EPISODE 1204

[INTRODUCTION]

[00:00:00] JMayerson: Blockchain technology has a wide variety of potential applications. Fields such as finance, supply chain management and maybe even voting have seen innovations driven by the development of distributed applications built on blockchains called DApps. However, developing a DAp on a blockchain often requires low level knowledge about cryptographic protocols or particular networks. Since no one blockchain platform has emerged as dominant and the field itself is rapidly evolving, there's a high opportunity cost for developers if they choose to invest significant time learning one blockchain paradigm or another.

Reach provides a platform for developing DApps complete with a high-level language based on JavaScript. Reach allows developers to write one set of code to specify the DAp and all of its components and which can be deployed onto any blockchain implementation under the hood. Reach's goal is to allow developers to focus on writing business logic for their DApps rather than worrying about low-level implementation details and aims to smooth the steep learning curve for developers new to the world of blockchain.

Chris Swenor and Jay McCarthy are the founders of Reach. Chris was formerly the co-founder and CEO of Alacris Protocol, an operating system for blockchain applications, and he's currently a technologist in residence and mentor at Harvard. Jay has been a computer science professor for over a decade and worked on the development of the Racket Programming Language. Chris and Jay join the show today to talk about the challenges of developing on blockchain, how Reach helps make blockchain developers more productive, and how the blockchain ecosystem might evolve in the future.

[INTERVIEW]

[00:01:35] JMeyerson: Guys, welcome to the show.

[00:01:38] CS: Thanks for having us.

[00:01:40] JMeyerson: It's early 2021. What are the applications of blockchains today?

[00:01:47] CS: That's actually a great question. And the applications of blockchains today, if you look at purely the number of users actually using it, are currency and DeFi are really where most of the blockchains are being used. I mean there're a lot of large companies looking at supply chain and insurance use. But if you're going to talk about actually real-world use cases that users are actually using it, it's just currency and DeFi.

[00:02:18] JMeyerson: And for building applications in those spaces, what do you need to know or what kinds of – Can you go a little bit deeper? Get a little more fidelity on what those applications look like?

[00:02:31] CS: Sure. So when I say DeFi, that stands for decentralized finance. And the type of applications that are out there are lending platforms, decentralized exchanges and the ability to exchange one token to another without having to actually go through a central entity. The other tool that are built on top of these platforms to be able to automate the generating yield stable coins, meaning paying a token to the U.S. dollar in a decentralized way. Things like that are in DeFi.

[00:03:07] JMcCarthy: And let me talk a little bit about what they look like in practice from a programmer's perspective. So when you build a decentralized application today, typically what you'll do is you'll really write like three different programs. You'll write a smart contract program. Those are the ones that probably get the most attention. These are written in languages like Solidity, for example. And that's the program that actually runs on the blockchain under a virtual machine like the Ethereum virtual machine. Ethereum, by the way, is like the market leader in this space. So it's really easy to explain things in terms of how it works.

And in addition to that smart contract program, what you'll do is you'll write like a middleware layer that actually interacts with the blockchain protocol, because basically every time you make a new smart contract, it's like you're making a new network protocol and you don't want that network protocol to be like directly spoken by, let's say, your frontend user interface. So you'll typically write like a middleware layer that will communicate with your specific new protocol and then you'll write a frontend application on the web or natively or something like that. And so these three programs all working together, that's what we call a decentralized application.

And then in terms of like from a program perspective, what does a solidity program look and feel like? Basically what you're doing is you can kind of think of it as you're making like a single object and you have to think about what is the state space of that object and what transitions do you want to allow that object to go through in changing its state space? And basically, for every single operation that you want your custom protocol to have, you're going to meticulously write down which other states it's allowed to be in to make this transition and what modification is going to happen next? And the role of the blockchain in this program is to essentially guarantee that everyone across the world agrees on the state changes that this particular object goes through and that's what constitutes the decentralized application.

[00:05:12] JMeyerson: What part of that application stack is the hardest to build?

[00:05:17] JMcCarthy: I think that the frontend is very straightforward. There are a few things that are a little bit strange about it because it's so asynchronous, but that's pretty basic for full stack developers. Similarly, the middleware, the main thing that's complicated with it is that it's really a totally separate program that you have to synchronize with the smart contract program. So it's almost like if you imagine kind of the old days of web programming where you would change your form labels and you had to make sure that you updated the form consumer on the server, it's kind of that sort of thing where you have these two programs that are evolving in tandem. Or another way to think about it is that you're writing the RPC and you're writing the RPC client at the same time. So that's a little bit complicated. And there's some strangeness to the way that you have to interact with these networks.

But really the most complicated thing is that smart contract. And the main reason it's complicated is because thinking about taking a high concept program like an automatic market maker and turning that into a single object with this bespoke protocol of what transitions do you want to allow and how do you verify the state and how do you check it? That's really complicated. In particular, it's complicated because it's so scary, because if you do it wrong, then you – Like what is does doing it wrong mean? Doing it wrong means that now your decentralized application is going to say yes to some transactions that you meant to say no to. Meaning that you're going to give away money that you didn't intend to give away, or you're going to say no to transactions that you meant to say yes to, which means that you're going to lock away money forever that no one will be able to get access to.

