MITOCW | 8. Forks

 NEHA NARULA: Welcome back.

Today, we are going to talk about forks.

So I think you guys have had a guest lecture last week from Alin Tomescu, who talked about his project, Catena.

And one of the really interesting things about Catena is that he's trying to prevent a server from equivocating.

So he's trying to keep a server from being able to say one answer to some people and another answer to another person.

That's kind of the goal of this whole blockchain thing.

But there's actually something really complex that happens when we start to change what's happening in the network, and we start to change the software that's happening in the network.

And so this lecture is going to be about forks.

And I want you guys to feel free to raise your hands and ask questions during class, because this stuff is actually kind of non-intuitive sometimes.

So a blockchain is a chain of blocks which have previous hash pointers inside of them.

So a question-- can a block point to two previous blocks at the same time?

Anybody know the answer to this question?

No, exactly, that is the whole point.

There is exactly one spot in the block header for a previous hash block.

And so a block can only point to one previous block.

A single block cannot point to two previous blocks.

However, can this happen?

Can two different blocks point to the same previous block header?

Yes, yes, and this happens all the time.

Now, part of the reason that this happens all the time is because the way that blocks are found in the bitcoin network is completely probabilistic.

There are many people who are trying to find the next block at the same time.

And it's entirely possible that two different miners will get lucky in a time period that is within the time it takes for a block to get gossipped around the network.

And so, totally, two people can find the same block at the same time.

This is actually what's known as a fork.

And you can kind of see why.

We have a fork in the chain right here.

But what does it really mean when there's a fork in the chain?

I just told you that this is something that happens all the time.

And these cryptocurrency networks seem to keep running.

They seem to be keep going.

So perhaps it's not as bad as it seems, even though it seems to be violating one of the major tenets of what a blockchain is.

Well, it is kind of bad, actually, because when you have these two blocks here, is there anything that makes one the right blockchain over the other blockchain?

Just if this is one blockchain right here, and this is another blockchain right there, how is one supposed to distinguish between this?

Yeah?

AUDIENCE: The one with the most work.

NEHA NARULA: The one with the most work, OK, that's a great sort of way of thinking about it.

What if they have similar amounts of proof of work.

Then what?

AUDIENCE: The one that's accepted by the majority of the network.

NEHA NARULA: The one that's accepted by the majority of the network.

How do we know which one is accepted by the majority of the network?

It gets extended, OK, great.

So that's a really good way of looking at it.

Before we go into what that means, do people understand why if this state were to persist it would be a really bad situation?

One thing that we could have, given that there's two blocks, what that really means is that we have two versions of history.

We have two versions of the ledger if these two things are not the same.

Different coins might be spent on one side of the fork, versus on the other side of the fork.

In fact, the same coin might be spent in two different people on either sides of these forks.

And so over here, Alice might be spending her coin to Bob.

Over here, Alex might be spending that same coin-- and, remember, coins can only be spent once.

Even if Alice has other coins, she specifies which coin she's spending.

Alice might spend that coin over to Carol.

And so if Bob sees this, he thinks he got paid.

If Carol sees this, she thinks she got paid.

And we violated one of the fundamental tenets of the blockchain, which is not to double spend coins.

You can't create money out of nowhere. If these two things would persist, if we don't resolve this fork somehow, then we literally have two different versions of the same currency.

This is kind of as though I took the money in my pocket, my dollar bill, and I xeroxed them all, and managed to convince some people that the Xerox was also real.

This is the exact same thing as what's happening when you have these two things persist.

OK, so how do we fix it?

Well, which is the right one?

And just given what I've shown you right here, it is not clear which is the right one.

And like I also said, this happens all the time in the bitcoin network because the bitcoin network is probabilistic.

So the way that people figure out which one is the right one is they wait.

They wait and see which side gets extended.

And by the nature of probability, one side is going to eventually win out, and get extended more often than the other side.

It's very unlikely for two forks to randomly sort of probabilistically exist at roughly the same way for the same amount of time for very long.

Now, as someone said, you might look at this.

How do you distinguish between these two things?

There is this common idea that in blockchains we take the longest chain.

Longest is a little bit misleading actually.

