EPISODE 437

[INTRODUCTION]

[0:00:00.6] JM: A Blockchain is a data structure that provides decentralized peer-to-peer data distribution. Bitcoin is the most well-known Blockchain, but in the next decade we will see many more Blockchains. Most listeners probably know that you could just fork the code of Bitcoin to start your own Blockchain, but wouldn't it be nice to know how to build a Blockchain from scratch?

Daniel van Flymen is the author of the Medium article, *Learn Blockchains by Building One*. In this post, he walks you through to how define the code for a Blockchain, just like any other web app. He starts with raw Python code and defines the data structures, and then he stands up his simple blockchain app on a web server to give a toy example for how nodes in a Blockchain communicate.

For me, this was a great article to read; *Learn Blockchains by Building One.* I've reported on Blockchains for over a year, but I have not seen such a clear example with this executable, simplified code that Daniel has in this post. So I really recommend checking it out. The link is in the show notes.

To find all of our coverage of cryptocurrencies, you can download the Software Engineering Daily app for iOS and Android. You can hear all of our old episodes. They're easily organized by category, and so as you listen the Software Engineering Daily app will get smarter and I recommend you content based on the episodes that you're hearing, the categories, the latent signals you're giving off as you listen to an episode.

If you don't like this episode, you will easily be able to find something more interesting by using our recommendation system that's custom-built for Software Engineering Daily. The mobile apps are open-sourced at GitHub.com/softwareengineeringdaily along with several other projects. We're building a new way to consume software engineering content. We would love to get your help if you're interested. If you're looking for an open-sourced project to hack on, we've got a recommendation system, we've got the web frontend. There's many more projects coming soon and the Slack channel is hoppin' with people who are contributing. So we'd love to get your contribution.

You can check it out at GitHub.com/softwareengineeringdaily. You can join our Slack channel, there is a link on our website on SoftwareEngineeringDaily.com. And you can send me an e-mail jeff@softwareengineeringdaily.com. I'm looking forward to your contribution, so we can move off of our Legacy Wordpress website as soon as possible. I hope you enjoy this episode. It was a great one with Daniel van Flymen.

[SPONSOR MESSAGE]

[0:02:39.6] JM: Spring Framework gives developers an environment for building Cloud-native projects. On December 4th through 7th, SpringOne Platform is coming to San Francisco. SpringOne Platform is a conference where developers congregate to explore the latest technologies in the spring ecosystem and beyond. Speakers at SpringOne Platform include Eric Brewer, who created the CAP theorem, Vaugn Vernon who writes extensively about Domain Driven Design, and many thought leaders in the Spring ecosystem.

SpringOne Platform is the premier conference for those who build, deploy and run Cloud-native software. Software Engineering Daily listeners can sign up with the discount code 'SE Daily 100' and receive a $100 off of a SpringOne platform conference pass, while also supporting Software Engineering Daily. I will also be at SpringOne reporting on developments in the Cloud-native ecosystem. I would love to see you there and have a discussion with you.

Join me December 4th through 7th at the SpringOne Platform conference and use discount code 'SE Daily 100' for a $100 off of your conference pass. That's S-E Daily 100, all one word for the promo code. Thanks to Pivotal for organizing SpringOne Platform and for sponsoring software engineering daily.

[INTERVIEW]

[0:04:10.4] JM: Daniel van Flymen is the author of *Learn Blockchains by Building One,* which is a blog post on Medium. Daniel, welcome to Software Engineering Daily.

[0:04:19.0] DVF: Hey, Jeff. Thank you so much for having me.

[0:04:21.5] JM: Well, thanks for coming on. I enjoyed reading your article. It goes into detail about how to build a Blockchain from first principles. It's a breakdown of the data structure of a Blockchain, much like in a software engineering class you learn object-oriented programming, you learn about how to build a stack or queue, or the other fundamental data structures.

In this article, you break down the private variables and the static variables and essentially how you build the object of a Blockchain and the you describe how to use it, how to build it into an API, how to basically build applications on top of it from a very raw standpoint. I want to cover a lot of that material today. Let's start with just a simple question that I've asked a number of times on the show, which is what is a Blockchain?

[0:05:20.1] DVF: So basically, a Blockchain is actually a very simple data structure. You could imagine it as a ledger or a spreadsheet, if you will, where each row in your spreadsheets contains within itself a reference to the row before it. To get a bit technical, that reference is actually a hash of the previous row. I'm not sure how deep down we're going to go, but basically if you know what a hash is, what this means is if each row has a reference to the row before it with a hash, then any change further in the beginning of the Blockchain is going to mean that all the rows preceding it are wrong. By wrong, I mean they'll contain different hashes. In its simplest sense, that's what a Blockchain is.

[0:06:09.6] JM: So we could put anything into this spreadsheet. We could put in financial data, or data about maybe the number of cars that are on the road for some kind of ride-sharing service. We could put all kinds of things into a Blockchain. The records that you put into a Blockchain go into a block. Describe what a block is and how a block is formed.

[0:06:37.8] DVF: So a block is we could think of it as a collection of records. As you just said, we could put anything into a block. We could record a list of transactions, maybe I'm sending some money to you, we record that transaction and put it in a block, we could store data in it, we could store – in the case of certain ICOs, we could store encrypted files. Basically, anything you can put in a block.

The way that blocks are formed is once data is inside the block, the block is then – a hash is calculated on that block. That hash is then stored in the next block, which will contain more collections of things. In Bitcoin and in Ethereum, these blocks are created through the process of mining where a cryptographic puzzle has to be solved in order to prove that you can build a new block, if you will.