So both of those kinds of mistakes are really common in the decentralized application world and they're really the source of these headlines that you read about, “So and so lost 20 million dollars,” or “So and so locked away 10 million dollars from their contract.” They come from errors like that. So it's very tense to build these programs. And in particular, all of these programs, something that's kind of happening in the background that I didn't mention before is that when you run a program on Ethereum, you basically pay per cycle. It was a way to think about it. So that when you write a function, if this function takes 100 cycles, it's going to literally cost more to run that than another program that takes 102 cycles to run.

And so because of that, there's a whole lot of stress and emphasis in most decentralized application, in most smart control programming languages, to give you the programmer detailed intricate control over exactly how the program gets compiled. So much so that in the most popular one of these languages, which is Solidity, it's almost like programming a GPU and that there's this very concrete difference between different layers of memory and you have to explicitly move things from one layer to another and even to the point where the Solidity compiler doesn't really abstract away details like variables from you. It's almost as if you have to know that there are exactly 16 registers and make sure that you don't go beyond using those 16 registers, because then the compiler will fail. And it's failing from its perspective for your benefit because it would cost more if you moved away from using those registers.

So if you're in the audience interested in blockchain programming, in some ways I’m trying to intimidate you by thinking about all of these weird details, but it's not a false intimidation. These are the kinds of issues that many blockchain programmers think about a lot and a lot of stress is put on these issues by the current way people talk about designing blockchain programs. And of course this is all preamble, of course, to what Chris and I are working on, which is a language and development platform that tries to get rid of all of these problems. But maybe you have another question before we get to that, Jeff.

[00:09:12] JMeyerson: I mean, you've given a good overview for why the language is actually difficult to program in. What about the current set of tooling and deployment mechanisms? Can you can you walk me through the actual experience of a developer who is building on Ethereum?

[00:09:30] JMcCarthy: I think this is actually one of the things that most existing Ethereum developers really have a lot of great tools. There're testing development servers that you can launch where they're decoupled from the real blockchain and you can test your program entirely locally. There are entire test networks that you can test deploy to that are shared among many people but they basically have like fake money on them and you can just go to a website and they'll transfer you ten thousand dollars of fake money. And there are also simulation environments. The most popular one of these is called Truffle. Where you can basically from inside of VS Code launch one of these development blockchains and see exactly what is happening behind the scene.

So for most, from like a deployment perspective of like the test experience, it's actually pretty nice on Ethereum. The main thing is that there're all these interlocking pieces that you really have to know about. And something again that's kind of in the background of this conversation is that even though we're talking about Ethereum because it's really the market leader, there's all sorts of other alternative consensus networks other than Ethereum that are being developed. So there are examples of ones like Algorand, and Conflux, and Cardano, and Nervos, and there's quite a few of these. Zilliqa is another one. And I could rattle off probably

20 or 30 of these. You don’t want to spend all time doing. And what are all these things doing? In part, they're offering different programming models. In part, they're offering different scalability models. And the thing is, is that the nice development tools – Sorry. The nice deployment tools that have come out of the Ethereum community, they basically don't exist on these other platforms. And in those other worlds, you're often dealing with stuff like, “Okay, well you got to learn how to compile the special version of the development node and configure it in this special way.” So you're really sort of back in the dark ages of internet servers. And I’m not trying to criticize these consensus networks. It's just that they don't have large communities developing things for them and they are focusing on the core development of the protocol in the network.

So yeah. So then kind of another thing that's related to deployment is when it comes time to really actually launch something, what do you do? Is there something like Heroku where you just point it at your Git repository and it launches it? No. There's nothing really like that. Typically, the way that people in practice really launch DApps is they will open up Firefox and go to the JavaScript console and copy and paste their bytecode into a handwritten call to the underlying Ethereum API and send that bytecode up to the network and wait for it to be confirmed. So when it comes time, when it comes down to really deploying something in sort of the last mile, this is testing. That is really, really low-level. We have to understand a lot of the details about how these things work. And there are quite a few decentralized applications that people actively use today that don't even have frontends. The way that you interact with them is by opening up a JavaScript console inside of Firefox connected to your Ethereum wallet and you call functions on it.

[00:12:45] JMeyerson: So let's start to get into what you actually do with Reach. How do you make it easier for people to develop blockchain applications?

[00:12:56] CS: Yeah. Reach is built along three pillars. And I think that it's useful to talk about what those pillars are and then how those address each one of these problems. So the first one is the pillar of having a different level of abstraction. So when you write a Reach program, you don't think about this distinction between the frontend, the middleware and the smart contract and about how the smart contract is really this complicated object with these state transitions. You don't think about that at all. Instead what you think about is you imagine that you're basically building like a board game or like a really simple script. And when I say script, I mean script in the sense of like know what actors do, what actors follow. Not script in Bash or whatever. And what we do is we focus on what the individual participants in a program are doing.