It's not the longest chain.

It's the heaviest work chain.

So we pick the chain that has the most proof of work on it, even if that happens to have a smaller number of blocks.

Usually that's not the case.

That's not what happens.

The reason for that-- does anyone know why we take the heaviest chain instead of the longest chain? AUDIENCE: So that nobody can cheat and lower the level of difficulty.

NEHA NARULA: Right, so it's not very easy to do this.

But over time, someone could slowly decrease difficulty level, and then create a nice, really long chain with many low difficulty blocks, and then say, hey, I've got the longest chain.

I have the most number of blocks.

Despite the fact that your chain has more proof of work on it, you should all take my chain.

And they could do that with less than a majority of the hash power.

Now, this is not easy attack to do.

But it is a possible attack.

And so that's why we take the heaviest chain, which represents the most hash power in the system, the majority of the system.

So forks happen all the time.

Forks are normal.

The way that people who see forks decide which side of the fork to use, is they check and see which one has the most proof of work, the heaviest proof of work.

Now, one thing that's kind of cool, is, as you probably learned about two classes ago, even SPV nodes, even clients, can tell which chain has the most proof of work simply by looking at the headers, the block headers.

They can see what the difficulty is.

And they can measure for themselves which side of the chain has the most proof of work.

They don't even need all of the chain.

So this is a nice property of light clients, is that they can download all of the block headers and immediately tell which side of a fork has the most proof of work.

Great, we call the block that ends up-- so what do we do with this block?

Does anyone know what actually happens with this block?

Anybody?

OK, so this book is basically abandoned.

This block is just left.

It exists.

It points into the chain.

That's great.

In bitcoin, in particular, it's meaningless.

It doesn't mean anything.

Every transaction that was in that block is not considered to have happened.

So those transactions need to be replayed on the chain that ends up being the longest chain.

And this happens, like I said, all the time.

We call that block an orphan, because it doesn't have anyone.

It's kind of the wrong way around.

I mean, I think of this as a child.

But, whatever, this block doesn't have a parent.

But we call those blocks orphans.

And, again, remember, we pick the side of the chain that ends up having the most proof of work.

Now, we draw these figures in two different ways.

Sometimes we draw the figures like this, like there's this blockchain that exists out there, that is written down, and that everyone is sort of looking at.

There's one blockchain.

But this isn't a logical representation.

In reality, there are many, many, many, many different blockchains.

In fact, every node in the system has its own version of the blockchain.

And it's what those nodes are doing together that ends up deciding what the blockchain actually is.

So if these are nodes that are running the bitcoin protocol, and when I say nodes, what's the difference between a miner and a node? AUDIENCE: Nodes verify transactions, but miners push blocks.

NEHA NARULA: Exactly, so miners actually create blocks in the system.

Miners have write access to the blockchain.

They actually produce new blocks.

People think a lot about what the miners are, and what the miners can do.

And there's a lot of game theory thinking about the miners.

And, oh, what if they collude, but the hash power does this?

And sometimes we forget that nodes in the bitcoin network are actually incredibly important too, because it's the nodes in the bitcoin network that are now creating blocks.

And, to be clear, someone could be a miner and a node at the same time.

They're distinct roles.

The same entity could certainly run a node and be a miner.

But nodes are doing something really important as well, which is they're reading all of the blocks.

And they're deciding which side of a fork to take, and which blocks to accept and which blocks not to accept.

So let's say that these are all bitcoin nodes.

And this is what the blockchain looks like right now.

And we've used this figure multiple times.

Every block has a previous hash, which acts as a pointer to the previous block.

Every block has a set of transactions.

And every block has a nonce, such that when you take the hash of the block header, you end up with something that has a sufficient number of leading zeros.

It has valid proof of work. So let's say that this is the state of the system right now.

And then, we see this new block come along.

So what happens here is that a miner has produced this block, and has sent it out over the bitcoin network.

They've started gossiping the block around the bitcoin network.

And so every one of these nodes, hopefully, will eventually, if they're well connected, will see this block.

They're going to see the block at different times.

And they're each going to make an independent decision about the block.

So every node is going to decide when it sees this block what to do with it.

