EPISODE 988

[INTRODUCTION]

[00:00:00] JM: If the internet was completely reimagined with the software and hardware infrastructure that we have today, what would it look like? That is the question that DFINITY is working on answering. DFINITY's goal is to build a decentralized, secure internet computer. DFINITY takes concepts from the cryptocurrency world, but it's focused on computation, not financial products.

DFINITY can be thought of as a decentralized cloud provider with redundancy and scalability properties that are achieved by operating on data centers across the world. DFINITY wants to host web applications such as the ones that we use today on centralized servers. A developer who wants to run their application on DFINITY compiles their code to WebAssembly and deploys it to the DFINITY centralized runtime.

Transactions across DFINITY applications are processed through a collateralized proof of stake system to ensure reliable decentralized computation. DFINITY is an ambitious project and it would seem nearly impossible to bring to market if not for the quality of the team. DFINITY has hired Andreas Rossberg, one of the co-creators of WebAssembly, and the company has also hired many talented engineers across security, web development and backend infrastructure.

For people who have completely turned away from anything related to a blockchain or running gas or decentralized computation, this is a show to listen to. It's definitely a departure from completely traditional web infrastructure. It definitely has a component of the broad ambition of the crypto community, but it is a nice merger between the practical normality of centralized computing and the ambitions of decentralized computing. It has a real strong software team.

Dominic Williams is the president of DFINITY and the chief scientist and he joins the show to talk about the vision for DFINITY and the roadmap to making it a reality.

We have partnered with SafeGraph for the SafeGraph Data Hackathon Challenge. We're giving away $4,000 in cash prizes as well as SEDaily and SafeGraph swag. SafeGraph is a geospatial data company which curates a dataset of more than 6 million points of interest, and SafeGraph provides a high-volume of location data. You can use SafeGraph to build apps and data science projects with that data, and if you've been looking for a creative opportunity to explore large datasets to try out data science, you have the potential to win $4,000 in cash prizes and have a lot of fun. This is a great opportunity to tryout data science.

The hackathon is hosted in FindCollabs and you can enter the contest by going to findcollabs.com and signing up. If you're planning a hackathon yourself, you can check out FindCollabs Hackathons. FindCollabs is a company I built and our new hackathon product allows anybody to create a hackathon and manage it and run it and award prizes and create rules. If you're planning a hackathon, check out findcollabs.com.

[SPONSOR MESSAGE]

[00:03:16] JM: Redis is a fast in-memory database system. Engineers have been using Redis for more than a decade because of its reliable object caching, but that's not the only use case of Redis. Redis can be used as your operational data store for queuing, streaming and other data applications.

We recently had an episode of Software Engineering Daily with Alvin Richards of Redis Labs in which Alvin described the use cases of Redis, and I enjoyed learning about the flexible architecture and how Redis uses memory and persistence to create an API that solves a variety of problems. You can listen to that episode or you can go to redislabs.com/sedaily to find out about how Redis can help as a data layer for your microservices.

Redis Labs is the company that makes Redis Enterprise, which offers performance, reliability and professional assistance with your Redis instances. If Redis is on the critical path of your application, go to redislabs.com/sedaily and learn about Redis Labs as well as some of the design patterns for Redis that you might not have seen before. That's redislabs.com/sedaily.

Thank you to Redis Labs for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:04:41] JM: Dom Williams, welcome to Software Engineering Daily.

[00:04:42] DW: Thank you for having me.

[00:04:43] JM: You're building DFINITY. The goal of DFINITY is to build a decentralized secure internet computer. Explain why that would be useful.

[00:04:52] DW: Well, DFINITY is a foundation and it's not building DFINITY. It's building something we call the internet computer. I think there'll be lots of internet computers, there'll be one internet computer. It's created by a product that combines compute capacity from independent data centers to create a compute platform that reimagines how we build software systems and internet services.

[00:05:15] JM: What are some applications that we could build with an internet computer that we cannot build today?

[00:05:21] DW: Well, the internet computer will enable you to build enterprise systems, internet services, websites and so. Very broad range of systems and it's intended as a complete replacement for today's legacy stack. Reimagines how a software works. You don't need databases, web servers, middleware, cloud services, content distribution networks, DNS. You can essentially write code and just push it on to the internet where it'll run and be hosted within the protocol and you can even serve user experiences directly from the internet computer into web browsers.

[00:05:58] JM: What are the shortcomings of the current status quo internet relative to the internet computer?

[00:06:05] DW: Well, today's internet, it only provides connectivity. The network itself is formed by a protocol called IP which combines thousands of private networks around the world. Anything from a nice a nice piece network, to a corporate network, to a sort of transit providers route and into a single global public virtual network, okay? But it only does connectivity and then there are high-level protocols like HTTP that connect hypertext webpages, for example.

But if you want to build a system or an internet service, you need components of the sort of legacy stack. For example, let's say you wanted to create a website. Probably you'll get an Amazon Web Services instance. You'll choose a database. You'll choose, well, obviously an operating system, a web server, a scripting language and so on and so forth and you'll assemble your system from all these different components.

There are problems with that. I mean, first of all, the vendors of all these components want to make you a captive customer. Second of all, it's impossible to make a secure system. It relies on constant patching of software, correct configuration of the firewall and so on, and we'll know about the dangers. Someone can hack your system, encrypt your server with ransomwares, steal your confidential data.

Also, it's very expensive. If you look at the breakdown of costs, the majority IT operations, i.e. people. If you really analyze what people spend the majority of their time doing, especially in enterprise software, it involves managing complexity. Trying to integrate different systems and resolving integration bugs and so on. We attack that. Finally, we also address some of the difficulties involved in creating internet services today. We live in a very monopolistic ecosystem and we want to change how internet serves as a belt.

[00:08:14] JM: How does your perspective on what an internet computer should be compared to let's say the Web 3 visions that were described in detail over the last couple of years by various people. Maybe people who are writing about combining Ethereum with IPFS, like that kind of stack. How would you contrast the architecture of the DFINITY stack?