So if we take something like an automated market maker. In that situation, there are people who provide liquidity and there are people who are traders who want to swap one currency for another. So in that program there are two types of participants, and you focus on what they do, and you write down the allowed interactions that they can have. And then what Reach does is that we determine from that what the state of the protocol must be and what transitions there must be inside of it.

So we find that this – So our intention is that this is a much higher level of abstraction that you can program at. So you don't need to worry about those low level details. But because we have this constrained language in this advanced compiler, we can actually perform optimizations on your code that would not really be plausible to have performed by hand if you were writing it really incidentally. So there actually isn't much of a sacrifice in terms of the performance in those gas prices I talked about. That's kind of the first pillar, that we want to lower the barrier to entry to becoming a developer drastically by raising level of abstraction.

The next pillar is the pillar of safety. And so one of the things that we do with your Reach program is that we take your Reach program and we basically derive from it a system of equations and think like in high school algebra where you have X plus Y has to be greater than 10, and three times Y has to be greater than two or whatever. So we have this system of equations. And this system of equations, if it has a solution, then we know that there is an attack on your program and we can tell you exactly what that attack would be.

And so we use formal verification tools, in particular SMT solvers. Right now we use Z3. And we have this model of the Reach programming language that is in perfect correspondence with

SMT programs. So we can derive the SMT program and then we can check to see if your program has mistakes. And there are mistakes that we automatically check for everybody. Simple things like can you get to the end of your program? And when you get to the end of your program, does your program have no money left over? Because if it didn't have money left over, that means that that money would be locked away forever. Or when you try to transfer money from one person to another, do you actually have that amount of money? Those are things that we check for every single program that are the kinds of things that every decentralized application should have. But then we also allow the programmer to write any assertion in their code, and this is assertion just like you do in normal programming where you would say like, “Oh, I’m going to assert that X is a positive number or that X and Y are greater than 2 or something.” And so all of those assertions we can statically check to make sure that they will always be true based on the other equations in the program.

And so what this does is it lowers the barrier to entry by making it so that you can be more confident that you got your program right. That you won't have those kinds of attacks that lead people to lose tens of billions of dollars. Then finally the last thing that we do is that we're actually a blockchain agnostic programming language. And this really ties in with that first thing. By raising the level of abstraction, we actually abstract away and hide some of the low-level details that typical smart contract developers have to worry about. And then by raising that abstraction, we can make it so our compiler targets different networks that are vastly different from one another.

So right now we support Ethereum and ALgorand and we've recently made a partner with Conflux to support them as well. And the algorithm network and the Ethereum network, I mean, could not be more different. They are extremely different backends. And we just adapt our compiler and our runtime library to target them differently.

[00:17:26] CS: So the big value of that last one is that – Like Jay said, like each way to program on each one of those is different. But the values that they provide are also quite a bit different. That way a developer can write their application once in Reach and choose which blockchain satisfies the needs of their application without having to learn a completely new set of rules and systems.

[00:17:54] JMcCarthy: Yeah. And there are really big tradeoffs between these different environments too, which we could we could talk about those, but that's kind of a little bit distracting. And, of course, behind the scenes of all this, we have our own Reach deployment and testing environment. So when you program it in Reach, it's an entirely Dockerized system where you know you can just from VS Code install our extension and then it'll download all the Docker images behind the scene. Launch the compiler. Launcher test environments for all the networks we support. So you're totally insulated from all the details of actually doing testing and deployment on all of those chains.

[00:18:34] JMeyerson: It's worth taking a step back here and thinking about who is actually building applications or who wants to build applications on blockchains today. So are you catering exclusively to a DeFi audience? Or who do you think wants to build applications?

[00:18:54] CS: So far – So we released our documentation in September of 2020. So about five months now. And since then we've had about around 200 developers come in and build applications. We don't know exactly what they're building. But our main target is the traditional developer. Not necessarily the actual blockchain developer. And the reason for that, like everything that Jay said earlier, is making it easy for them to come in and actually to learn.

Now, what types of applications that they're building is, yes, right now there's a lot of use in the DeFi space. But because we've actually lowered the cost of development, meaning that you can get to product much faster, we've completely opened this up to new types of applications. The whole goal of what I see with Reach is by making it so that more developers can come into the space and throw spaghetti at the wall to really kind of figure out what they can do is it increases the likelihood of a killer app coming out of the blockchain space. There's a great research paper out there that's called some simple economics of blockchain that does a great job of breaking down what the value of the blockchain is. And what it says is that the value of blockchain is not trustlessness or decentralization. Those are all great things that the blockchain provides. But the true value of blockchain is it lowers the cost of verification.

So for a developer to really kind of think of like, “Okay. Well, what industry can we use blockchain to really show a good ROI?” You need to look at industries that have a high-cost of verification. This is things like insurance fraud or just insurance in general. By lowering the actual cost of verification so that individuals can work with each other directly and not have that overhead is very important. But you can really kind of look at anything that has a cost verification even like social media where can you guarantee that the person that said the thing is the person that actually said it and not somebody else?