And there's basically two things it can do with it.

One, it can add it to its own local version of the blockchain.

And number two, it can reject it.

Does everyone see why that is?

So sure, every node is maintaining this local copy of a blockchain.

There is no single, global copy that everyone is referring to physically.

There's no single database running the blockchain.

Every node has its own local copy of the blockchain.

Every node is making its own local decisions about whether or not a block is valid, and about whether or not it's going to add that block to its own local copy of its blockchain. So the way that the node decides this is that it has a set of validation rules.

So it has its own set of rules, which are specified by the bitcoin protocol.

And it checks that block against its set of validation rules.

So every node, every bitcoin node, is constantly validating, constantly validating blocks.

Here are just some of the validation rules.

And we've talked about a lot of these sort of off and on during the class.

So first of all, there's a max block size.

And the way that this works is if at any point you fail one of these validation rules, you reject the block.

So first, the block has to be smaller than this variable max block size.

At the moment, max block size is set to 1 megabyte.

Every single transaction in the block also has to be valid.

And what does that mean?

Well, it needs to be formatted a certain way.

And it also means that when we concatenate the script pub key and the script sig together, and then run that through the interpreter to evaluate the script on the stack, it ends up returning true.

And if at any point, any transaction-- and not just any transaction, any input in any transaction, does not satisfy this, we reject the entire block. Another thing that each transaction needs to satisfy if it's valid, is if the transaction specifies an end lock time, than that end lock time needs to be greater than whatever is specified in the transaction.

That transaction is now is now far enough in the future that it can be satisfied.

The block as a whole needs to have valid proof of work.

So it needs to have a nonce, such that when you hash the block header, you end up with a hash that has a certain number of leading zeros.

And what's really interesting is the protocol also specifies exactly how many leading zeros there need to be, depending upon how many leading zeros there have been in blocks in the past. There cannot be any double

spends.

So given the version of the blockchain that I, as a node, have, this better not be double spending any outputs.

There are some rules around what the block can have as its timestamp.

It cannot be more than-- if I see a block that has a timestamp that's more than two hours in the future from my own timestamp, I will reject it.

And if it isn't after some point that's the median of the previous block's timestamps, then I'll also reject it.

And, of course, the block needs to actually point to a block that I've seen before.

And as a node, I will keep around blocks which have a previous hash pointer that I don't know about.

But I'm not going to keep them around forever, because that's a potential Dos attack.

And in order for me to actually insert this block into my version of a blockchain, I need to know which block to put it after.

Note also, note what's not on here is that it needs to be the only block pointing to a previous block.

I'll keep around multiple blocks at the same level until I have some sense of which one is going to win out, because of the probabilistic nature of maybe this block I'm seeing is not going to end up being part of the blockchain.

Maybe it's going to be an orphan. So these validation rules are what are called consensus critical rules in the bitcoin network.

And what that means is that if you want to tweak these rules, if you want to change them, then you may end up changing which blocks nodes will accept, and which blocks nodes won't accept.

Now, something that's kind of interesting to note here is that the entire script interpreter in bitcoin is consensus critical.

Yes?

AUDIENCE: If you reduce the amount of transactions, does that reduce the size of the blocks as well?

NEHA NARULA: Yes, it can.

Yes, so you can definitely have a block-- so that's a great question actually.

So let's say that we had a block that was half a megabyte.

What would nodes do?

What do you think?

AUDIENCE: I think an advantageous miner would potentially process it faster.

NEHA NARULA: That's true. What would you as a node do.

If you're running the bitcoin protocol as it exists today, and it has the following set of rules, and I hand you a block that's half a megabyte, what would you do?

You would say OK, exactly, because the rule that I specified here just says the block has to be less than 1 megabyte.

It didn't say it has to be greater than anything.

And is half a megabyte less than 1 megabyte?

Yes, so you're going to pass the checks.

And you're going to accept the block. AUDIENCE: Is that a good thing for miners, so maybe remove the transactions from a potential block, because they want to process it faster?

NEHA NARULA: Sure, so the question is, should miners remove transactions from a block, or perhaps not put transactions in a block, because it seems like they could process a small block faster than a large block, right?

So that is very true.

