.

[SPONSOR MESSAGE]

[0:14:44.8] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use, and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plug-ins. Use the value stream map to visualize your end-to-end workflow, and if you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on-the-fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations. You can check it out for yourself at gocd.org/sedaily.

Thank you so much to ThoughtWorks for being a longtime sponsor of Software Engineering Daily. We are proud to have ThoughtWorks and GoCD as sponsors of the show.

[INTERVIEW CONTINUED]

[0:16:06.6] JM: So let's take a step back. So let's say we have a smart contract on the Ethereum blockchain today, and if you contrast that with how that smart contract would operate in a world where Plasma was implemented and deployed, what would be the difference in how that smart contract would function?

[0:16:28.1] CR: I didn't want to get so early, because the problem is you probably have to heavily modify the smart contracts for it to still work in a Plasma setting. The reason — You can draw a kind of analogy to big data computations nowadays. If you run very complex computations on gigantic amount of data, if you take a certain computation, run it locally on a small amount of data, it probably works reasonably well.

Then if you want to run it on a gigantic amount of data, you rent a compute center somewhere which has a MapReduce way of dealing with data. People say, “Oh, yeah. Use that. It's very fast.” Then it will probably not work as you thought. You won't get that speed up that other people that say is possible. The reason is because you have to modify the algorithm. You have to modify the way you process the data so that it’s compatible with this MapRroduce topology of the compute center. A similar thing has to be done for smart contracts to scale on Plasma.

[0:17:39.5] JM: I know that Plasma is co-authored by Joseph Poon, who wrote the Bitcoin Lightning Network paper. What is a he lightning network and how does that relate to the ideas explored in Plasma?

[0:17:54.2] CR: So a lightning network is a way to move transactions from a blockchain, so to move them off a blockchain. The main idea is that if you have two parties that send money back and forth between each other, then if I send you $10 and you send me $8, then I could've just sent you $2, right? It’s much easier and takes just one transaction instead of two. If you send even more money back and forth, then we can save even more transactions.

Lightning or the more generalized thing, state channels, do that by the basically — So we send each other intentions of sending money and always keep a kind of balance checkbook and we write checks to each other, “Okay. I now owe you that amount of money. This is the sequence number four, and then sequence number five if the check is over. I know you this amount of money,” and so on. At some point if we say, “Okay. That's enough.” While this this channel is running, you don't actually have access to the money. You just have this promise from the other party that says, “Oh, yeah. Eventually I will give you this and that amount of money.”

If you want to actually get access to the money, you have to close the channel. We send these checks back and forth between each other, but we never — They never touch the blockchain. So we just did via normal connection over the internet. When you want to close the channel, you could take one of these checks and go to the smart contract, and then the smart contract gives you the money and closes the channel.

[0:19:37.2] JM: Yup. Is there a relation to be drawn between that idea of the lightning network, the off-chain computation, the occasionally touching the Bitcoin block main chain in order to reconcile the transactions that occurred off-chain with the main chain? Is there an analogy to be drawn between that idea and the ideas of Plasma?

[0:20:03.2] CR: It's not really the same, but they are similar. So the similarity is that in Plasma you have this — Okay. Actually I didn't paint the full picture earliest. So I said you have one blockchain that is connected to the main chain, and this blockchain kind of represents this payment channel. As long as you stay inside this external, in this child chain, then you have state updates, but the state updates stay in the child chain. Only when you want to use your money in the main chain or use your state in the main chain, then you have to use a special transaction that moves data from the child chain to the main chain. So in that way they are quite similar.

The main difference is that with state channels, you only have two participants at least at a single hop. There are ways to extend that to multiple hops, but one elementary channel is always between two people. With Plasma, you have these child chains which are actual fully-fledged blockchains which can have an unlimited number of participants.

[0:21:11.9] JM: Okay. I see. That's interesting. So when you have these child blockchains, you have entire blockchains that are transacting off of the main Ethereum blockchain. Are these child blockchains, are they getting validated or proved by a high volume of nodes just like the transactions that occur on the main Ethereum blockchain?

[0:21:38.4] CR: So the child chains are fully-fledged blockchains, but they probably won't use proof of work. They might choose proof of take or proof of authority. But the way they scale is by just repeating the same process over and over again. So you can create a child chain and this child chain is also — It is also controlled by the main chain. The reason is — So when you take a look at state channels and you close a channel, then everything that happened inside the channel is verified by the closing transaction. This is of course — A state channel is a very simple — Or at least a payment channel is a very simple thing to validate. You just have to check whether the balances don't exceed the initially deposited amount.