[0:07:38.6] JM: As you've been suggested, blocks are chained together with hashes. Describe how a hash is used within a Blockchain. What's the importance of a hash?

[0:07:51.7] DVF: The hash function is a function which takes in – We could think of it, let's be very liberal here, we could think of it as any input. No matter what input you give to a hash function, you're going to get – let's just for example say that you'll always get a 20-digit random string out of your has. I could put in the name Daniel, I'll get 20 random characters out of that hash. If I change any sort of letter – if I pluralize my name and say "Daniels", I'll get an entirely different hash. Basically, these records that go into a block go through the same hash function. In the case of Bitcoin, it's a algorithm called SHA256, and they output that same – in this case, it's a 256 bit strength.

[0:08:41.5] JM: We'll get into a little more details about how these different aspects of Blockchain work. Let's motivate the conversation a little bit. Why would somebody want to know how to build a Blockchain? Or furthermore, why would somebody want to build their own Blockchain?

[0:09:01.5] DVF: I mean, that depends very much on their particular application of it. I think the interest in it at the moment is just about creating any mutable record of data, which is – has a lot of applications. One of which is currency. You could store historical transactions you might want to have bills of lading. If you are in distribution or mining you might want to record certain delivery notes. There are a lot of applications, but they all boil down to having this immutable historical data record, which is super useful.

[0:09:43.9] JM: Right, okay. So a Blockchain is a data structure, like we've said. What are the properties, what are the private variables of a Blockchain, if we're thinking about in the object-oriented programming perspective?

[0:09:58.2] DVF: Well, I go through a little bit of that in my – sorry, in my article. This depends very much on what kind of implementation you're using. In the case of Ethereum or Bitcoin, you have a very different data structure for a block. But essentially, it boils down to what a block looks like. In my example, I try and make it as simple as possible by having a timestamp, which records the Unix time upon which the block was created, a previous hash referring to the hash of the block before it.

In my example, I'm using a list data structure of data, of records to go into the block. The proof which we can all probably get into in a few minutes, which is the result of mining, and the index of the block which in Bitcoin we call it height, but that's the order incrementing ID from the beginning of the Blockchain.

[0:11:04.7] JM: Every block in this Blockchain has its own properties as well. What are the properties of a block?

[0:11:12.3] DVF: Sorry, I think I just described the properties of a block.

[0:11:15.5] JM: Okay. Sorry.

[0:11:16.7] DVF: A Blockchain itself would be, in my example it's simply a list. It's a list of blocks. That list is linked together by the hashes of the previous blocks.

[0:11:30.6] JM: Right. Okay. All right, I'm sorry. Yes. So does every new block represent a single new transaction, or does a block in the Blockchain aggregate multiple transactions together?

[0:11:46.5] DVF: Again, that depends on the implementation of which Blockchain we're talking about. In the case of Bitcoin, historically I believe the block size has been 1 megabyte. Basically, whatever you could fit into that space is considered your list of transactions. In my example, we don't have a max block size. I'm just leaving it open-ended for now, so you could insert as many transactions as you want into a block, which is not the case in the real world, but the limit is basically what the protocol agrees upon.

[0:12:23.9] JM: Okay. I got a little confused a moment ago. The private variables of a Blockchain, or I remember I was reading your article, you've got some arrays. There's an array, at least in your class outline of this data structure. You've got an array that is the chain and you've got another array that is the current transactions. Can you just describe what these two arrays do? Where are they arrays of? Are these arrays of blocks, or transactions, or what exactly?

[0:12:52.4] DVF: Okay. In my example, we have this – the current transactions is a list of transactions that are outstanding that are yet to be included into the next mined block. So perhaps, it might be a bit confusing to see that in the Blockchain plus, but current transactions will be cleared when the next Blockchain is created.

For example, we have a list which is the Blockchain itself called ‘Chain’ and that’s going to be a list of blocks. Then we have another list, which is current transactions, which is just a holding area for new transactions coming in that our node sees, that are going to be included into the next block when it gets mined.

[0:13:41.6] JM: Right. This is something that I think we’ll get into a little bit more when we hop out of the higher level, but I guess to give people a little more context on what you’re talking about, you’ve got these different nodes in the network and they are racing to validate the current transactions, and that’s this current transactions array that you’re referring to. Different nodes are going to have different perspectives for what current transactions actually means, because it’s a gradual process when the transactions propagate throughout the network.

In any case, whatever set of transactions the node we’re talking about is looking at, it can look at those current transactions and try to mine against them. Basically, in the mine process, which we’ll get into a little more detail later, you’re looking at this set of current transactions, you’re viewing it through a mathematical property, like you’re hashing the current transactions, I think – correct me if I’m wrong, you’re hashing the current transactions and then you’re looking at the previous block in the Blockchain and trying to find a number that connects those two hashes in a distinct way. Am I describing things correctly?

[0:15:07.8] DVF: Yeah. I think there’s a little bit of ambiguity there. This is actually what we talk about when we talk about mining. What we’re actually doing when we talk about mining is we are solving a cryptographic puzzle. We can get into this now if you like, but this puzzle must be very easy to verify. It must be very easy for me to verify that you as a miner have found the solution to the puzzle. But that solution must be very difficult to calculate.

In this case, in my example, what we’re doing is we’re taking – we’re working on a very, very simple algorithm, which is we are concatenating the previous block’s index and some new number, and we’re determining if the hash of those two things has four leading zeros at the beginning of it. If we can find that number – I do an example in my article. If you can find that number, then basically we have proven that we found a solution to this puzzle.

Then the network basically agrees that we can – that it can accept this new block, because you as a miner have found a solution. You found the proof, which the algorithm is called the proof of work algorithm, which proves that you've done the work.