Imagine Elon Musk coming out and saying, “I’m going to sell my shares for Bitcoin.” And how big of actually an effect on the world that would be? And not knowing who that actually is could actually have a big effect without having the wrong intentions if Elon Musk isn't the one that actually said that. So really, blockchain is great for any industry that you're dealing from a P2P type of communication.

[00:21:32] JMcCarthy: I think that one of the things right now is that because the massive expense in terms of developer experience, verification burden, and this is correctness verification burden. It's basically made it so that when we look at blockchain today it's like we're looking into the past at computing around like World War II where the only reason to do computing is to like simulate nuclear explosions and to like simulate trajectories or something like that. Like these places where you’re doing extremely expensive things that have a lot of capital behind them.

And what we're trying to do with Reach is we're trying to drastically lower the barrier to entry so that people can actually experiment safely with what kinds of new things they might want to do. We're trying to create a revolution like the personal computer revolution where every kid can have a computer in their house and they can experiment with these kinds of programs. And so we want to do something like that for the blockchain space.

I think that Chris did a great job of talking about the kinds of domains that people might be looking into. I think, also, just in general, the dream of blockchain is that it will drastically lower the cost of transactions. So right now it's not really feasible to make a credit card transaction for a single cent or something like that. So what people associate right now with the concept of the micro-transaction – And I think most people now when they think of micro-transactions, they think about like in video games where you're spending five dollars. Yeah, that's a micro-transaction from the perspective of a sixty dollar game. But real micro-transactions are on the order of a few cents. And existing network, existing payment networks don't make it profitable to do that sort of thing or to provide that kind of service. But the dream of blockchains is that we'll be able to provide that far more cheaply.

And the problem right now is that the congestion on Ethereum and the huge price of Bitcoin and Ethereum basically make it so that it's not really viable to use for these low-level transactions. And so one of the goals of some of these other networks that I talked about is that they'll be more likely to be able to allow those kinds of very fine-grained transactions for much smaller purchases.

[00:23:38] CS: For example, right now on Algorand, a single transaction is 1-1000 of an algo, which ends up being a fraction of a fraction of a penny compared to the Ethereum right now where a transaction – I think I looked at the transaction to do this morning, it was going to cost $25. So that's why the strength of being able to write your application in Reach and choose the network that really satisfies your needs is so important.

[00:24:07] JMeyerson: So your vision is to be able to have a platform to write applications in, smart contracts in and have that contract cross-compile to different smart contract platforms?

[00:24:24] CS: Yeah. Our vision is monopolizing all blockchain developments. We don't want there to be Ethereum developers and Algorand developers and Conflux developers and Cardano developers. We want there to be blockchain developers and we want them to be able to share, have mind share with other blockchain developers without having to worry about the underlying networks. This is similar to the personal computer where Microsoft came in and they abstracted the hardware away so that developers could just be developers and help each other out no matter what type of hardware that the Windows or the operating system actually ran on. And that's what we're doing for blockchain, is making it so that not necessarily – We don't necessarily see a future where a developer will launch their application on multiple different networks. We just don't want them to care about that anymore. I want them to care about the features that the networks provide and be able to Alice to help Bob build an application even though that they launch their applications on completely different networks.

[00:25:21] JMeyerson: Do you think that's realistic? Like isn't that from the current point of view – Or I guess you know tell me about the current point of view like? For DeFi applications, are they all being written in raw Solidity and being deployed to Ethereum or are other platforms being used for smart contract execution?

[00:25:43] CS: Now, you're asking a very near-term question. Yes, DeFi exists on Ethereum. Yes, there's liquidity in Ethereum. But there're a couple of things that you need to kind of think about here. Number one, this is the long game. Like we're talking about blockchain being hopefully in my mind as big as the Internet is today eventually, which require multiple networks. So DeFi will exist on all of the major networks eventually. But also you can't think of blockchain only for DeFi. I mean you can think like blockchain could be good for NFTs where you don't really necessarily need that liquidity that the Ethereum currently provides.

There're also companies out there that are building bridges to actually bring liquidity from one network to another. So, today, yes. If you're going to build a defy application, you're going to want to probably put it on to Ethereum. But tomorrow, that most likely is not going to be the case. I mean like I said earlier, I literally had a transaction that I had to spend $25 to make it happen. That is not a scalable solution.

[00:26:50] JMcCarthy: For an analogy, like in the 70s, people would buy individual microcomputers and an accounting firm would buy a new micro-computer and they would have to pay a programmer to write an Assembly program for their particular microcomputer that they bought. And there were whole businesses that were, “We provide this software,” but they didn't really sell the software. What they did is they sold their services to port that software to each individual platform. And that's kind of one of the things that's going on right now with these future networks. And for traditional blockchain developers, or maybe we really shouldn't say traditional, because it's so new. But for current blockchain developers, one of the big problems that they have is this verification burden that I mentioned before.