[00:08:43] DW: Well, the internet computer involves a lot more computer science than something like Ethereum, right? Ethereum is essentially a scripting language overlaid on a proof of work blockchain. I'm not even sure you can describe the internet computer as a blockchain. Software running on the internet computer gets the same security guarantees as small contracts, but it's designed to compete with the mainstream stack. It has unlimited computational and storage capacity. It can serve request extraordinarily quickly. You can't really compare it to sort of these blockchain systems. The security problem is the same, but the way it works is very different.

[00:09:29] JM: The most primitive description for a computer is like the Turing machine, basically. The Turing machine model for a computer is just very simple. You could arguably build a Turing complete system on top of Bitcoin scripting language, on top of Ethereum scripting language. You would have to have layers and layers of abstraction in the case of Bitcoin, because Bitcoin has this time bound property where the scripts will run out of time eventually. They can't execute forever. In the case of Ethereum, they're just so slow. I mean, they're both very, very slow, but you could build, whatever, side-chains or lightning network-like stuff on top of it to eventually get to a Turing complete computer.

Then it seems conceivable that you could build all of the consumer technologies on top of those Turing complete levels of abstraction at some point. I'm just trying to understand the –

[00:10:30] DW: I have no idea how that would work. I mean, at least I'm not aware of any projects that you'd have blockchains and host them on contracts that have good performance that's infi nitely scalable that provide software engineering models that would allow them to compete against the legacy stack.

I think Bitcoin and Ethereum exist in the cryptocurrency world and they have their own set of objectives and criteria on which they evaluate success. The internet computer isn't competing with Ethereum and Bitcoin. It's competing with the today's legacy stack. We want to provide people with a different way of building software systems, enterprise software systems, internet services, the whole thing.

I mean, it's true that the internet computer can be used to host small contracts, but that's like a set of tiny niche application. What we care about is can you build open Salesforce on the internet computer? Can you build video streaming, open Netflix? Can you build open WhatsApp? Can you build open LinkedIn? Can you build a messaging, like a replacement for something like Gmail? Can you build enterprise systems? Can a large fortune 500 company trust it to build its inventory management system?

We're not suggesting that people use the internet computer as some kind of small contract or logging platform or something like that. We're suggesting that people completely throw away everything. It's an alternative system. It's not designed to particularly work with. I mean, you can integrate it with the existing stack. It's designed as a replacement for the existing stack.

[00:12:06] JM: Right. The replacement, is it – I just really want to highlight like the higher level before we dive into the lower level guts. The replacement, like if you compare it to the contemporary stack, it's the idea that it's decentralized, it's transparent in its execution. It's transparent in its code. It's fully transparent?

[00:12:28] DW: Look. I mean, first, the word decentralized is very loaded. I mean, our view is that people will choose to use the internet computer for much more concrete reasons. Broadly speaking, I supposed there are four. One is that the captive customer, you're building on the internet, and building on the internet itself is very different than building on Amazon Web Services using Microsoft.net and internet information server and some database where you're going to become a captive of those different platform stack components, right?

First of all, we want to enable corporations to build systems without becoming captive customers of all these legacy stack vendors. Secondly, we want to solve the security problem. We think that the legacy stack is meltdown. It's impossible to make systems secure. It relies upon people creating these kind of assemblies of systems, databases, web services, servers, middleware, systems, firewalls, VPNs and so on and then correctly configuring them and constantly patching them.

Sooner or later mistakes are made and someone gets in. We think that it's impossible to build a secure system today and we're sort of reversing the model so the systems are secured by default and you build an enterprise system on the internet computer. You don't have to protect it by a firewall and you don't have to patch the internet computer to keep it secure. It is secure by default. It's a tamper-proof environment. It provides the same security properties as smart contracts do, for example.

Secondly, we're solving security problems of today, which we think are very pressing and only going to grow. Thirdly, we're addressing the cost of building and maintaining systems, which we think are exorbitantly high owing to the way that systems are built today. Fourthly, we're providing a way to create a different kind of internet service that can compete with today's monopolistic internet services like, for example, LinkedIn, Salesforce, WhatsApp, Instagram, you name it.

[00:14:30] JM: To get into the scalability model of DFINITY, I'd first like to dabble a little bit in the scalability architecture of Ethereum and a little bit about the struggles in implementation details there. When you look at the scalability of Ethereum and the scalability bottlenecks of Ethereum, what do you think are the most acute problems and how viable do you think are Ethereum scalability solutions?

[00:15:02] DW: I mean, look, I was involved in playing around with blockchains probably 5+ years ago, and it is true that DFINITY was originally conceived as a solution to some of the challenges that Ethereum faced, but it's become something very different. You can draw analogies between the design of Ethereum and Bitcoin and the design of the Internet computer protocol.

It's probably fair to say the internet computer protocol is orders of magnitude more complex. Leans on modern cryptography and a completely different architecture and there is no direct parallel. The Internet computer is not just a faster blockchain.

[00:15:46] JM: Can you give a brief overview of the DFINITY blockchain scalability model, and then we can go into more specifics about it?

[00:15:55] DW: First of all, DFINITY is a foundation, not-for-profit foundation based in Switzerland, and it has several search centers around the world that work on the Internet computer protocol and software implementations that data centers can run such that the Internet computer comes into being. We probably got the strongest distributed computing and crypto team in the world, but I'm not comparing it to blockchain [inaudible 00:16:23]. I'm comparing it to Google, and Facebook, and Microsoft, and IBM and so on.

The Internet computer itself involves a lot of cryptography, a lot of distributed computing, a lot of virtual machines science. For example, the low level instruction format we use is WebAssembly. The primary creator of WebAssembly, Andreas Roassberg works at DFINITY. We're also developing new languages. For example, at the end of last year, November 1st, 2019, we released an SDK with a new language called Motoko. There are other SDKs coming, a Rust SDK and a CSDK. It's pretty broad projects that touches on many different areas.

There are some components that we have talked about. We don't generally talk about the work of the last few years because whenever we release information, we find that we don’t really get any credit for it, but everyone copies us, right? But some of the pieces that are told about, like for example threshold relay. It’s a mechanism for creating random numbers in a decentralized network that is unmanipulable, unstoppable and unpredictable. We use those random numbers to drive various other protocols. That's well-known.

[00:17:39] JM: The delegation selection.