[0:16:28.6] JM: I think we'll get into that a little bit more later. Just covering again the components of a block, so a block has a list of transactions, it's got a timestamp, it's got a proof and it's got the hash of the previous block. I think we've defined – Did you want to say something?

[0:16:51.4] DVF: No, no. Sorry, there's also the index.

[0:16:54.5] JM: The index. Okay.

[0:16:55.5] DVF: Yes.

[0:16:56.1] JM: Right. So I think we've defined what a block is. We've defined what a Blockchain is. That's the basics of the data structures. How does the first block within a Blockchain get instantiated?

[0:17:11.0] DVF: So the first block is called the Genesis block. In certain implementations, it's simply hard-coded. In Bitcoin, Satoshi Nakamoto made the first block that was considered the Genesis block. Anyone that became a node in the network would have that block.

[0:17:31.9] JM: Does a Genesis block represent any transactions, or is it just something just like a pointer for all the further blocks to refer back to?

[0:17:40.7] DVF: It could have transactions. In the case of Bitcoin, I believe that there was a transaction that included 50 Bitcoins that he sent to someone else.

[0:17:50.4] JM: Okay. How are subsequent new blocks created? Maybe you could just describe the end-to-end high-level process for how a new block is created after you've got the Genesis block.

[0:18:03.0] DVF: Yeah. I think this is when most of the confusion comes into play. Basically, in certain Blockchain like Bitcoin, we have the side here called ‘Proof of work,’ which I likely described earlier. And proof of work just means that we’re looking for a solution to a puzzle that’s easy to verify, but difficult to compute. What we do with that solution is we unanimously agree any node in the network that if I was mining, that Dan has found a solution to the next block containing a list of transactions. If I can prove that I have that solution, then all the nodes in the network are obliged to accept this new block as a source of truth and add it to their Genesis block, if you will, on their local node.

[0:18:55.0] JM: So how do transactions get added to a block? Can you clarify that a little more?

[0:19:00.5] DVF: If we want to talk about Bitcoin, or if we want to talk about my example, transactions are propagated through the network using some peer-to-peer protocol. If I want to make a transaction, miners will be listening for those transactions and they will choose to include them into the next block that they’re trying to mine. By trying to mind, I mean they’re searching for the solution to their puzzle, if that makes sense.

[0:19:27.8] JM: Yes.

[0:19:28.6] DVF: It’s luck really. This is why in the case of Bitcoin, it could take an hour or two to verify that your transaction is actually part of the Blockchain, because the network needs to reach consensus. By that, I mean, the new block has to be added and that block might not contain your transaction.

[0:19:51.6] JM: Transactions that get added to a Blockchain, they eventually need to be approved by everyone. How does that happen? Because if one person verifies a chain of transactions, that’s the chain of transaction that that node happens to have. How do these end up be getting accepted by everybody?

[0:20:12.8] JM: Right. One thing that I likely talk about in my article that I don’t implement, I’ll go into, is a transaction verification mechanism. In the case of Bitcoin, these are called UTXOs, Unspent Transaction Outputs. Basically, each node has to have some way as you said of knowing that I actually had Satoshi’s Bitcoins or whatever you want that I actually had that currency to begin with.

In my example, we don't do this. I've mentioned that I'm going to implement something in step two. The reason why I don't go into it is because it didn't seem very necessary to explain what a Blockchain was. To answer your question, we would have to implement some sort of transaction verification algorithm, which might traverse a tree of transactions and look at all my historical transactions and decide whether or not that transaction should be valid.

[SPONSOR MESSAGE]

[INTERVIEW CONTINUED]

[0:22:57.4] JM: Now before we get into your description of the nodes as servers and the nodes interacting with each other, let's review the transaction process. So let's say I've got a Bitcoin node that stood up on my computer and people can use my node to submit transactions between their wallet and somebody else's wallet.

They submit a transaction to me, maybe I get a chain of transactions and I'm also going to – I'm a miner, so my process of mining is I'm going to aggregate those transactions that I have seen that I have received from other nodes or from directly people submitting transactions to me. I am going to mine, which is I'm taking those transactions, I'm hashing them to sort of unique number, and I'm comparing that to the hash of the previous block and I'm trying to find – Sorry, I may be getting this wrong, but I'm trying to find a number –

The process of solving the puzzle is finding a number where if I take that number and I think hash it with the current transaction, it gives me some number that has – oh, gosh I'm getting all [inaudible 0:24:29.0]. A number of trailing zeros that matches with the previous block, just won't refresh us on the – This is proof of work, a described proof of work. You give such an elegant and simple example of proof of work and I'm butchering it, so go ahead and describe it for us.

[0:24:45.4] DVF: Yeah. I mean, I've implemented a very simplified proof of work algorithm. That's actually in reality not too different to Bitcoin's one. Just to describe it very, very simple, and this is probably – this wouldn't stand up in production, but essentially what we're doing by proof of work is we might take the hash of the previous block and we might find some new number to concatenate that hash with.

Now remember that the hash of the previous block is something that cannot be changed. It's a constant. It's not going away. So we take that hash, and now we start looking for a new number. We put it in an endless for loop and we start incrementing one, two, three, four and we concatenate this new number to the end of the previous hash, and then we hash this result.

We has the concatenation of some new number that we're looking for, the previous hash, and we see if this new hash has four leading zeros. The number of leading zeros might determine the difficulty of mining. You'll find that if you try this out, the addition of a new zero will make a gigantic difference in the time it takes to find this new solution. The new solution is obviously this number that we're searching for.