So real blockchain programs, they essentially all have to come with hundred thousand dollar audits where a company will audit it one time. So let me just step back one moment. So you write a blockchain program and you say, “Please send in your 10 million dollars into it.” Of course only crazy people will do that. And yeah, there are lots of crazy people out there who will send you money. But most people want to see some explanation for why this program is trustworthy.

Now, when we're talking about these programs, we're not talking about a 50-line program that anyone can look at and understand what it's supposed to do. We're talking about a thousand line program written in like a sort of a mix of C and Assembly and kind of Javaness. They're not obviously right. So what most of these companies do is they will pay an auditing company to look at their program and basically write a report that says, “We are really smart. We read this program and it seems pretty good to us.” And then that's kind of gives it a stamp of approval.

Now, if the blockchain developer updates their program, then now the audit doesn't make sense anymore because it's about a different program. So sometimes they'll have to go pay for another audit. And maybe the blockchain developer will be making so many changes that they decide to internalize this cost themselves and then they'll pay yet another company, a verification tool company, to build a verification tool for them to show that their particular program is right and teach them how to become a formal verification expert and apply it to their program.

I mean, we are talking to quite a few different existing blockchain companies that are doing exactly this. They essentially have to hire the equivalent of mathematics logic PhDs to come verify their code every single time they make a change. And that becomes extremely expensive. This is what I was talking about before and I said right now blockchain is being applied to spaces like early computers where you can afford these massive expenditures of effort. And so all of those problems go away with this higher level of abstraction in Reach where we automatically do the verification for you. So even if you never use another blockchain platform, using the automatic formal verification tools in Reach pays off for that existing blockchain developer.

Furthermore, there is a space for auditing in the Reach world because you could look at the Reach program and say, “Ah! You should have had this assertion right here and that assertion over there and we're going to add those to your program.” But the nice thing is that you only have to pay for that once because now when you modify your program the assertions are still there and the verifier will check to make sure that those continue to be true.

Finally, programs written in Reach are an order of magnitude simpler than programs written in other languages because of this higher level of abstraction. So as an experiment, there's this fairly popular program called Uniswap. Maybe calling it fairly popular is an understatement. I mean I think the vast majority of all transactions on Ethereum are just calls to Uniswap. And so that program is about maybe like 1400 lines of code. And the Reach version of that program is about 100 lines of code. And so that means that it's a lot simpler to analyze it to make sure that you understand it and that it's correct and of course simpler as well to verify it formally.

[00:30:54] JMeyerson: So what goes into verifying that a high-level smart contract written in Reach is equivalent to whatever it compiles down to in Solidity? Because I mean have to do that for any kind of contract, right?

[00:31:19] JMcCarthy: Yeah. So I could talk for a very long time about this. So I’m a research professor at the University of Massachusetts Lowell and I've been a professor for the last 15 years doing formal verification research. So I’m afraid that if I get going I’m going to talk for an hour. So I will try to give a short explanation.

[00:31:36] JMeyerson: Time box it.

[00:31:38] JMcCarthy: Yeah, exactly. So basically, verification in general is a question of whether two things are equivalent. So you want to know whether or not one side of an equation is the same as another side of equation. In this case, you want to know whether or not the behavior of your actual system is inside of the behavior allowed by your specification. And so that's kind of another way of saying that you're checking to see that the program is equivalent to the specification. It's just a looser kind of equivalence, because it's containment.

Now, those two sides, the specification and your program were both written by people. And so there's kind of this meta issue, which is that like why should we trust the specification? And that's a very interesting philosophical thing. Generally what happens is that people tend to argue that you trust the specification because, A, it's written typically by other people. So it's kind of like testing one system against another one.

There's a really interesting NASA report that was written in the 80s where they had NASA software written by three different teams and they tried to evaluate whether or not they would make different kinds of errors. Okay. So it's written by other people. But the other thing is that it's written at a higher level of abstraction. So it's simpler to understand the specification and check that it's right than it is to check that the program is right. So that's the reason to believe the specification. And so those are sort of two issues. And when you use Reach, we don't get around that philosophical difficulty.

What happens is you write your program and your program embeds in it the specification. And when I say embeds in it, think of it like when you program in Java or ML or Haskell or even Typescript. You write down types. Those types don't actually do anything in your program. Instead they're just your program and like another version of it. So there's like one version of your program that actually works on real numbers, like 5, 12 and 19. And there's another version of your program that operates on types like int, string and bool.

And essentially when you program in Java, you're writing two versions of your program, the version about types and the version about numbers. It’s just that they're interweaved together.

And what the Java compiler does is it checks to make sure that the type version of the program is consistent with the value version of the program. Well, when you use Reach, you're really doing the same thing. The Reach programmer writes their program and then on the side they write down assertions, and those assertions form the specification.

And so that specification is a higher level perspective on what the program is supposed to do than the underlying value manipulations. Furthermore, we, the Reach compiler, automatically insert in a bunch more assertions like the ones I mentioned before. And for another analogy, these assertions are stuff like a traditional programmer would want to make sure that there's no pointer dereferences or that there's no array bounds overruns and things like that. So we insert these in automatically.