With Plasma, the ideas is that the child chains are fully-fledged smart contract block-chains, but that's not a problem. So you can still verify the execution of a smart contract inside the child chains — Sorry. You can still verify the execution of a child chain transaction inside the parent chain. You can do that for some. Of course, if you do that for every single execution, then it doesn't scale anymore. The idea is that if there's something wrong in child chain, you go to the parent chain and complain. Then the parent chain either — Yeah, again, there are different models. Then the parent chain either checks the transaction or it just directly allows you to move all of your data from the child chain to the parent chain.

Now since the child chains are also fully-fledged smart contract blockchains, they themselves can act as a parent chain and you can basically create a whole sequence of children then that themselves — Have children themselves and so on. Then, of course, a single blockchain can have multiple children, and by that to get a kind of a tree structure. Once you have this tree structure, the hope is that it will scale very — So it will scale more or less infinitely, because there are so many leaf nodes where you can go to, and it is probably also the case that the closer you are to the root of the tree, the more expensive the transactions will be. So you always have this — Okay. The more expensive transactions will be, but it will also be better secured, because it's closer to the main chain. It's easier to do this complaint. There is a trade-off

between security and transaction costs and everyone will be able to find a node inside the tree where the trade-off is best for them.

[0:24:23.3] JM: Could you give a hypothetical example of a kind of smart contract that I would want to create where I would want to have a child blockchain associated with that and then maybe I would want to have additional child blockchains off of that blockchain. When you're talking about this tree of different blockchains, I'd love to have an example to think about for why I would be creating these other child blockchains.

[0:24:56.0] CR: So an example is a simple token contract, and this is also the main example that currently people are building Plasma systems for. The ideas that — So you start in the main Ethereum chain and the transactions are too expensive for you, so you move into the Plasma chain. To do that, you send a special transaction that locks your token inside the main chain and creates a new account in the child chain that has the same amount of tokens that you previously had on the main chain. Then you can either directly use stuck child chain and send your tokens somewhere or you move further down into the tree with essentially the same mechanism.

You can also move up again by posting a special transaction and that will destroy your token in the child chain. Once you have a confirmation of this, of the tokens being destroyed, you can go to the parent chain and claim your tokens there.

[0:25:59.9] JM: If I am the creator of smart contract like that, let's say I spin up a token, do I get to choose the structure of the child blockchains or is that architecture decided by all the constituents that are involved to all the different token holders?

[0:26:19.6] CR: So in the end, you don't even need to be the creator of the token, and it all depends on who will use it. So everything is extremely flexible. You can create your own Plasma tree. You can reuse an existing Plasma tree. In the token example, since everything that all the tokens that go down the tree are locked in the parent and the transaction that moves tokens upwards can only free tokens that were previously locked. You can't create tokens out of thin air, and because of that, it is reasonably secure and you can do it with every token that exists.

[0:27:01.9] JM: And to revisit one of the points that you made where you can have these trees of blockchains where may be different blockchains in the tree have different approving structures. Maybe some of them are proof of work. Maybe some of them are proof of stake. Maybe some of them are proof of authority. Why would we use different consensus mechanisms under different circumstances of that blockchain tree?

[0:27:28.6] CR: So inside the same tree, I probably makes sense to use the same consensus algorithm everywhere and it also makes sense that if a contract is deployed so that contracts are always equal to deployed on all nodes of the three. So you have a copy — So you have this main parent node that sits directly underneath the Ethereum blockchain and everything that is present there, so all the consensus algorithm and all the smart contracts is also present in the children except for the state. The state is always summarized in a hash and is not repeated than children and the other way around so that the state of a child is summarized in a hash and only posted to the parent in form of this hash.

[0:28:16.7] JM: Okay. How often do the child blockchains need to reconcile with the main chain or how often are they touching base with the main chain?

[0:28:27.0] CR: That is something that needs to be calibrated. So probably — I don't know. Every 10th block or — I mean, this also depends a little about on how you want it to work, because if you want to move money from a child to a parent, you have to wait for such an update. If it only happens every 10 blocks, then it takes longer for data to move between the chain. If you do it every single block, it's much faster, but then we also have a higher overhead by these update transactions from the children.