Let's say that we have the hash of the previous block for simplicity sake, it says ABC, and now we concatenate that with a number five. Then we try and hash them and we see if that new hash that four leading zeros. If it does have four leading zeros, then that's the proof, five becomes the proof of the current block which we're mining and that gets propagated to the rest of the network and all nodes can see, because it's very easy to verify remember? It's very easy to verify that the hash begins with four zeros when concatenated with five.

Obviously, I'm using a very, very simple example just for an explanation sake. But essentially, that is what proof of work is. When you hear about things like ASIC miners in the case of Bitcoin, these are hardware implementations of hashing algorithms that are now incredibly fast and allow you to hash a really, really big data set of possible solutions.

[0:27:06.3] JM: The feature that we're looking for with a proof of work problem is that it is difficult to find the solution, but it is easy to verify. Just to dive deeper, and an example where there is only one node on this Blockchain network that we're building. The proof of work doesn't really make sense. It doesn't really matter, because let's say we had a single node that's essentially like a bank. People can just submit transactions to it. The bank is the single node on the ledger, and the bank just takes care of everything.

The beauty of Bitcoin is that you have multiple nodes, and those different nodes can accep different streams of transactions. You can have competing streams of transactions as you're going to have different views of the present, but first of all the longest chain will win in any given scenario, but the longest chain also has to – you have to solve a problem – I mean, each of these – You might have nodes with different views of the world and they're trying to solve different problems. Whoever solves the problem first and manages to propagate that solution to the network the fastest is going to have the view on the world that will stand. Am I describing things correctly?

[0:28:30.1] DVF: Yes. Exactly correctly. At any given time though, there might be multiple sources of truth as nodes claiming to have the longest line, longest chain. This is why usually when you submit a transaction in Bitcoin, it's always safe to wait for a few new blocks to be mined just to be sure that your transaction made it into the Blockchain.

[0:28:53.5] JM: Interesting. So that's actual practical concern for people who are doing stuff with the production Blockchain.

[0:29:00.1] DVF: There are ways and means of verifying transactions, but essentially yeah, that's one of the reasons why it takes so long to transfer – sorry, to transact.

[0:29:10.5] JM: Because you have to wait for –

[0:29:14.2] DVF: You have to wait for consensus, which is to say that we have these – we have multiple chains going – each claims be the source of truth. Let's see which one, the network as a whole reaches consensus to.

[0:29:29.4] JM: I see.

[0:29:29.9] DVF: This is why it's called a race, because we have lots of miners all racing for these solutions and they're not unique solutions.

[0:29:41.1] JM: Okay. You sound somebody that I could ask some questions about the scaling debates, but let's stick to the basics a little bit more. After we have set up these data structures and the basic methods, are proof of work methods for building a single node of a Blockchain, we're going to want to stand it up as an API that can be used to request between nodes. I should just mention right now, people should really check out this article.

If you're looking to learn the nitty-gritty details of how to build a Blockchain, I found this to be such an elegant article. You've got code. You've got code that will run. It's just Python code for how to build a simple Blockchai, it's just really unique and I'll put this in the show notes. I encourage people to check it out, but what are the methods that the Blockchain API needs to be able to service?

[0:30:39.2] DVF: If we're talking about actually let's say that we're having a semi production environment, the most important thing that we would have to implement would be this idea of some gossip network. In other words, we need some way of having peer discovery. This might be agreeing that all nodes that's been up, has been up all listed on the same port. In the case of Bitcoin, it's a triple three, let's just say that they all agree that we listen on the same ports. In my example, it's 5,000.

Then maybe what your node could do is start trying to hit IPs on port 5,000, see if anything responds within actual request to tell Dan who is looking for a new node, to tell Dan that I am in fact another node. Then what I can do is I can start aggregating as a node a list of other known nodes that I know about, and if a new node reaches me I can then broadcast this new list. This is the basis of a peer-to-peer network.

Once I have this knowledge of other nodes on the network, when I start receiving transactions, maybe I need an endpoint called receive transactions. I can start broadcasting those transactions to other nodes that I know about in an agreed upon way. These are the main endpoints that you need. If you need an endpoint to do some peer discovery that would dump a list of known nodes on the network, I would need one to broadcast transactions. Then in my example, we talk about having an endpoint for mining, which isn't necessary. This is just for example sake, but if you were a node, if you were a mining node, I would imagine that you would always be mining, and that node would look pretty different to a semi-node or a full node on the network.

[0:32:39.3] JM: To initiate a transaction on the Blockchain, a user sends a transaction request to the transaction endpoint on our Blockchain server. If we're talking about the global Blockchain with a bunch of different nodes, the user could just send it to any node seemingly, and the idea is it should propagate. That's the idea of a peer-to-peer network, where there is consensus involved. What happens when a user sends that transaction to one of these random nodes?

[0:33:16.8] DVF: This depends on the protocol being used. But essentially, that node should be responsible for propagating that transaction, so that mining nodes in the network would be able to see it, and start including that transaction in the next block that they're currently mining.

[0:33:33.4] JM: Why is it within the interest of those nodes to propagate the transaction, when you want to just grab the transaction and then start mining immediately?

[0:33:41.6] DVF: So we can probably get into this, but in the case of Bitcoin, there are transaction fees. So someone submitting a transaction might advertise a transaction fee. In which case, it would be in the miner's best interest to include that transaction in the next block. But essentially, this is the protocol that we have to agree on in order to form a peer-to-peer network, that we can broadcast new transactions, new nodes can accept new transactions, and we can perform peer discover about other nodes.