Miners, if there are fewer transactions, they have fewer things to verify.

So they can check and see if the book is valid faster.

However, the fewer transactions, the fewer the transaction fees.

And so the less the miner will earn from the block. Yes?

AUDIENCE: And also, the energy put in, most of it mining, is like mining the block, not actually verifying.

NEHA NARULA: Yeah. So verifying transactions is nowhere near as CPU-intensive as it is to actually find the nonce and produce the proof of work.

However, miners should verify transactions before even starting that process.

And so if a miner didn't verify transactions, they would get a head start on the proof of work process.

So there's that sort of balance.

And, in fact, I think we've talked about this in this class before, there have been times in history where things have happened that have shown that miners actually haven't been verifying blocks.

They'll just mine on whatever.

It's caused some problems.

And they ended up losing a lot of money because they should have been verifying blocks.

Great.

So these rules are what are called consensus critical.

They are the rules that each node independently uses to decide whether or not they're going to accept the block.

Yes?

AUDIENCE: When you talk about nodes, you talk about SPDs, right?

NEHA NARULA: No.

I'm not talking about SPDs when I talk about nodes.

I'm actually talking about full nodes.

So I'm talking about nodes that are holding onto the entire bitcoin blockchain, that are looking at every block, every transaction.

AUDIENCE: So the difference between a miner and and a node are [INAUDIBLE]?

NEHA NARULA: A miner-- yeah, they're distinct roles.

So a miner is running this proof of work algorithm to try to find-- to produce blocks.

A full node does not write blocks.

It doesn't create new blocks.

It receives blocks, and then validates them, or not, and adds them to its local copy of the blockchain.

A miner might also be running a full node.

Yes?

AUDIENCE: Miners should be running full nodes.

NEHA NARULA: Yeah, they should be.

But it's not clear that that happens, for the reasons we were talking about before.

I'm sorry?

AUDIENCE: You can force miners to [INAUDIBLE] NEHA NARULA: OK, so let's go back to this image right here.

These nodes are all running some version of the software.

They see this transaction.

And they say, yes, this looks like a valid transaction to me.

And note that they have to-- or, sorry, block.

They look at this block.

Yes, this looks like a valid block to me.

And note that they kind of have to evaluate that block with respect to all of the other blocks that it's seen.

No double spends, all of the scripts eggs, the script pub keys, work out correctly, it's the right size, et cetera, et cetera.

And then once it's decided that that's all OK, each node will independently make the same decision to add that block to its blockchain.

So you can see here that determinism is actually really important.

It can't be the case that there's any randomness in this process.

It can't be the case that sort of randomly, depending on the time of day, one node will decide this is a valid block, one node will decide it's not a valid block.

That's not the greatest situation to have happen.

We want this to be as deterministic as possible.

We want them to all independently arrive at the same decision.

Great.

So this is how validation works.

This is what nodes do.

Major problem, though-- sometimes, we need to change the validation rules.

Sometimes, we need to upgrade the software.

I mean, software is not a static thing.

The software that these nodes are running, there's going to be bugs.

There might be security vulnerabilities.

We might want to add new features.

And so we're going to want to change the validation rules.

It's pretty much inevitable.

And, yet, given the nature of the system, we can't upgrade every single node at the same time. So I think a useful analogy here, something to think about, is actually browsers.

So web browsers-- right now, a lot of people run Chrome.

But not everyone runs Chrome.

And Chrome actually has auto update where it will push changes to your browser.

Google will automatically upgrade you.

But the way the browsers used to work is that you didn't always have auto update.

Or it would bother you, and ask you if you were OK with auto updating before.

And people would say, I don't deal with this right now.

I'm in the middle of something.

I don't want to auto update my browser, and have to restart the whole thing.

And so what would end up happening is you'd have significant numbers of people on very different versions of browsers.

And this was kind of a disaster for the internet, because people who were developing websites, they wouldn't have any confidence that the majority of their users, or a significant majority of their users, had upgraded to the new features.

So they had to support all these old versions of browsers.

And it was a nightmare.

So that's what we're dealing with here in cryptocurrencies, except it's even worse.

So you definitely cannot guarantee that everyone will upgrade their software at the same time.