Because we're inserting them automatically and we're kind of like an alternative third party, this increases faith that the specification is right because it's a standard that all programs have to live up to. Now what we do is we then check to make sure that what your program does actually obeys the specification. But just like there was the potential of the specification being bad, there's the potential of this being bad. So like what is like attack or weak spot in the Reach compiler? Well, the thing is, is that we ultimately turn your program into a Solidity program and then we compile that Solidity using the normal Solidity compiler. So there could be a problem with our compiler where our compiler may generate incorrect Solidity code. And when I say incorrect, what I mean is Solidity code that does not respect the invariance that the Reach verification thought that it did.

So right now we don't have an answer to that. But the plan is, is that we're going to build a verified Reach compiler. So a verified reach compiler is a compiler written in a programming language that allows you to give extremely expressive types to your program. So a traditional compiler is like a program that takes in a C program and produces an x86 program. So that's a compiler. But a verified compiler would be one that takes in a P, which is a C program, and produces an x86 program where the meaning of that x86 program is the same as P.

Now there are programming languages like Coq and Agna that allow you to write compilers like that. And my PhD dissertation is an example of such a compiler written for cryptographic protocol programming language that is the ancestor of exactly what Reach is. So I've basically done this before, and we're planning on doing that in the future. And that C to x86 one, there's a really famous verified compiler called CompCert that's written in the Coq programming language. So that's kind of our plan with that. But then maybe we produce the Solidity program and the Solidity compiler is bad. Well, if the Solidity compiler is bad, then this whole space is messed up. So at least we're in good company with that. But that's not enough for us either. What we're going to do is, again, we're going to produce not a verified compiler test Solidity, but a verified compiler to the EVM.

Now suppose that the EVM is bad. Well, if the EVM is bad, that's sort of like the universe not matching our expectations. That’s sort of not a problem. That's just the way that the universe is. So anyways, that's sort of the current holes in where you would say like where are we not yet verified enough. And now the thing is, is that if that sounds like I’m saying, “Here are all these problems.” The important thing to realize is that all of these problems exist individually for every single blockchain program. The advantage of using Reach is that now we're centralizing those problems into one place that will solve them once and for all rather than every single blockchain developer having to worry about these issues entirely on their own. And if they have to worry about them entirely on their own, then that means that blockchain programs will only be written for extremely important, sensitive and expensive activities that can justify the expense of dealing with those problems. And this brings us back to the original mission of Reach, which is to drastically lower the barriers to entry to blockchain development so we can create a revolution in this space of having more and more interesting programs and to discover what the killer app of the space really is.

[00:38:15] JMeyerson: So it makes a lot of sense. And I'll fully admit to not being the best equipped to ask super critical questions about how to build a blockchain platform, application platform like the one you’re building. I am curious, you ran a company. Chris, you started a company back in 2018 called the Alacris Protocol, which was an operating system for blockchain applications. And I guess you worked on it for a year and eight months. And this is a developing space. I've had my own failed projects, not in blockchain, but in other things. But I’m wondering what that experience taught you and how you're carrying that into building on Reach.

[00:39:08] CS: That's a great question. So I jumped in with both feet into Alacris three plus years ago. And the original plan was to build a scaling solution for blockchain, because this was during the CryptoKitties explosion and we saw that Ethereum just couldn't handle it. So scaling was the most important thing in the entire world. As we were building that out we discovered – And we built out a prototype. We discovered that scaling isn't actually today's problems. It's going to be tomorrow's problem. And that the true problem was actually getting developers interested in building on blockchain at all because of all of the issues that we said. It’s just too difficult anybody. That even decided to try, it would take them months and months to get to a point where they could build anything simple. So we knew at that point developer onboarding and developer accessibility is the actual problem to fix. That's when we, my co-founder and I, brought on Jay, the person that’s on the call right now, to come in to Alacris to actually start building a solution to make more accessible to developers.

As time went on, my co-founder I of Alacris disagreed on the actual future of the company and I was falling more and more in love with what Jay was working on. So what I decided to do is because we were having so much disagreements, I decided to sunset Alacris and move over and focus purely on the actual development accessibility and start an actual company with Jay. And Jay was able to take all of the learnings that he spent there and rewrite Reach from scratch. Jay, if you want to talk about kind of the transition for you, that'd probably be great.

[00:40:46] JMcCarthy: Yeah. So when I first heard about Alacras, it was with Chris's co-founder, and really came on, talk about some ideas about how you could do verification for this kind of problem. As I mentioned, this has really been my research area for like the last 20 years. And just starting having casual conversations about it I sort of found that every other question was like, “Oh! Well that's the same as this research problem that I – This paper that I wrote in 2008, and this is a paper that I wrote in 2013.” And it was really natural to apply these things. And then we sort of – They were building the scaling solution, which I don't really know much about the technical details. And then we built this prototype. And then co-founders had this problem. And so by this time I was really interested in this space and this problem. So I said, “Oh, I want to keep working on this compiler.”