[0:29:00.8] JM: One mention in that is in the Plasma whitepaper is to construct economic incentives for globally persistent data services, and this is in contrast to how Ethereum is mostly used today. Like you said earlier that kind of what we have with Ethereum today is it's not like we have a world computer. It's like we have certain parts of a world computer. We've got the CPU of a world computer, but maybe we don't have the file system or the in-memory structures. We've only got a few parts of it, and that's part of why Ethereum is not used for all kinds of things.

So this idea that Plasma could construct the incentives to make globally persistent data services, this sounds like it could give us access to more parts of that world computer that we would like to have. How does Plasma create those mechanisms?

[0:30:02.6] CR: To be honest, I don't know what they mean by that statement, because it doesn't sound like participants in the Plasma system are paid a lot of money to be a validator on one of the nodes in the trees. The problem I also see there is that the more nodes of the tree you have, the more validators you have, the more expensive it will be to keep the system secure to pay the validator.

In my view, I think it is more likely to have kind of nonmonetary incentives for people to keep these nodes running. The reason is that a Plasma system is only secure for the user if they watch the nodes. I'm not sure if we went to that in detail. The reason behind is that validators can misbehave inside a Plasma system, and the best way to cope with misbehaving validator is to perform an exit, and this means to move all your data one level up in the tree or as many levels that is required to get on to safe ground again.

Because of that, whenever you have tokens or data inside a node in such a Plasma tree, you have to verify that all transactions inside that node and inside all of the parent nodes are processed correctly. Whenever you notice something is something is weird, then you perform an exit. That creates an incentive to watch and which also makes it secure, but it's not a monetary incentive.

[SPONSOR MESSAGE]

[0:31:54.1] JM: LiveRamp is one of the fastest growing companies in data connectivity in the Bay Area and they're looking for senior level talent to join their team. LiveRamp helps the world's largest brands activate their data to improve customer interactions on any channel or device. The infrastructure is at a tremendous scale. A 500 billion node identity graph generated from over a thousand data sources running a 85 petabyte Hadoop cluster and application servers that process over 20 billion HTTP requests per day.

The LiveRamp team thrives on mind-bending technical challenges. LiveRamp members value entrepreneurship, humility and constant personal growth. If this sounds like a fit for you, check out softwareengineeringdaily.com/liveramp. That's softwareengineeringdaily.com/liveramp.

Thanks to LiveRamp for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:33:02.3] JM: This brings me to a question I'm not sure you'll have the answer to, but I haven't asked anybody. I've done a bunch of shows about Ethereum recently, but I haven't asked anybody about — I think it's called Swarm, like the persistent data layer that's on Ethereum. Is that Swarm? Is that right?

[0:33:19.7] CR: Yes, a decentralized file system.

[0:33:22.0] JM: Yeah. Do you know anything about that?

[0:33:24.0] CR: Yeah.

[0:33:25.4] JM: Tell me about Swarm and what are the limitations, because that sounds like a globally persistent data service. Why is Swarm widely used?

[0:33:33.5] CR: I think it's still in testing phase. So Swarm turns who pays for what little around. The problem with blockchains is that you can store data on them and you only pay once with the transaction that creates the data, or in case on Ethereum. Then data is there for all eternity.

[0:33:52.3] JM: Sorry. The transaction that creates the smart contract or the transaction that interacts with the smart contract and makes an entry in the database that stores the smart contract.

[0:34:02.4] CR: Yeah, exactly. For Swarm, the idea is that you pay for your data being stored for certain amount of time, and this is done — Why a set of smart contract? So the data itself is not stored on the blockchain, only hash of it is stored in the blockchain. So everyone can run a

Swarm node and reserve a certain amount of disk space for that, and then a request can come in from someone who wants to store data and you accept the request to store data and you get paid for that.

Then there is a mechanism that ensures that you actually keep the data for this certain amount of time. If you don't keep the data or if at some point you can approve that you still have the data, you have to pay a fine, and that's the incentive structure, a rough overview of the incentive structure behind Swarm.

[0:34:56.2] JM: Do you have an idea of how that compares to the IPFS file coin system?

[0:35:01.8] CR: I am not 100% up-to-date about IPFS file coin, but I would assume it is very similar. The main difference is that Swarm was initially built with that idea of paying for storage, and IPFS, they first built the storage layer and then added the incentive structure to pay for the source, I think. The teams, the projects, they also collaborated a lot. Yeah, these are very similar concepts.