[00:17:42] DW: We rely on threshold signatures, multi-signatures, all kinds of things. There’s a lot of cryptography, we have to have like 20 the world's best cryptographers on the team. Even one of the pieces of cryptography that we use will probably requires set of substantial amount of explanation and sort of PhD level knowledge of crypto to really get a handle on. But all of these stuff – And we’ve started releasing technology into the wild.

The end of last year, we pushed out an SDK which reveals various aspects of the programming model together with this language called Motoko. There’re a number of releases taking us to the public launch of the network hopefully later this year. The SDK was copper release. We got a bronze release coming up, which demonstrates Internet computer that’s being incubated at the moment. It’s data centers and we’re actually going to also demonstrate an open Internet service called LinkedUp that’s been built on top of it.

We think that will be a sort of another significant sort of milestone in terms of what we pushed out into the public domain. Then following from, there’s a release called Tungsten, which will make the incubated Internet computer or make access to the incubated Internet computer more widely available.

[SPONSOR MESSAGE]

[00:19:16] JM: Looking for a job is painful, and if you are in software and you have the skillset needed to get a job in technology, it can sometimes seem very strange that it takes so long to find a job that's a good fit for you.

Vettery is an online hiring marketplace to connect highly-qualified workers with top companies. Vettery keeps the quality of workers and companies on the platform high, because Vettery vets both workers and companies access is exclusive and you can apply to find a job through Vetter by going to vetter.com/sedaily. That's V-E-T-T-E-R-Y.com/sedaily.

Once you're accepted to Vettery, you have access to a modern hiring process. You can set preferences for location, experience level, salary requirements and other parameters so that you only get job opportunities that appeal to you.

No more of those recruiters sending you blind messages that say they are looking for a Java rockstar with 35 years of experience who's willing to relocate to Antarctica. We all know that there is a better way to find a job. So check out vettery.com/sedaily and get a $300 sign-up bonus if you accept a job through Vettery.

Vettery is changing the way people get hired and the way that people hire. So check outvettery.com/sedaily and get a $300 at bonus if you accept a job through Vettery. That's V-E-T-T-E-R-Y.com/sedaily.

Thank you to Vettery for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:21:06] JM: One thing I did I'd really like to highlight is the fact that if you want to have distributed computation or decentralized computation in the way that cryptocurrency people typically talk about decentralized computation, you need to have some kind of breakthrough in scalability to have all of the transactions and the Internet computer propagate in a decentralized fashion, right?

[00:21:37] DW: Well, I mean I think – Look. There are two things. First of all, there's a lot of, frankly speaking, crap talked in the cryptocurrency community. There is a lot of sort of phony engineers and scientists that don't really know what they're doing. You just can't debate it with them, because you end up going down all kinds of blind alleys.

[00:21:56] JM: I mean, assume I'm your smartest blockchain engineer.

[00:21:59] DW: The last time I tried talking to someone in that space to explain some technical things, I ended up and I just remembered they turned around to me and said, "What about weak subjectivity?" I said, "What about weak subjectivity?" This was something. Some theory of Vitalik. I mean, we got the best scientists and engineers in the space on our team and all of the technical materials will be released in due course and people can make their minds up about how –

[00:22:33] JM: I mean, the stuff that's already been published is impressive. The delegation –

[00:22:37] DW: Threshold relay and PSE.

[00:22:39] JM: Yeah. Well, and just the concept of – I mean, maybe this was not novel, but my impression is that it was novel. The idea that when, obviously, every transaction needs to be processed by some number of nodes and the population of nodes are going to be chosen via this trusted delegate situation where the delegates all have to have some kind of skin in the game. They all have to have some – There's a proof of stake model.

[00:23:10] DW: Yeah. The Internet computer leans much more heavily to the Internet model itself. It's not proof of stake. You can't like take some cryptocurrency moniker and use it to explain the model that it had, the technical architecture and the model.

[00:23:27] JM: Elements of proof stake though, right? I mean, people have a stake. If you have a stake in the system, you can get chosen as a delegate to approve a transaction.

[00:23:35] DW: No. No. Generally, first of all, the Internet computer is created by independent data centers in the same way that the Internet is created by independent networks. In the case of the incident, there’s a protocol called IP that combines thousands of private networks worldwide to create a single global public network, a virtual network.

In the case of the Internet computer, there’s a product called ICP that combines compute capacity provided by traditional independent data centers to create this global virtual computer, the Internet computer, which is something like a seamless universe for software where you cannot load software. There’s no files. We have a think called orthogonal persistence. If I load some software and you upload some software, my software can call into your software if you’ve given me permission. It’s seamless. There’s no concept of distribution.

The same way the Internet is created a protocol called IP that combines private networks to create a global public virtual network. The Internet computer is credit by protocol called ICP that combines computer capacity from data centers to create a single global public virtual computer, the Internet computer.

Within that network, that Internet computer acts more like a cloud in the sense that it is designed to store unimaginable amounts of data processed trillions of transactions a second. There's no way that an individual could possibly install some clients and get a copy of all those transactions in all that data in the manner of a blockchain. It just wouldn't work.

The access to that, the role of data center is permissioned, but is permissioned by an open governance system that runs on top of the network called the nervous system, meaning that those ideas had been sort of – I think I talked about them some years ago. I think it was originally called the blockchain nervous system which we’re having a laugh about that it was proposed back in like – I don’t know, 2016, 2017. That concept, we’re still running with the concept, that you have an open governance system sitting atop of permission network. That’s similar to how the Internet works.

The Internet has ICANN and then there's a part of ICANN called IANA. For example, IANA will grant you an ASN number that allows you to run a BGP router, and you need to run a BGP router if you want to connect to multiple transit providers, say, BNISP or a hosting center. The difference is that the Internet was obviously designed many years ago. It was impossible to create an algorithmic governance system, whereas the internet computer is a much more recent thing and it’s a computing platform, so it can have an onboard governance system.

[00:26:31] JM: Can you explain how the DFINITY blockchain and the currency fits into the Internet computer model?

[00:26:42] DW: Again, I’m not sure blockchains an accurate description for the Internet incident. There is –

[00:26:49] JM: Or just the currency. You can just explain from the currency standpoint.