[SPONSOR MESSAGE]

[0:34:21.6] JM: Heptio was founded by two creators of the Kubernetes Project, to support and advance the open Kubernetes ecosystem. Heptio unleashes the technology-driven enterprise, with products that help customers realize the full potential of Kubernetes and transform IT into a business accelerator. Heptio's products improve the overall experience and reduce the cost and complexity of running Kubernetes and related in technologies in production environments.

They build products, they build open source tools and they provide training and services and support that bring people closer to upstream Kubernetes. You could find out how Heptio can make Kubernetes easier at SoftwareEngineeringDaily.com/heptio. Joe Beda and Craig McLuckie who are the founders of Heptio have both been on Software Engineering Daily and they were fantastic when they came on. They were really fluent in what is going on in the Kubernetes ecosystem, because they helped build it.

To find out more about the company that they're building with Heptio and to find training and resources and products built for Kubernetes, go to SoftwareEngineeringDaily.com/heptio. Thanks to Heptio for being a sponsor of Software Engineering Daily. I'm really proud to have the company onboard.

[INTERVIEW CONTINUED]

[0:35:54.8] JM: You define the mining endpoint as well. You described the Blockchain API just like any other API would be described. You have public-facing endpoints, or just endpoint server endpoints and then you have the methods that are defined with – that happened when you execute that remote endpoint. We already talked about the transaction endpoint. Another one is the mining endpoint. So what would the mine endpoint on the server – when would that be called? When would we want to call that?

[0:36:36.3] DVF: So the mining endpoints is just there for illustrative purposes. As I said, you might haven't – if you're a mining node, I would imagine that you would always be mining. You wouldn't have to tell your node to mine. This is just part of my example. That endpoint is simply there to tell your node to begin mining, whatever transactions it has.

[0:36:57.4] JM: What happens during that call to mine? I know we've broken this down a couple times, but I think it's worth emphasizing once more.

[0:37:03.2] DVF: Yeah, yeah. So when you call that endpoint, your node begins – Your node looks at the list of current transactions. It then includes that in the next block, and it begins searching for a – it runs the proof of work algorithm as we described and it starts searching for a solution. If it finds that solution, it adds its new block to its block chain and it clears the list of outstanding transactions.

[0:37:31.3] JM: As we said before, the single node could theoretically service these requests. But we want to build a network with a consensus mechanism. Let's say we want to setup four other nodes within our network, so we have total nodes. What methods that we have not discussed yet, do we need to add to these nodes? How do we get them networking with each other?

[0:37:56.2] DVF: Right. Essentially, we need two new methods. The first one is to verify – is to return a node's current Blockchain from beginning to end. In my article, one of the endpoints we implement is called – let's just say Get Blockchain. What that does is that nodes, it returns its entire Blockchain for the inspection of another node.

A neighboring node might see four new nodes on the network and it might say, "Okay, everyone I need to make sure that I've got the source of truth Blockchain." Maybe it does it does these periodic intervals, but that one serving that chain, the rest of the nodes have an opportunity to discover if their chain is valid. If so, their chain longer than my chain? If it is, then I'm going to replace my current chain with this new chain that appears to be valid. This is how consensus is propagated through the network. We have competing chains, and whichever one appears to be the longest chain to me as a node, I'm going to accept as truth.

[0:39:08.8] JM: What happens –

[0:39:10.6] DVF: Sorry, go ahead.

[0:39:11.1] JM: No. You please.

[0:39:13.0] DVF: Yeah, that's the one endpoint. We'll call it '/Chain,' which would return my chain. Another endpoint might be called register, which is another – which would be an endpoint that could receive a new chain from a neighboring node in order to validate the chain.

[0:39:34.2] JM: Okay. So what happens if we want to add a sixth node here? When a new node wants to join the network, what does it have to do?

[0:39:43.5] DVF: It has to first see if there are any neighboring nodes and start accepting lists of neighboring nodes so that it can maintain some directory of other nodes on the network. In my example, I think I would call it node/register, which will accept a list of new nodes in the form of IP addresses.

[0:40:06.6] JM: Okay. Let's refresh people once more. We've talked about accepting a transaction with a single node. You receive the transaction, you add it to your list of transactions and you're mining to process the transactions and accept them. Give more of an overview of how the transaction processing differs if there are multiple nodes involved. And some of the conflicts that can happen, like when nodes end up with different chains, that you could talk in the context of your simple Blockchain example, or if you want to give some more complicated, real-world examples.

[0:40:47.3] DVF: Okay, let's talk very high-level in my simple example. Let's say that node A has – receives one transaction and its change length is 10. Node B also has a chain length of 10, but receives two new transactions. This is one of the cases where it's simply a race. Let's say that both these nodes are mining. So we have node A, that's got one new transaction, and we have node B that also has the exact same Blockchain but receives two new transactions.

Whichever node finds the solution to the proof of work algorithm first is basically the winner. Any outstanding transactions, so let's say that node A finds the solution first and node B is stuck with these transactions. In this case, we have this race. Node A is the winner, because node A's Blockchain is longer. That node can then start propagating that Blockchain to the network and that happens very much organically at periodic intervals. It reaches consensus by propagation. That's one of the cases that can happen.

In that case, the transaction will just have to be retried, the outstanding transaction on node B, which is have to be retried until some miner might include it in the next block. That's one thing that can happen. The second thing that can happen is let's say that node B has a valid chain, node A has a valid chain, but the last block for whatever reason is invalid on node B. In this case, we would talk about having a notion of an orphan block.

In which case, it's simply unfair to node B, he wasted computational power trying to find these two new blocks where the rest of the network has gone in the direction of node A. This is just a part of the course. I mean, we have to have some way of resolving them and unfortunately, it's by longest chain first and there's a good deal of luck involved too. Does that answer the question?