So then we split off and we started Reach in 2019, at the end of 2019, and had a bunch totally different ideas about how to do it after having built a prototype. And yeah, we've really gone from there. And I feel like, for me, working with Alacris was like just sort of giving casual advice to a friend about doing something. And now Reach has become everything that I think about all of the time. I was never the kind of person that wanted to be involved in entrepreneurship or doing a startup or anything like that. I’m a professor for the ability to disappear for the summer and just hang out with my kids and that sort of thing. But this is a really exciting domain. And it's cool because we're building something for the full stack developer, but we get to use all sorts of exciting advanced compilation and verification technology behind the scenes. So it's sort of exciting technically and it hopefully will solve this real problem in the world.

There are people who are in the blockchain cryptocurrency world because they basically want to make some really crazy bed and become billionaires. These are the people who bought graphics cards to turn them doing Bitcoin mining all the time. And there are people who think that they're like rewriting the world's government software. And from my perspective, I feel like we, in America and the west, we take for granted a lot of like financial institutions that really make our lives drastically better, like easy credit, fast transaction processing. And I look out at the rest of the world and we see things like India demonetizing and just putting people completely destitute by just taking away their hard currency. And we look in history at like massive inflation that people had like during the Congo and – In the Congo and in the Weimar Republic and that sort of thing. And like I think some people, they see that as some horrible tragedy where the thing that we need is just like a better government or something like that. But I see it as a tragedy of people being able to abuse their powers and the structure of their financial system to hurt other people. And I sort of see the real thing that's important about blockchain is not so much making it so that like me and my friends can play poker and bet real money with less of a middleman or that I can have CryptoKitties or something like that. I really think that this will be a way to drastically raise the standard of living across the whole world and be a major factor in helping people get out of poverty by being able to create financial institutions when their governments and people in power around them are unwilling to do something to support them, to give it more into the hands of the average normal person.

And so we've been talking this whole time about like, “Oh, formal verification this and higher level abstraction that, and assembly and this kind of thing.” But there is this real important mission behind the scenes of a company like Reach where we think, or at least I think, that this new technological platform has a real chance for extremely positive social change. But there are so many barriers to entry right now that it's really hard to make those things really come to fruition. And I am glad to be at least doing a small part of that.

[00:45:19] CS: I will say that I agree with what Jay says as well. So you can say, we, Jay. And also, Jeff, kind of talk about Alacris being a failure and the difference between the two is, yes, Alacris was a failure. I mean, we didn't even really get to a point where we had any real users. Learned a lot there, and it was the wrong bet to focus on scaling. It was too early. And the difference is night and day between Alacris and Reach where already in the amount of time that we've been around the Reach, we've generated multiple real skin in the game partners from large corporations. And large blockchain companies have actually contracted with us to rewrite their applications in Reach. And like I said, we have 200 developers since last September with a 10% to 20% growth every single week from just word of mouth.

One of the pieces of advice that actually was given at a long time ago when I started my entrepreneur career is you'll know you have product market fit when the market pulls you. And that is what we feel with Reach is that anytime somebody actually puts in the effort to go through a tutorial, they don't ever want to look back. And that what feels real special to me.

[00:46:37] JMeyerson: Do you think that the near future of applications on Blockchains is just more DeFi? Or do you see any other categories of applications being built in the near future?

[00:46:53] CS: Yeah. So DeFi is definitely near future, because I mean if you just look at the stats, it's growing exponentially every day. So that's not going away. What I see though is that you're going to start broadening into more applications. NFTs specifically right now are really starting to actually pick up. And NFT is a non-fungible token. It's just a representation of a thing that can't be divided. So you can kind of see this as collectibles. Like CryptoKitties is an NFT, but you can actually see them being integrated into games. So think about World of Warcraft, which many, many people play. You could turn your weapons, your armor into NFTs and they could be traded and you could actually own them yourself as an individual. And we'll see the blockchain gaming really blow up or just gaming with NFTs tied into it in this year as well.

But I mean, once again, this is just the start. It's impossible to tell you what the next killer app is, because if you could predict what the next killer app would be, VCs would only invest in one company, not in 10 of them and hoping one succeed. So like as more people come in, start thinking about things, get creative and start building new things where it decreases the cost of verification and increases the ROI for many different types of applications, it's going to get really fun in the near future.

[00:48:19] JMeyerson: When you look at DeFi, what utility do you see it providing? Is it just a big casino where people are essentially betting on different coins and not really having any fundamental value? Or is it actually providing capital to people who need it?

[00:48:42] CS: This is something I talk about a lot, and I have a – Me being 100 in crypto and talking to people that are in DeFi, I might have a different view than a lot of people out there. But today, it is a lot about leveraging your current positions in the cryptocurrency market. And depending on how you actually look at cryptocurrency, you could see that as gambling. But then isn't most things gambling if you're really kind of drill down to the actual bare nuts of things? But what I see the true value today is more in actually understanding game theory and tokenomics or economics of the future types of applications. So what DeFi is providing us today is understanding the proper incentives to actually build a decentralized applications in the future. So that's one of the big great things that DeFi is providing today.