[00:26:50] DW: Yeah. I’m not even sure currency is the right word. It's designed as a computing platform. Now, it is true that computation and storage involve gas and there’s a very simple reason for that. You have to mediate participation and control of the network somehow. It's an open decentralized network and there has to be some means for mediating participation and control.

If there was no gas, for example, somebody could just upload some software that just stores an ever-increasing stream of random bites and all of the capacity of the data centers would eventually be considered. You have to have some means of mediating participation and control. We have gas.

The difference is that we do not expect users of the Internet computer to hold tokens, right? One of the great difficulties I think people have with some of these early experimental systems like Ethereum was that you could build a service, like CryptoKitties, but then you had two problems. Firstly, only users that holes Eth tokens can use it. They had to go to Coinbase or somewhere for first to buy them. Then, secondly, it’s not really a decentralized service, because actually there are some small contracts and then there’s a website frontend running on Amazon Web Services.

That kind of architecture doesn’t work for our purposes. We want people to be able to build, say, open LinkedIn and for users to be able to interact with open LinkedIn without acquiring any tokens. There isn't a cryptocurrency model. It is true however that in order to get gas, you need tokens. But for the most part, the software pays for its own gas and we think that people go over intermediaries to get tokens that converted to gas.

Typically, say for example you are an enterprise user and you had some kind of enterprise system running on the Internet computer. You might just go to an intermediary, give them your credit card and say, “Hey, put $100 worth of gas on my software. Here’s the ID.” You wouldn’t go near tokens. Yes, in the Internet computer you can see influences and history related to the whole blockchain movement, but it's a very different thing.

[00:29:31] JM: So let's say I am going to a webpage that is hosted on the Internet computer. Presumably, I can go into my web browser and enter in the domain name and I press enter and then that request propagates through eventually a stack that is different than the present day computer. Can you walk me through the architecture? How does that request propagate through the stack and how does my webpage eventually get sent to me?

[00:30:03] DW: Well, first of all, the Internet computer is integrated with DNS. I can’t give you the full details, but you could imagine that a domain name would be put in. It would resolve to some symmetric replica of your system, probably one that’s near to you and sort of default function calls would be made by the boot loader to retrieve the components of the user, the UI. For example, it might be React. It might be fragments of HTML and JavaScript and so on. They'd come down together with some cryptography that allowed the boot loader to verify that the user-interface hadn’t been tampered with.

The user interface would then start making REST calls to the Internet computer, one of which of course would be done in a cryptographically secure way, although the user would have nothing to do with this, and that there are two kinds of calls on the Internet computer. One is an update call, which modifies the state of the – It’s got a software canister, and that obviously has to be run across all the replicas, it’s a symmetric replication systems. Depending on –

[00:31:10] JM: Replicas of the website.

[00:31:13] DW: Yeah, and all the software and data behind it. Yeah, it's completely self-contained.

[00:31:18] JM: Every applications is completely self-contained.

[00:31:20] DW: Yeah. It doesn't have to be, but why not? Yeah, that's the best way of designing it. The uptake call, we replicate it across all the replicas in the sub-network that hosts your canister or canisters, and that might take a little longer, sort of like one or two seconds depending on the size of the sub-network possibly. Possibly even as long as five seconds, but that would be an outside case.

The other kind of call is a query call, and a query call doesn't modify it state, or least any modifications that state makes are persisted and without getting into how this is done securely, this would typically be answered by a single replica and would happen obviously extremely fast. Faster even than it happens today with legacy content distribution networks like Akamai, because you've got symmetric replications.

Let's think about an example. Let's think about an open Reddit. Okay? You type in some domain name. It's integrated by DNS. First of all, the components of the UX load into the browser in a way that allows the boot loader to verify that it haven't been tampered with, because you end-to-end security for this to be meaningful. That's something that's missing from systems today where you have things like blockchains and then websites hosted by Amazon Web Services. How do you know you're interacting with a blockchain?

The UX loads into the browser and probably you've got an identity, which of course is a public key. Previously, if you like signed up to open Reddit, possibly the homepage is displayed to you is customized, right? Maybe it's like your own personal feed which combines posts from different forums that you're interested in.

You might click on some of these forums to drill down or post to read it, and all of this happens very much instantaneously because you're communicating with the nearest replica to you. This compares very well to today's model where Akamai can – You can pull like video files and other sort of images and so on and React fragments from the edge from a server near to you. But if you want to actually interact with the source and get like a customized homepage or something, well, what Akamai will do is root you across the network of premade connections to reduce the latency, but you still interact with the source server.

If you are in London and the source server is in Tokyo where you got 480 milliseconds of latency, right? But it's different here, of course, because now when you're getting your customized open Reddit form listing, you’re actually interacting with a local symmetric replica of all of the data and software. Better performance than today's Internet built on sort of the proprietary legacy stack.

Then you might want to post something, and that will take a little bit longer because you’ve got the – That would be involved and update calls. It has to be run across all the symmetric replicas in the Internet computer sub-network which hosts the software. You might post something and let’s just say it takes –You see the little spinning wheel for two seconds. Big deal, right? What you really care about is that when you're browsing open Reddit, it's super snappy and we’ll be able to do that better than people can today. It's only when you actually want to update the data that it takes a little bit long. We think that two seconds is reasonable.

In the end, people will prefer these kind of services not just because the performance and things like that, but because they’re secure. They can be open and give them guaranties about how their data is processed and there are a whole bunch of other things the Internet computer allows you to do, which people – There’s just no way of doing today. For example, you can guarantee APIs. You can mark APIs as permanent. When you've done, say, as a creator of open Reddit, not only will the Internet computer prevent you from deleting that API when you upgrade the software either as an individual or through a governance system, which would make it autonomous. It can also even reverse degradations of API functionality.

This lays the groundwork for as a most sort of collaborative dynamic Internet ecosystem where people can freely extend services, pre-existing services by relying on the API calls that give them access to the user relationships and data. Even if in the case that let's say open Reddit one starts developing very slowly and everyone gets very dissatisfied with the developers behind it that a new team can come along and create open Reddit two and pull the data around of open Reddit one.