[0:43:03.5] JM: It does. Let's talk a little bit more about that though. How does a consensus resolution happen? How do we come to consensus during a conflicting event? Like if there are two conflicting views of the world, how do you come to consensus?

[0:43:23.9] DVF: Well, these are agreed upon rules as part of the protocol of your Blockchain. Let's stick to the example, and we'll say that the longest valid chain of blocks is authoritative. In other words, the longest chain of blocks is the source of truth. That's something that can't be argued with, because remember these proofs are very easy to verify.

That means that a new node when faced with a conflict, if I'm node A and you're node B and my Blockchain length is 10 and yours is 11, it's my duty to check that if you're propagating – if you're propagating a longer chain, it's my duty to check that it's valid. If it is valid, I'm going to replace my entire local chain with your one.

[0:44:14.7] JM: Let's bring this to the real world. Explain how different your toy example is from the Bitcoin Blockchain.

[0:44:27.5] DVF: Yeah. In theory, at a very high-level, it's not too different. Bitcoin also uses the proof of work algorithm that's actually quite similar. With Bitcoin, the difficulty of the mining process has changed that certain intervals to maintain a certain number of blocks being mined per hour. There's definitely more complication there. There's also the notion of transmitting only the heads of blocks, which we haven't got into, but the heads basically contain some meta data about the block, which means that the entire block itself in the case of Bitcoin doesn't have to be downloaded by each node.

There is a difference between my toy example in Bitcoin obviously, but I chose not to complicate it, I just wanted people to understand what a Blockchain is. Another difference is in terms of the propagation of transactions works a lot differently in Bitcoin. Of course, Bitcoin has a very well-known and published protocol for peer discover, which we don't do at all in my example.

In my example, we're just basically registering new nodes on the network manually, which is something that in a follow-up article, I'd love to change, I'd love to make it more dynamic. But in the toy example, obviously you can't get too complicated.

[0:45:57.6] JM: What you referred to with, the meta data and new nodes not necessarily need to store the entire transaction history. Can you talk about that more? Like if I stand up a new node in Bitcoin, do I need to download all the past transactions, or do I just need the head of the Merkle tree to essentially be a condensed history?

[0:46:22.6] DVF: So in Bitcoin – I don't claim to be an expert about Bitcoin, but I do believe that there are three kinds of nodes in Bitcoin. You could be a semi node, which is a way you – as you just described, you don't need to download the full Blockchain. Or you can be a full node, which is I think what – if you download Bitcoin Core, I think that's what it attempts to do. It requires about a 160 gigs of space on your machine and it starts downloading the authoritative Blockchain from the network. Then the third node of course is if you want to be a miner.

Bitcoin has many implementations of depending which client you download, they're all got to agree on the Bitcoin protocol itself. They're all going to implement the same functions, the same endpoints. It's just up to the implementation how it does that. At a very high-level, yes we have a Blockchain, no it's not nearly as complicated or in-depth as the Bitcoin Blockchain.

[0:47:31.5] JM: This idea where some nodes can have just the head of the Merkle tree, or I'm not sure if that's the right way to put it, but the condensed volume of transactions, rather than the entire transaction history. Is this a point of contention, the fact that some nodes can have – don't have that whole history?

[0:47:52.1] DVF: I believe that Bitcoin implements the SPV, which is the validation layer of transactions. As far as I understand it, if you are just validating transactions, you don't – I'm very careful saying this. I might be proven wrong. But I believe that if you only have the heads of blocks, there is a way to verify transactions without downloading the entire blocks themselves.

[0:48:22.4] JM: It seems like that will be totally possible. I just wonder at this point why people would still need to download the entire transaction history if they – why that's even an option?

[0:48:36.0] DVF: You got to remember that this is completely decentralized, and if we want to have validated consensus in the network and we want to maintain a 51% source of truth in the network, we need to have some source of authority, which is these entire blocks.

[0:48:54.6] JM: Sure. But if you've agreed that the history up to a T1 is defined as this hash, then why would you need the transactions that came before T1? Why would you need the actual transactions if you essentially have the proof of them condensed in that hash?

[0:49:24.1] DVF: I'm not a 100% sure. Though I'm hesitant to offer a guess.

[0:49:28.7] JM: No problem. No problem. We don't need to –

[0:49:30.6] DVF: One thing I can say is that I do know that the simplified payment verification of Bitcoin does have some weaknesses that needs to – that means that that need to be full nodes on the network in order for them to fully verify historical transaction.

[0:49:53.5] JM: Do you have any idea how your example compares to the Ethereum implementation of a Blockchain?

[0:50:01.1] DVF: Yeah, at a very high level, we can certainly talk about it in the cases of Ethereum. In my example, we are storing transactions. In the case of Ethereum, we actually have an opportunity to store vital code, which can be interpreted by each node and can actually be run. Ethereum if you think about it is essentially this distributed computer that allows new nodes mining to run code, which is – That code is called 'Smart Contracts,' because it's immutable code. It can do something upon validation.

[0:50:41.1] JM: Right. Given that you're somewhat embedded in this community, can you describe to me – and you're also good at describing things in a little more layman's terms, as layman as we can get in the Bitcoin world. Can you articulate the scaling debate, some of the recent scaling debates? Like I did a show recently on Segwit and I was really out of my comfort zone. I didn't really know how to approach the topic very well. I did as much preparation as I could've, but can you talk about that? What are the debates going on there?