You can't guarantee that everyone will even upgrade ever.

And so the designers of the software, when you want to create a new version of software, you have to think really, really carefully about it.

So let's say that we're in this situation right here, where these three nodes are running V1 of the software.

The developers have worked out V2.

V2 has some interesting new features, which slightly changed the validation rules about what a node running V2 will consider valid or invalid.

So now, when we see this new block, and we ask the question, is this block valid, these nodes might make different decisions.

In particular, maybe the old nodes will think it's valid.

Great, passes all the validation rules.

I will append this block to my blockchain.

And the new node will not think it's valid, and will say, garbage block, fantastic.

I will not append this block to my blockchain.

And what we end up with when this happens is we have two different versions of history.

The nodes running the old version of the software think that there are four blocks in the blockchain.

The nodes that are running the new version of the software have decided that that block is invalid.

And they only have three blocks in the blockchain.

So what does this mean?

First of all, is this like the situation we saw before, where if we just wait for enough proof of work this will sort itself out? What do you guys think?

I'm seeing some people shaking their heads.

And, yes, this situation is different than the situation we saw before, because the situation we saw before, all the nodes still had the same validation rules. They just saw the blocks in different orders, or they got later, whatever.

All of that can be resolved by looking at the proof of work.

Sometimes when this happens, sometimes when nodes are actually running different versions of validation software, things can be resolved by just looking at proof of work.

But quite often, actually, they can't.

And so in this situation, what might happen is another block comes along.

So this is a different block than this one.

And this set of nodes might decide that this is the next block in the blockchain, whereas the blue nodes will never, ever, ever validate this block.

They will never decide that this block is valid, given the version of the software they are running.

They will never switch to a blockchain that has this block in it, even if it has the most proof of work.

So I think this is a really important point.

And I think it sort of fundamentally gets to why miners don't actually have as much power in the system as one might think that they do.

And the reason for that is that the way bitcoin is designed is that a node will never accept a chain with a single invalid transaction.

If, according to its set of rules, a single transaction is invalid, it will completely ignore that chain.

Doesn't matter if it has the most proof of work. Does that make sense to everyone? It's kind of intense.

It's kind of crazy, right?

Like, even if it has mountains upon mountains of proof of work, it will completely ignore that chain because there's something invalid about it.

So miners can mine what they want.

But they better mine within the validation rules set of the economic majority of the nodes.

Otherwise, those nodes are all going to ignore whatever they mine.

And their money is not going to be worth anything. Great.

So in this situation, we end up with two different blockchains.

And we actually end up with something quite irreconcilable, unless we change the software again, somehow.

And what we've ended up with that's irreconcilable is we now have a block over here, and a block over here.

This node will never accept a chain with this block in it.

And so we've got two different blockchains. So when this happens, we call it a fork.

Now, the word fork can be used to mean many different things in software.

For example, you fork a project on GitHub.

When you fork a project on GitHub, you might make changes to it.

This is a little bit different because of the linearity of history here.

When a fork happens in a cryptocurrency, in particular a sort of a fork that can't easily be resolved, that isn't just going to, oh, one side has more proof of work, OK, great we're all going to switch to that other side.

When you have a fork that can't be resolved that easily then it's actually a pretty big deal, because like we were saying earlier we now have two versions of history, two different versions of money.

It's like I took the dollar from my wallet, and I photocopied it.

And now there's $2 where there was only one.

So maybe you like this kind of dollars.

Maybe you don't like that kind of dollars.

We've basically taken this entire network that all sort of agreed on what this form of money was, and we've divided it into two. So there's two different kind of forks that people will talk about mostly in this industry.

But forking is actually much more complicated than that.

You can't just talk about forks as in-- so the two different kinds of forks are soft forks and hard forks.

And I'm going to explain what a soft work is, versus a hard fork.

But, actually, that delineation, it gets much more nuanced because different things happen in a soft fork versus a hard fork, depending upon who's decided to adopt the rules.

OK, so a soft fork-- a soft fork is a fork which is backwards compatible.

So we have some rule set.

And under that rule set you apply that rule set to every single possible one megabyte chunk of data.