We think that's going to create much better network effects in a much more dynamic, creative, innovative Internet ecosystem that progresses faster. In the end, it out-competes today's monopolistic system.

[00:36:21] JM: That's a great answer. The canister model, this canister, if I understand correctly is the abstraction that every application replica fits within?

[00:36:31] DW: Yeah. The computational unit used by the Internet computer is called a canister. You can probably imagine why it's called a canister. It's a bundling of software and state and the Internet computer moves it around between sub-networks depending upon load and other things, protocol-related needs. I canister doesn't have any files associated with it as such other than some sort of resource files that you might be using for the user experience.

The canister persists main memory. Canisters introduce something called orthogonal persistence, where developers don't think about persistence at all. They just write their abstractions. You could, for example, create a very simple social network by writing canister code akin to the people equals new dictionary, angle bracket, string for these name, profile. Closing angle bracket. Essentially, the Internet computer persists memory pages and it's based around an actor model..

[00:37:38] JM: Can you describe like maybe the storage footprint or the memory footprint of a canister? I'm just trying to understand like how many canisters it would take to run my blog or something, like my podcast.

[00:37:53] DW: Yeah. Obviously, we want to get the Internet computer out. The first version of the Internet computer won't be as capable as subsequent versions. At least from where we are now, due to restrictions with the design of WebAssembly that are going to be lifted. Currently, a canister will have a sort of, won't be able to start off 4 gig of memory. But that's the restriction we're working on solving very actively, and it wouldn't be around that long.

That means if you wanted to create an Internet scale system such as LinkedIn, say, you would need to use multiple canisters. Typically, you'd use canisters as buckets for user data. The two important points there, the first point is this limit on the amount of memory that I canister has is going to be lifted and they're going to have vastly more capacity soon. Secondly, anyways, if you want to create an Internet scale system, you're always going to have to think about partitioning your data, right?

Bus the canister model, and for example, the Motoko language provides ways to make that much easier. When one canister makes a call into another canister, that call is asynchronous, which means you'd sort of do in your code some other canister.function call and then you'd write the handler. That handler would be – It's like a callback, like a closure executed asynchronously.

For example, the Motoko language which we released November 1st last year provides something called the async keyword that makes it possible to write direct sequential code within this asynchronous environment. One of the challenges that has existed historically is that it's very difficult to write sequential code that scales out, but you can't really.

Motoko introduces something akin to the async keyword in EC6 JavaScript that is more powerful that makes it possible to write direct sequential code that can scale out. Crediting a multi-canister project may not be as onerous as you might imagine.

[00:40:28] JM: Let's say I just have a blog, like the blog has three posts. One of them is Hello World. One of them is My name is Jeff and the third one I like computers. What's the process of deploying that to the DFINITY canister?

[00:40:42] DW: Well, I mean that's obviously a very simple application. You would create your canister software. Let's say you wrote it in Motoko. This canister would share some functions. You'd permission those functions appropriately. You would mark them as either query functions which don't modify the state or update functions that do modify the state. You would create these resource fi les that implement the UX, which are actually accessed via default function calls.

I believe in the current implementation, those results files are actually kept in main memory. I'm not sure whether that's going to stay the same, but that's the current way it works. You would test and develop your blog canister locally. When you are happy with it, you'd push it to the Internet computer.

This canister would have an identity, which is a public key. You would hold the corresponding private key, and using that private you could update that canister in place and something you may have noticed I haven't mentioned is there's a complexity relating to the heap becoming incompatible with the new software.

The technical difficulty with orthogonal persistence is that you need to transform the heap, migrate the heap when you upgrade the software. Motoko actually has language extensions that make that possible. It sort of does it for you in a type safe way, but there are other models. The RUST SDK will do it its own way. The Internet computer itself defines the canister framework at quite a low level. You got some software. You've got some low level API calls that the runtime can make to call into other canisters, and you've got some WASM linear memory. You can imagine soon it will be multiple memories and so on to break passed, move past this memory limit.

[00:42:46] JM: Can you help me understand once my blog is deployed to the Internet computer, when do gas payments come in? Am I going to need to pay gas? Do other people pay gas? Where does that gas go?

[00:43:02] DW: Some of these details we can't talk about fully right now, but generally speaking, I don't believe it's possible to design a system like this without gas, because otherwise there's always some security vulnerability. People ultimately have to pay for the computation that's consumed and there can't be any loopholes. But typically, you yourself would charge it with gas, maybe not directly through an intermediary. That gas will be consumed by query calls, which also include serving the UX and update calls. There are mechanisms in place that make it defend against a gas exhaustion attack. If that makes sense?

[00:43:42] JM: Yeah, sure. I can host my – I mean, it's just like I'm paying for a web hosting provider. Instead I'm just paying –

[00:43:49] DW: Yeah, exactly, It's more akin to the traditional model, except obviously done in a way that allows this to work securely in a public network.

[00:43:59] JM: Right. Okay.

[00:44:01] DW: Yeah. With an enterprise system, obviously, it’s more straightforward. You can imagine if your blog was only for consumption within a company, well, a systems administrator could permission that blog appropriately. Only people within your company will be able to use it and consume the gas. You need to do some different things for a public Internet service, and that's – Probably we’ll not go into details to that right now. I mean, a lot of this stuff is going to be pushed out imminently this year.

[00:44:34] JM: What are the security guarantees once I deploy my blog to DFINITY and how are those security guarantees manifested?

[00:44:45] DW: Well, assuming that the blog is running on a sub-network with a sufficient level of replication, and bear in mind, replication is two purposes. One is to deliver security, and another is potentially to scale out query calls.

For example, if you created open Netflix, very likely as it became more popular, the sub-network would scale out the number of symmetric replicas so it could increase the streaming capacity of the system. The other purpose is security. Let's imagine that it had a replication factor 28. It uses a traditional presenting fault-tolerant sites, a bit new science, but has the same [inaudible 00:45:23] bounds.

It could withstand nine of those becoming faulty and it will continue without a hitch. The way the Internet computer works, the replicas in that sub-network hosting your blog would be independent data centers. By default, there are lots of possible configurations, right? By default, you’d have 28 replicas in 28 different data centers. The system could withstand nine of those data centers becoming faulty.