[00:49:34] JMcCarthy: A question that you might have is like what is the difference between a coin and a stock price? So the difference of course is that there was an initial public offering of that stock and that sale capitalized the company and the company continues to pay dividends and potentially does share buybacks. And those are ways that investors in the company actually make real revenue and produce real value from the stock price trading.

And so the question is, “Is there something like that with the tokens that are out there in the DeFi space? Of course, like when you look at the DeFi space, there are tools like market makers, there are tools like yield farming. Really, you want to think of those as like the kinds of financial instruments that you could build in the normal finance world like features and things like that. But in terms of like the core, like IPO stock capitalization process. In that thing, that does actually happen in this space. So just as a really concrete example. Take a network like Algorand. So Algorand has a token, the algo, that currently I think is like trading for like 33 cents or 35 or something like that, somewhere around there. And there was an initial coin offering of that where the foundation behind the company retained a huge number of the initial algo tokens. And the investors who bought tokens, the money that they bought them for capitalized the Algorand company, which allowed them to fund the development of it just like when you did an initial coin, initial public offering of a company like Facebook or something like. That capitalized it so that they could expand. So that initial sale funded our Algorand so that they could build the product. And as the share price changes. Sorry. As the token price changes, it allows Algorand the foundation to sell some of those tokens for cash to continue making improvements.

Where does the dividend process happen? Well, the Algorand Foundation can buy back algos, which gives things to investors. Furthermore, one of the things that happens in some of these protocols is that rather than explicitly paying dividends, dividends come as a consequence of running nodes in the network. So if you run an Algorand node, then you get a portion of basically the mining proceeds that people think about across the entire network. And if you hold algos on an exchange like Coinbase or something like that, then they run nodes and then they pay those to you. So those initial investors, basically they're seeing the dividends in the form of those staking rewards and changing of the price.

So all of the normal things that we think about happening in the normal stock market, they happen in the cryptocurrency space in some cases. Of course, in other cases, there are coins that are just pure speculation that have no intrinsic value. They're not tied to anything. And I think that there are really crystal clear cases. There are really vague and murky cases. But I think that this sort of real value question is one of the biggest things. I think that this space will totally change if we can figure out some legal way to electronically trade actual stocks and actually pay dividends and really invest in real companies in the cryptocurrency space.

[00:53:06] JMeyerson: Well, we're up against time.

[00:53:08] JMcCarthy: Oh, by the way, I am not a lawyer or an investment banker or anything like that. So I'm not giving advice about or explaining in detail about what the rules are with Algorand or anything like that. I'm not really an Algorand spokesperson. It's just an example because they're one of our partners. So I think about them all the time.

[00:53:27] JMeyerson: Got it. Well, just to wind up because we we're almost out of time. Do you guys have any other predictions on speaking in 2021? We're in the midst of a bull market. Can give it one short term and one long-term prediction about crypto?

[00:53:44] CS: I'd say short-term prediction is that you're going to see the rise of custody this year. You're going to see banks starting actually holding crypto for consumers, because holding keys is something that traditional regular people probably don't want to do. So you'd see a lot of custody services that are insurance-backed. So you'll see a lot of that. And long term is – I tell this to everybody, is that blockchain will be bigger than the Internet is today. And the reason why I say that is if I ask you the question, "Which type of network is larger than the Internet?" And the answer is the financial markets. That's a larger network than there is today. Now if you combine finance and just currency in general with Internet, that's where you have blockchain. So in 5, 10, 15 maybe 20 years, you're going to see where the amount of value in the blockchain is going to be something that would blow our mind today looking at what it is.

[00:54:42] JMcCarthy: My short term prediction is that there will continue to be incredibly befuddling hacks of people losing tens of millions of dollars and people standing around befuddled about why they didn't use verification tools. And people will be scared away from blockchain because of this and people will be attracted to blockchain because of the prospects of doing it right. That's my sort of short-term prediction, more of the same. And my long-term prediction is that in 100, 200 years, we will look back at blockchain in the same way that we look at something like double entry bookkeeping, where the category of person that is the stereotype of a boring person is an accountant. Because what was an amazing financial technological innovation hundreds of years ago is now completely boring in routine and we don't even consider it worthwhile talking about often when we're talking about exciting interesting things that have happened. And I think that blockchain will just be a fundamental part of the way that we think about organizing decentralized systems. And it won't even be controversial or interesting in a long time. It'll just be the way that things are, just like something like packet switching, right? Like 50 years ago, packet switching was a really wild, cool concept. And now nobody even talks about packet switching as an interesting idea anymore. It's just of course that's the way that things are going to work. Of course we're not going to build point-to-point connections between everything that we want to talk to. That would be crazy.

[00:56:10] JMeyerson: Okay. Well, that sounds like a good place to wrap up. Guys, thank you so much for coming on the show, and very interesting to learn about Reach.

[00:56:18] CS: Yeah, thanks so much.

[00:56:18] JMcCarthy: Thanks for having us.

[END]