[0:51:18.7] DVF: I wouldn't claim to be – again, I'm not an expert on this. I'm worried to get involved in a debate. But a high-level, there's a bunch of core developers of Bitcoin Core who agree very strongly in this new algorithm called Segwit. There's a bunch of other people who don't agree with it and once Segwit2x – and there's a huge debate in the community right now. I don't know. I don't want to get out of my comfort zone by getting into this.

[0:51:53.0] JM: This is such a hard topic for me to cover. I'm just continuing to beat my head against the wall, and I hope the listeners don't care, because – It's like this and deep learning and there is some other things where I just – I'm not sure if the casual podcast host that tries to be a dilatant in all kinds of different software engineering topics is the right person to explain these things. I'm not sure if I should be doing these at all.

[0:52:20.0] DVF: Well, I mean you're not wrong. This stuff is so complicated. I mean, part of the reason my article has attracted so much attention and has done so well is because this stuff is difficult to understand. I think I've done a pretty good job of simplifying some of that complication. The fact that it's so popular goes to the fact that there's so much confusion in the space. It's always going to be tricky to understand everything.

[0:52:47.7] JM: What are the other hang-ups that people have? When they talk to you, when they contact you and they say, "Hey, I love your article. I didn't get X." What are the things that people are confused about?

[0:52:57.7] DVF: Yeah. I think a very thoughtful problem that people have is they ask me, "Well, how can we know that this transaction is valid?" That's a very loaded question, because in the form of Bitcoin, you have this notion of unspent transaction outputs, so UTXOs. This is one of the problems of my example is I don't implement any way of verifying transactions.

One thing that people also don't get is how can they prove that a transaction belong to me? Well, in the real world I'll tell you that if you want to submit a transaction then you have to sign that transaction with your key, and then the rest of the network can prove that your public key is the source of truth for that transaction. We don't implement that in the example, because it's just – it's too complicated for the person who just wants to learn what a Blockchain is; to dive into and say, "Okay, well here is how public-private key cryptography works." Now you need to have this death traversal of transactions and you need to start creating a UTXO table.

It just gets more and more complicated that you've got to cut it off at some level and say, "Right, these are the atomic things that we want to explain and these are the things that you need to know. If you want to learn more, I can point you in that direction."

[0:54:25.1] JM: Talk more about some of the differences between your toy example and a production Blockchain. What are the other aspects of the infrastructure that you cannot cover, because you are just discussing the basics?

[0:54:43.5] DVF: So if we were to productionize my example – Well firstly, for any protocol to be accepted by the community at large, it has to have solid academic backing, which I don't have. The yellow paper for Ethereum or Satoshi's original paper for Blockchain, they're amazing because they took into account so many different things. They took into account block difficulty and what would happen down the line and how things would scale. As you know, that's already a giant source of contention at the moment.

These are the kind of questions that we would have to answer if we wanted to take out our small Blockchain and turn it into something that's ready for production. Unfortunately, they don't have easy answers. One of the giant contentions with Bitcoin, this whole Segwit drama that's unfolding relates to the block size of the Bitcoin block. Some people want it to be 2meg, the original limit was 1meg. Now we have to make another layer underneath Bitcoin to support this bigger block size.

These are all things that have to be – they have to reach consensus in the community at large if people are willing to play ball. Unfortunately, I'm not a professor. I don't have a team of people that are going to tell me how things are going to break down the line. If you're talking about in the interim, how would you productionize this? I would say that you would have to implement some solid peer-to-peer protocol for this new network and a very easy way of validating transactions, just at a basic level.

[0:56:26.1] JM: All right. Well, Daniel this has been great. It's been really informative. I found your article really useful. What else are you working on these days? Who are you? What do you do?

[0:56:38.3] DVF: Thank you. I'm a software engineer. I'm from South Africa. I'm sure you could tell by my accent. It's still pretty thick. I work for a company called Blink Health. In the health tech space, we make prescription drugs very cheap for Americans, for all Americans. You can Google what Blink does, but we're an exciting startup to work for. We based in Manhattan. The way that Blink works is you can buy any drug that you're being prescribed through the Blink websites and go pick it up at any pharmacy at a massive, massive discount.

[0:57:16.2] JM: Distributors or you manufacture them cheaper, or what exactly?

[0:57:21.7] DVF: Well, because of the number of patients that we have, we are able to negotiate cheaper prices for certain drugs. Yeah, I'm very interested in things that help people. I view the tech space as something that could be used for the good of everyone. Therefore, I'm drawn to solutions that make life a little bit better. One thing I've really been thinking about lately is using some Blockchain technology in order to democratize healthcare.

[0:57:53.6] JM: Like EMRs, or what exactly?

[0:57:56.9] DVF: Yeah, yeah. I mean, it struck me as pretty weird. You as a patient don't actually own your medical history. These giant EMRs, doctors keep your charts, doctors keep your history. If you were to move somewhere else in the world, you would be faced with two problems. The first problems is how do I get all my medical history out of this really conflated system? And doctors use different EMRs.

The second problem is this information isn't standardized in a format that's universal. These two problems make I think Blockchain a very compelling solution for this. For example, you might encrypt your medical history that might be stored in the Blockchain, nodes might be incentivized to store this information. That would demand that there would be some standardized way of representing healthcare information. But yeah, there are a lot of interesting applications.

[0:59:02.6] JM: Yeah, that sounds great to me. I certainly I'm hoping for that future. How far do you think we are from people being able to build that kind of stuff? It still feels like that's like – I mean, of course you and I would love to say that's going to happen in the next two years, but feels like five or 10 years at a minimum, right until we can get to that kind of stuff?

[0:59:30.8] DVF: I think that the technical challenges are probably – they pale in comparison to the bureaucratic governmental challenges. You dealing with the US healthcare industry, which is at least 10 to 15 years behind almost every other industry. It's probably the biggest challenges is regulation and dealing with government. I'd say that that's the massive thing that's going to further.