Kim Yong-jun could take control of the data centers and kidnap the owner’s children or something. I don’t know, and your blog will continue working without a hitch. I mean, that obviously provides an absolutely extraordinary level of security, and that's one of the reasons that we use this architecture. We need to guarantee decentralization to get the kind of security properties that we want, and that's one of the – I hate making comparisons with blockchain projects because it's really something very different.

One of the great problems with blockchain projects is that you don't really know, especially with proof of work, who's running these replicas? There's no control. Five big mining pools controlling the Bitcoin network, and that’s just not secure, and a similar thing happens with Ethereum. Sort of Genesis mining and so on. Anyways, there are unsecure P2P network. Everything goes through this sort of super nodes run by the foundation. The Internet computer is a very long way from that, and we provide not only – We’re trying to compete with the traditional stack, but we’re providing security that is better than traditional blockchain.

[00:47:06] JM: If I understand correctly, the subnets, the people who are doing the hosting on those subnets, are the randomly selected? Is that how you can ensure against –

[00:47:18] DW: No.

[00:47:19] JM: Because just to say what I’m thinking, it seems like if you only had 28 replicas, I mean, that’s much easier for somebody to kind of gain control of your network relative to something like – If we’re talking about you need to be one of these gigantic mining pools. Well, if you have all these subnets and you can log on to somebody’s subnet and take control of – I don’t know. I’m just trying to understand like –

[00:47:43] DW: I mean, in the past, I think somebody’s Bitcoin mining pools have edge towards 51%. Whenever you have such a small number of individuals, especially individuals that whose identity is maybe somewhat opaque and to have rather strange economic incentives. Yeah, sure. Look. I mean, the Chinese government could take off Bitcoin any day it liked. He just have to noble a few mining pool operators.

The security of a system, a decentralized system, is derived not only from the number of independent participants with approvable independence of those participants. The control of the independence of those participants, and that's why we lean more towards the Internet model where we have an open governance system called the network nervous system which effectively permissions blockchains. Sorry – Permissions data centers. You can apply to the network nervous system as a datacenter. You can get a data center ID.

If you're granted a data center ID through the application process, you can then make nodes which have set specifications available to the network, but the network nervous system only inducts individual nodes into the main network as it needs them. If the network nervous system decide that it needs a new sub-network, at that point, it can look at the pool of available nodes in data centers and combine them to produce a new sub-network according to the appropriate specification.

Now, there are many possible specifications. Obviously, a default specification is you have, say, 28 replicas in 28 different data centers, and that provides you with a very high-level of security. But you might want have more replicas, say, to create a video streaming, to host a video streaming canister, something like open Netflix. You might also want to concentrate, have more replicas in different regions depending upon the user profile. If 80% of your streaming – 80% of the uses of the open Netflix are at the U.S., then there's probably going to be a preponderance of U.S. data centers, which produces a lower level security than if you'd selected your data centers, select your replicas from data centers in different jurisdictions and different geographies.

[00:50:05] JM: To go back to my blog example, if I'm deploying my blog, do I have to choose which data centers my blog is getting deployed to?

[00:50:19] DW: For you as a user or a developer, the Internet computer is just the Internet computer. Currently at last, this may change in the future. You're not even able to hint where you think the users are. The network nervous system would observe usage and make certain adjustments. It might be that one day that the developer has some ability to influence this, but currently it's the network itself, the protocol itself determines where the replicas should be. But by default, it will select them for security.

[00:50:51] JM: Just to make sure I understand it right. I deploy my blog with some amount of gas. The gas is dedicated to being able to pay for the serving of requests by users.

[00:51:08] DW: Either query calls or update calls.

[00:51:09] JM: Either query calls or update calls.

[00:51:12] DW: Updates calls are much more expensive because they have to be executed across all of the replicas and actually modify the state.

[00:51:19] JM: Let's say an update happens, like a user makes a comment on my blog. Can you walk me through how the hosting providers of my blog at that point are getting paid? How much gas they get and how that gas gets transferred to them?

[00:51:38] DW: Yeah, this is another area in which it works differently to a blockchain site. There's an economic system, but it's managed by the government system. Your canister is charged with gas. It gets consumed by updating query calls. Presumably, either you're just paying for that because you've got some other kind of revenue stream or maybe you're deriving income from advertising.

The tokens, DFINITY tokens that we use to create that gas are effectively burned. The gas is burned, right? In that sense the tokens that were used to create the gas are also burnt. If there wasn't anything else going on, obviously the system would be that the tokens would deflate, but we need to remunerate data centers.

Now, remember this isn't a blockchain. It's not a cryptocurrency. There's no – We're not asking miners to speculate on the future value of the tokens. It needs to be a very stable business, otherwise we wouldn't have a stable network. The governance system determines what returns data centers should receive. Based upon the current value of the token and other considerations, it's trying to pay them some amount in real terms, okay?

Obviously, you've got some kind of economic system where users are buying tokens, usually not directly through an intermediary in order to charge their software with gas. Tokens are being issued to data centers who can choose if they want to. I mean, they can speculate it if they want to, but they can also sell them on the exchange to receive revenues. It may well be that they sell them through an intermediary too because they want to get rid of volatility risks.

[SPONSOR MESSAGE]

[00:53:41] JM: Embedded analytics is the way to add dashboards to your application. Are the dashboards that are in your application engaging your end-users or are they falling flat? According to analyst firm, Gartner, the UX of embedded analytics has a direct impact on how end-users perceive your application. Fortunately, you don't have to be a UIUX designer to build impressive dashboards and reports. Logi Analytics has come up with six steps that will transform the user experience of your embedded analytics.

Logi Analytics is the leading development platform for embedded dashboards and reports, and unlike other solutions, Logi gives you complete control to create your own unique analytics experience. Visit logianalytics.com/sedaily to access six basic principles that will transform your dashboards. That's logianalytics.com/sedaily. That L-O-G-Ianalytics.com/sedaily.