[0:35:31.9] JM: Yeah. I really want to convey to people that if people are not building incredible decentralized applications yet, is that Ethereum is not very fast. It's not superefficient. We are missing certain parts of the world computer, but these are going to be incremental things that we're going to work towards, and we can get there. Like if you look at the technology that is there today and you look at all the different projects that people are working on, you can see the work getting towards the overall infrastructure of that world computer. I want to emphasize that, because I think sometimes people think that there is some sort of debate here, like there's a fundamental problem with Ethereum, as if the programming paradigm doesn't work. I don't think that that's the case. Everybody smart who I've talked to seems to think that the programming paradigm is amazing. We just have scalability issues. We've got pieces of infrastructure that we need to kind of figure out, but we're not stuck on anything, and it's just going to take time. Would you say that's an accurate perspective on the blockchain development or do you see any fundamental issues?

[0:36:45.1] CR: So the EVM is turing complete or at least pseudo-turing complete, and because of that we can do just about anything. The problem is that this anything might be a little too

costly, but since it is turing complete, we can implement any scalability solution that might be conceived — I don't know, 10 years from now. So we have these kind of moonshine map things.

[0:37:10.5] JM: What did you say? Moonshine map.

[0:37:12.6] CR: Yeah. Isn't that the term?

[0:37:14.7] JM: What does that mean?

[0:37:16.0] CR: Where things look impossible, but somehow they actually seem to work in the end. So examples of that are the scaling solutions in SNARKs, STARKs and Bulletproofs, where there is some way to take in computation and condense into a much faster thing that does the same thing in the end, and kind of sounds like that can't actually work, but it does.

There have been continuous improvements over the last years, and if you can use all of these improvements without any change to the virtual machine. Because of that, I don't think that we actually have a problem.

[0:37:52.8] JM: Yeah. So one other area that I'm not sure you are well versed in, but when I talk to people about scaling computation, computationally-intensive tasks, people often bring up Golem. Do you know much about Golem? Do you know what issue Golem is tackling and how that contrasts with solutions that are built on Ethereum?

[0:38:15.5] CR: Golem is doing — I don't know. If you know things like [inaudible 0:38:19.2] or — What is it called?

[0:38:22.0] JM: Oh, yeah.

[0:38:21.9] CR: Where you can use your computer to solve costs for scientists to do computations, and this is very comparable to Golem where the main difference Golem is that you get paid in cryptocurrencies [inaudible 0:38:37.8] that you get paid it all. I mean, for [inaudible 0:38:38.9], we don't get paid.

The difference to scaling computations on blockchains is the trust. So Golem plans to add a trust layer, but I think currently they don't have it. They do spot checks whether some computations are correct, but it's also not — So when you send a task out via Golem, then you better not — Yeah, bet your house on it. What they want to add is a — So at least in the beginning they wanted to add verification system that uses TrueBit. So an interactive mechanism to check a computation and be 100% secure about the outcome of their check in the end. Once you have that, you can also kind of scale computation infinitely.

[0:39:29.0] JM: Okay. I think that's probably a distraction. I should probably just get back to discussing Plasma a little bit more and then we can wrap up the conversation. But talking a little more about Plasma. So as you said, today this is just an idea. I don't think any steps have been taken towards implementing Plasma.

[0:39:48.9] CR: There are implementations of Plasma.

[0:39:50.3] JM: Oh, there are. Okay.

[0:39:50.9] CR: They are in testing phase, but as far as I know, they only implement the token type of Plasma and not the smart contract general computation types Plasma. [inaudible 0:39:59.4] also that this fundamental thing, the possibility of an exited case of an era is much easier in the token model than the generalized smart contract model, because it is specific to a smart contract what actually an exit means. So for a token, it's easy. So you take a token balance in the child chain and you have the same balance on the parent chain. That's an exit.

But for a smart contract, what does it mean? I mean, take out your CryptoKitties. That's what it means for CrytoKitties, but that's also the — I mean, that's the token part of CryptoKitties. What does it mean? So if you have an ongoing auction to sell a CryptoKitty in a child chain, what does it mean to exit from that? So to get full smart contracts on Plasma, somehow you need to implement the exit functionality as part of the smart contract.

[0:40:44.7] JM: Sorry. How would you define the term exit?

[0:40:48.3] CR: I mean, that's basically the question, right? For tokens it means move your balance up the tree. For smart contract, that somehow means move your state up the tree or — Yeah.