[1:00:02.1] JM: I talked to this company Oscar recently. Oscar was really interesting, because they're basically building an entire healthcare stack. They started with insurance, but they're opening up clinics and you've got places like One Medical that are opening up their own chain of clinics. You've got Front, I think is the new one in San Francisco, where they basically say, "You know what, screw the existing healthcare system. We're building full stack everything." These kinds of places I think are promising, because they could be the early adopters of this kind of stuff.

[1:00:38.0] DVF: Yeah, I agree a 100%. I think that the startup space in healthcare is one of the most interesting spaces to watch. Healthcare itself is quite interesting. It's the only industry where the cash prize doesn't fetch you the cheapest price. What I mean by that is if you try pay cash for medication or for a doctor's visit or for anything, you're going to pay the highest price possible, because you're not incentivized to pay cash. You're incentivized to find the right insurer, which is super nuts to me.

[1:01:10.1] JM: That is perverse. Well, we should do another show in the future sometime about healthcare stuff, because it sounds like that's where a lot of your passion lies. And I would love to talk about Blink Health, because the technology around prescriptions and I would love to dive into how you solve the chicken and egg and you've got enough volume coming through Blink Health to negotiate those prices down. It sounds like a really interesting conversation we can have.

[1:01:36.5] DVF: Yeah, definitely. I'm more than happy to have that conversation.

[1:01:40.0] JM: All right, Daniel. Well, thanks for coming on Software Engineering Daily. It's been a real pleasure and I enjoyed your article.

[1:01:44.9] DVF: Jeff, thank you so much for having me. I'm so happy that my article has helped people understand this complicated world that we're in. I look forward to speaking to you in the future. Have a wonderful day.

[END OF INTERVIEW]

[1:01:56.9] JM: We are partnering with Indeed Prime to give you some tips on thriving in the modern workplace. If you're looking for a new job, checkout Indeed.com/SEDaily. Now let's get to today's tip. Today's tip is about making the most of a new job.

So once you're in a new development role, what can you do to be successful on the product and working with your team? Well the first is do good work. That might sound obvious, but there are a lots of things that are not doing good work, such as just sending random e-mails, maybe making some comments on code, doing half-hearted code reviews, writing a couple of unit tests. I mean, that stuff is useful, but make sure that you're doing the core of the good work that you are hired for. If you're not fulfilling the thing that you were hired for, it's unlikely that this job is going to go well.

The next thing is to get along with the team; the other engineers, as well as the PMs, the designers, the management, all of the staff. Software engineering is a team sport and you have to figure out how to work productively with people. You cannot be the lone wolf who just gets everything done on their own and expect to excel.

There are people that do that and you will stay as a lone wolf for the rest of your career. If you want to elevate yourself and get into management roles or learn how to lead a team, learn how to get scale out of your software engineering work, you have to learn to enable other people and to work productively with other people.

Part of getting to that productivity with other people is communication. You got to communicate, communicate, communicate. You have to be open to learning and doing your best to improve that communication. There are a lot of books about how to improve interpersonal communication. I've read a lot of those books. I think they've helped me improve a lot. I still have a long way to go.

One thing that can help with communicating is actually listening which sounds like a lack of communication, but actually that is – it's the keystone of communication is listening, and digesting what other people are saying.

If you can learn to listen, then people will love working with you. If you're just somebody where – anybody can come to you and they can tell you what's on their mind, and you can repeat that back to them in your own words. That's the definition of listening and you will be a valued member of your team, if you're great at listening and communicating.

Another tip is to be reliable and be helpful to others. You need to be reliable, because if somebody asks you to do something or if you say that you're going to do something and then you don't do it, it's almost like the person who you're interacting with has tried to execute a line of code and that code just refused to execute.

When somebody is not being compliant with the things that they say they're going to do, it makes it really hard to move productively as a team, because then you lose trust. If somebody says – if you say you're going to do something and then you don't do it, your teammates are going to start to lose trust in you and that's going to break down the speed in which an organization can move.

As part of the ability to move fast is I can say to my friend who is on my team, "Hey, can you go and do X?" Then I can expect a callback in a couple days that, "Hey, I did X and this is what happened." If you can't get that callback, then it's going to be problematic.

Another way to make the most of a new job is to leave room for innovation and new ideas, and share those ideas within the company, because oftentimes the fresh eyes of a new hire can really bring new life into a company. So don't be afraid to share your new ideas with the company, as long as you're doing your core work, that's why the first thing I said was to do the good work, as long as you're doing that work, people will love to hear your new ideas.

Another tip is don't overwork. You need to find your balance. It's easy at a new job to just be drinking from the fire hose, taking on all kinds of new projects, but you don't want to burn yourself out. You want to find the balance and find a sustainable amount of work that you're going to do.

Another tip is to ask for feedback and make improvements where you can. Everybody has flaws, and the people who thrive in the workplace are the ones who are constantly asking for feedback and knowing where they can improve. The final tip I'll give is take on additional responsibilities when you can, to help your team and to learn new things. Taking on new responsibilities will stretch you to your limit and will help you grow as a person and will help the organization grow.

That's the end of these tips. Thanks to Indeed Prime for sponsoring these tips. You can go to Indeed.com/SEDaily to find out more about how Indeed Prime can help you find a job. It's a great tool for finding jobs at places like Facebook or Squarespace or tons of other great engineering organizations that would love to have you as a great engineer.

Thanks to Indeed and thank you to the listeners who stuck around for the end of this episode.

[END]