[INTERVIEW CONTINUED]

[00:54:50] JM: I'm starting to see why you don't like the comparison to a blockchain, because there are some ideas borrowed from blockchain, but one thing I'm thinking about as you're talking is like there are a lot of efforts being made in "edge computing" right now, and you see a lot more computation moving to the "edge", and it's really raising the question of what is the purpose of edge computing. People are moving storage to the edge. People have computation at the edge. Is there really anything special about these edge servers? No. They're just servers because you just got more data centers.

If you have all these data centers everywhere and they're capable both storage and computation, it's like we're attached to the idea of a CDN being the idea of an edge data center and all the CDN can do is serve you dumb images and videos and stuff. But actually it can be a fully-fledged server. If that's the case, why don't we just deploy our entire application to the edge especially since packaging those applications has gotten de minimis with things like Docker or WebAssembly.

[00:56:03] DW: Yeah. My view is the whole thing is messed up. Look. Back in the day when I was a kid, I remember getting a computer with 128 kilobytes of RAM and being blown away. How could I have a computer with 128 kilobytes of RAM? It was Sinclair or such spectrum. But now the latest 16 inch MacBook has 30 gig of RAM. The amount of RAM on this personal computer has increased 25,000 times.

I remember having a 9,600 board modem and thinking I was the boss when I've got my 14.4 K modem. The hardware landscape and the network landscape has changed unimaginably in the last decades, yet the software stack that we use really evolved organically from these very humble beginnings. No one's ever asked the question. Have we gone down the sort of wrong evolution we branched?

I mean, look, think about this in extremis. Imagine that we could fit all of humanity's data into a storage device the size of a Coke can. Clearly, the concern would be how to replicate humanity's data across lots of Coke cans and to introduce security that would ensure that modifications to the data and replications were properly propagated, right?

A lot of the systems that are used today are designed to minimize the hardware footprint, and it's true, the Internet computer – We use a lot of advanced computer science, math cryptography, everything. Of course, we also replicate computation and data. But hardware isn't where the costs are today. The costs are in the impact of being a captive customer of all these vendors in the legacy stack. The costs are in being unable to make your system secure.

I mean, look at Equifax when it got hacked. It lost 2.5 billion on its market cap and they had to pay 700 million of something in compensation to their users, and that's just the start, because hackers are moving to ransomware now. They encrypted all of the city of Baltimore servers and they're realizing they can get far more money by ransomwaring, encrypting all your servers with ransomware and getting the Bitcoin than they can selling uses data on the dark net because it's been done so many times. That's a cost.

Also, if you look at like say typical fortune 500 company, 85% of the costs are IT operations, people, even with all the licensing and waste, underutilized hardware and the rest of it. Still, 85% of the costs are IT operations.

Then if you would look into that, you'd see that like 90% plus of that cost is related to the complexity of this legacy stack the way we have to assemble – Form systems from assemblies of all these complex components like, "Well, I'm going to choose my database vendor. I'm going to choose my software stack. I'm going to choose my middle line [inaudible 00:59:06], all these stuff. You made this kind of Rube Goldberg machine that is completely unstable and then there's people tinkering with it like Chernobyl. Because you can't possibly make that thing secure, you sort of surround it with a firewall, right? That still doesn't work because if you haven't patched one of those bits of software inside the firewall, people can jump over it and it's a constant battle.

If you're a CEO, you know that just one person in your IT administration team or security team can make a single configuration error and the hacker is going to jump over the firewall. It's a constant race against time. The business model of this thing is complexities. If you're a CIO and you're having sleepless nights, you're going to talk to Palo Alto networks and they'll say, "Look. We'll sell you a new firewall, and this new firewall has got big data and AI built-in, and this AI is going to watch your network for anomalies and send out the white blood cells to neutralize the attack when it's detected." I mean, it's all bullshit, right?

Most people working in IT have no sense of the fact they're just meeting this gigantic sausage machine where you've got a system that's evolved and the design is no longer cognizant, coherent given today's hardware and network landscape. Moreover, that this whole sort of stack has also been – The organic evolution has also being driven by the commercial incentives of the vendors to create captive customers. I mean, whole thing's a mess.

That's why not here in Silicon Valley, but if you'd go into the bowels of a lot of corporations and you talk to the IT guys, oftentimes they're pretty unhappy because all they want to do is a good job and create systems that provide a good user experience and deliver the systems they've been asked to build on time or make the modifications required quickly. They just can't do it. They're just constantly battling this complexity, and it makes a lot of people miserable.

Obviously, if you're in security, you're super miserable. Securities got the highest turnover in IT because you realize that you're in a losing wicket. You're not going to be able to make these systems secure no matter what you do and you’re going to get the blame for it even if it's not your fault.

[01:01:22] JM: Very cool and impressive, and we’re almost out of time and I want to let you get back to building the Internet computer. Last question, and I’d love to do more shows on this subject. Maybe if one of your engineers want to come on the show or something. Can you give me a sense of the timeline? I know there's a lot of technical bottlenecks and a lot of moving parts, but just – I don't know. Give me a couple minutes of pontificating on what is really hard that you're working on right now and what your estimation of the timeline is with any kind of caveats you want to add.

[01:01:56] DW: Well, we’ve been battling our way through lots of challenges. Obviously, building it is one challenge. The good news there is that the design is stable and we’re on the final build branches that will actually make it into production with the MVP. That's why we started pushing stuff into the public domain, and we pushed Motoko and the SDK out in November. It was early alpha. There’re been 10 releases since then.

The reason we push that out is because that's going to make it into the – We call it a production system, the first publicly available Internet computer release and was sort of progressing with releasing other things and you know eventually before it goes live, we’re going to dump all of the technical documentation. There’ll be thousands of pages of it, and we’re hoping to make the Internet computer available to the public sort of Q3, Q4.

[01:02:50] JM: This year.

[01:02:51] DW: This year. Yeah. But it’s taken a long time. There’s been a lot of work involved, and it’s not just the sort of theory work and the software implementation work. We’ve also had to build an organization capable of implementing such a complex thing and supporting it. If the world is going to have faith in the Internet computer, they’re going to need to know that those are pretty strong ecosystem of organizations and a lot of scientific and engineering talent behind it.

We've been building that out and that in itself actually is a challenge. I mean, a lot of my time is spending theory and engineering work, but probably an even greater amount of time is being spent scaling out the org. We have a number of sort of like world famous people working on the project now.