[0:41:01.2] JM: I see, the reconciliation process where you're touching base with the main chain.

[0:41:07.0] CR: No. There is a process where the validators, all the child chain regularly pose the root hash or the block hash to the main chain. That's not an exit. Yeah, you can say that this is the transport layer that allows an exit, because — So when a node, when the chain inside this tree does something it shouldn't be allowed to, from that point on you cannot trust that chain. You cannot trust that node anymore for anything.

So what you have to do this you have to do the exit starting from the parent chain, because the parent chain you can still trust. You have to post a special transaction to the parent chain that includes multi-proofs against the root hash the child chain posted earlier. That's the way you can do an exit.

[0:41:54.8] JM: Okay. Just to wrap up, one thing that I find curious is the way that crypto economic systems are implemented and how they are approved to work, because I find it funny that crypto economics is all about proving. You've got a proof of work. You've got a proof of stake. You've got a proof of authority, and yet the only way that we can actually prove that the overall systems work is through empirical data. We have to roll these things out and actually test them and see if somebody can break it.

It seems like we can't make formal proofs for these things working. Do you have a frame, a mental framework for how you think about validating, like how you think about rolling out and validating crypto economic systems?

[0:42:46.3] CR: There is a quiet old branch of — I think, at the intersection between mathematics and computer science called mechanism design. Somehow the community of mechanism designers and blockchain, they didn't really connection yet, because mechanism design, that's exactly what they're doing. So they design games and then reason about the

incentives of participants of these games, and that's exactly what crypto economics is about. I mean, it's not exactly, but it's a big part of it. So they do former proofs about the behavior of the participants in that game.

The idea is not that you have a specific game and analyze it. The idea is that you want to enforce participants to behave in a certain way and then try to find a game that has the correct incentives for them so that it's the best strategy for them to act in that way you initially wanted them to act in. That's exactly what smart contracts or in general, yeah, incentives designed on blockchains is about.

[0:43:52.9] JM: So maybe it's not about actually proving that it works in practice, but it's about doing your best to make sure that the incentives are aligned to make the system function.

[0:44:03.5] CR: Yeah. I mean, it's always — I mean, we're building actual practical system. So it's always both worlds. The theory has to be correct, and the good thing about theory is that you can have proofs. Then it also has to prove in practice to work correctly. Yeah, I think we're missing formalism in the incentive design a little. So we could really improve there, I think.

[0:44:28.4] JM: Not that we've ever had formalism in economics. Also, I don't think we've had much formalism in production computer science systems as well. I think most of formalism in computer science has been relegated to the formal verification people working in distributed systems, or maybe programming language people, and it never really finds its way into production systems.

[0:44:52.5] CR: Yeah, it depends how far you go, right? So you can prove an algorithm correct without actually automating this proof. Yeah, I mean you cannot — For previous systems, it was mostly sufficient that things work in practice, because they weren't as open to adversaries as smart contracts or blockchains are now.

[0:45:15.8] JM: That's a good point. That's a very good point. Okay. Well, Christian, it's been great talking to you. I feel like I learned a lot in this conversation, so I appreciate you coming on Software Engineering Daily.

[0:45:25.4] CR: Yeah, thanks for having me. I think we covered a lot of ground here. Thanks for the nice conversation.

[END OF INTERVIEW]

[0:45:33.5] JM: Users have come to expect real-time. They crave alerts that their payment is received. They crave little cars zooming around on the map. They crave locking their doors at home when they're not at home. There's no need to reinvent the wheel when it comes to making your app real-time. PubNub makes it simple, enabling you to build immersive and interactive experiences on the web, on mobile phones, embedded in the hardware and any other device connected to the internet.

With powerful APIs and a robust global infrastructure, you can stream geo-location data, you can send chat messages, you can turn on sprinklers, or you can rock your baby's crib when they start crying. PubNnub literally powers IoT cribs.

70 SDKs for web, mobile, IoT, and more means that you can start streaming data in real-time without a ton of compatibility headaches, and no need to build your own SDKs from scratch. Lastly, PubNub includes a ton of other real-time features beyond real-time messaging, like presence for online or offline detection, and Access manager to thwart trolls and hackers.

Go to pubnub.com/sedaily to get started. They offer a generous sandbox tier that’s free forever until your app takes off, that is. Pubnub.com/sedaily. That's pubnub.com/sedaily. Thank you, PubNub for being a sponsor of Software Engineering Daily.

[END]