[01:03:38] JM: I noticed.

[01:03:39] DW: Yeah. Everyone's inspired by that, it's because it's 50 years since ARPANET first allowed two computers to communicate, sort of end of 1968, 37 years since sort of 400 computers on ARPANET upgraded their protocols to TCP/IP and formed the Internet on the 1st of January 1983. [inaudible 01:04:05] auspicious for this set of next evolution. The world really needs this, right? The legacy stack is absolutely broken. It's awful, and people need a better way of building systems and internet services.

With respect to Internet services, the world also needs a way to break this big tech nightmare. When the Internet first sort of entered mainstream use in the 1990s is sort of triggered this explosion of innovation and creativity because it was an open permissionless system, right? The problem is now that big tech is managed a sort of crawl user relationships in data. If you want to build a new system, really, you have to build on their APIs. That's building on sand as Facebook showed first famously when Zynga built all these social games and its APIs, and IPO, and Facebook changed the rules, changed the APIs. A few months later it lost 85% of its value even though it's a public company.

A few years later, LinkedIn had thousands of startups drawing professional profiles through its sort of programmable web APIs and then it just pretty much revoked everyone's APIs and wrecked everybody. Now if you go to a VC and say, "Look, I want to build this really interesting new service but it's going have to build on top of these APIs provided by a big tech", they'll laugh at you because of the platform risk, right?

This has meant that the Internet has lost as dynamism. It's really sad. The secret of Silicon Valley has always being that really it's about building monopolies. You try and leverage, be super smart and aggressive and leverage first mover advantage and capital you can get from the VCs to corner a market. That could be anything from LinkedIn, through Salesforce, through WhatsApp, through pretty much any – Airbnb, anything you can't imagine.

Then if you can establish monopoly and corner the market, the network effects of having all of that user data and all of those user relationships inside your system [inaudible 01:06:09] money, pretty much not forever, but for a very long time. I mean, look at eBay, right? I mean, no one would say that eBay is the most fantastic innovative company, but they had first mover advantage with their online auction. They got the network effects and only now Amazon's beginning to take chunks out of them.

The bigger monopoly is – The mega monopoly is beginning to swallow the smaller ones. Now, there is no way of competing with that using the legacy stack. If you're a young entrepreneur or a developer today, you're locked out of this world, right? We've gone from – it's almost like we've gone back. In the mid-1990s, we had this battle between the open Internet and closed systems. Microsoft wanted to create the information superhighway. We had AOL. We had CompuServe, and the Internet 1 because it was open and permissionless.

But today, big tech has crawled all these user relationships and all these data and sort of taken us back to this hideous sort of walled garden model where the whole Internet is controlled by a few massive corporations. There is no way using the legacy stack that you can compete with this. There's no way you can build a system that uses Facebook's APIs and builds on Amazon. You're going to take this on.

The Internet computer can allow you to take on big tech and to create new systems that can succeed and it does this by providing completely new properties you can lean on. You can build open Internet services that are autonomous. They're updated and controlled by an open governance system that provides certain guarantees to their users.

But most importantly, they have better network effects. Network effects are what enable you to win in tech. If you create an open Internet service, you can guarantee APIs. You can crate, for example, an open version of LinkedIn that guarantees APIs. That provides an API where someone can put in their username and get back their professional profile associated with that username. You can guarantee that that API will never disappear and it will never be degraded and it won't just start returning the professional profile of Joe blogs, whatever username you put in. It will always work.

That means that other people can build on top of your APIs and extend the functionality. You get kind of mutualized network effects much greater dynamism. That's how we want to sort of bust apart this big tech nightmare and sort of return on the Internet to its more open, dynamic, creative, innovative roots.

We're doing this demo of the – Is a milestone called Bronze in Davos in Switzerland two weeks, and it was part of the network demonstration we're showing an open kind of professional profile network that I think a couple of engineers have cooked up in a couple of, that's not quite true, three weeks and we'll be demonstrating this. People forget, it's all part of the mythology of the Valley like, "Oh! Those engineers behind LinkedIn is a work of genius," and Reid Hoffman writing the Blitzscaling book and so on. Look. It's nothing to do with the engineering. It's to do with capturing these relationships in data and then securing the network effects. Most talented high schoolers could knock up a LinkedIn themselves in a couple of months.

[01:09:21] JM: And they crawled your address book.

[01:09:23] DW: Yeah. Exactly. We want to enable a different kind of Internet where it's not all about finding a way to sort of hijack user data relationships through first mover advantage and then sort of holding it securely so nobody can compete with you. We want to create an Internet where the currency, if you like, is the dynamism and your ability to collaborate and build out with other developers and thereby achieving far, far greater network effects and producing in the end much, much more interesting services and a much, much more interesting ecosystem for people to use. The Internet computer is much more than an alternative stack. It's also a vision for an alternative world.

[01:10:08] JM: Don Williams, thanks for coming on the show.

[01:10:10] DW: Thanks very much for having me.

[END OF INTERVIEW]

[01:10:20] JM: As businesses become more integrated with their software than ever before, it has become possible to understand the business more clearly through monitoring, logging and advanced data visibility. Sumo Logic is a continuous intelligence platform that builds tools for operations, security and cloud native infrastructure. The company has studied thousands of businesses to get an understanding of modern continuous intelligence and then compile that information into the continuous intelligence report, which is available at softwareengineeringdaily.com/sumologic.

The Sumo Logic continuous intelligence report contains statistics about the modern world of infrastructure. Here are some statistics I found particularly useful; 64% of the businesses in the survey were entirely on Amazon Web Services, which was vastly more than any other cloud provider, or multi-cloud, or on-prem deployment. That's a lot of infrastructure on AWS.

Another factoid I found was that a typical enterprise uses 15 AWS services, and one in three enterprises uses AWS Lambda. It appears serverless is catching on. There are lots of other fascinating statistics in the continuous intelligence report, including information on database adaption, Kubernetes and web server popularity.

Go to softwareengineeringdaily.com/sumologic and download the continuous intelligence report today. Thank you to Sumo Logic for being a sponsor of Software Engineering Daily.

[END]