And we're going to end up with some set of what are valid blocks.

A soft fork is when you add rules to the rule set, such that the set of valid blocks actually shrinks. So what does that mean?

We had a whole bunch of rules before.

Blocks have to be less than 1 megabyte.

Transactions all have to be valid.

It means that their scripts certify, like validate, under the certain script interpreter, et cetera, no double spends.

Let's take one of the rules that's actually pretty simple, which is that blocks have to be less than one megabyte.

Let's say that I change that to blocks have to be less than two megabytes.

Is that a soft fork? AUDIENCE: Yes. NEHA NARULA: Oh, wow, there's some actual dissent over this, right?

OK, so someone who says no, why is the answer no?

AUDIENCE: Because you have something that is now accepted that wasn't accepted before.

NEHA NARULA: Right, so remember, valid blocks before newly valid blocks fit inside of this.

Now, someone who said yes it's a soft fork, why do you think that?

AUDIENCE: All blocks still have less than two.

NEHA NARULA: Right.

So it's actually not a soft fork.

And the reason that it's not a soft fork is because a block that is 1.75 megabytes, previously, the old rule set, was invalid.

New rule set, it's valid.

But we just said everything that's newly valid has to fit inside what used to be valid.

So a soft fork is restricting the set of valid things.

You can think of it as expanding the set of rules that a block now has to comply with.

So a feature of this is that if you don't upgrade-- so V1 is the old rule software.

V2 is the new rule software.

If you don't upgrade, you will continue to see the new blocks as valid.

And this is actually a very nice property, because, remember what happened over here, we had this property

where the people on the blue side were just never, ever going to accept one of these blockchains.

And if it were the case that this were a soft fork, the old side would never, ever-- well, if this sorry, if it were the case that this weren't a soft work, the old nodes would never accept this new stuff.

So soft forks are really nice, because they're backwards compatible.

Old nodes will still see the new blocks as valid.

They won't reject them.

If that chain gets a ton of proof of work, then the old nodes will happily switch over to that chain.

So, what does that mean?

So let's say that the black nodes are the old rules, and the blue nodes are new rules, new rule blocks.

And, again, we have to look at this from two different perspectives.

There are the people creating the blocks.

And there are the people reading the blocks.

And sometimes those are the same people.

Sometimes, they're not.

So we're looking at blocks here.

So these are people who are creating the blocks. And, again, both of these sets of people might upgrade or not upgrade.

So we might have miners who are following the old rules and miners who were following the new rules.

Then we might have validating nodes who are following the old rules, and validating nodes who are following the new rules.

And this is not even talking about SPV yet.

OK, we're just talking about miners.

And we're talking about nodes.

So let's say some of the miners upgrade to the new rules, but not all of the miners upgrade to the new rules.

The miners who didn't upgrade to the new rules are going to continue producing old rule blocks.

They're going to continue producing V1 version blocks.

But if the majority of miners upgrade to the new rules, remember, the new rule people have new rules.

So they have more restrictions on the set of blocks that they will accept.

So they might not see that block as valid.

So they're never going to accept that chain, because it has invalid things in it.

Once you've upgraded, you might no longer accept the old chain as valid.

And so what's going to end up happening is something that looks very much like the orphaning situation that was happening before, where if the majority of the hash powers, which is over here, this chain is going to keep going bigger.

These guys might keep trying to produce old style blocks.

But they just keep getting orphaned off.

It's actually like very much in their economic interest to hurry up and upgrade, so they're not wasting hash power on orphan blocks. Are there questions about soft forks? Yes?

AUDIENCE: So are soft forks also used for changing software other than validation?

NEHA NARULA: Yes, that's a great question.

Are soft forks also used for changing software other than validation?

So we can certainly upgrade more than just the validation parts of the bitcoin software.

If it doesn't change the set of blocks that would be accepted or not accepted, we don't even bother calling it a fork.

So let's say that we wanted to change something about the peer to peer network, or the way that things are stored on disk, or something like that.

We would see that a change like that, if it didn't change the set of blocks that would have been accepted or not accepted, if after the change a set of blocks accepted or not accepted is the exact same, we don't call that a fork,

a change that induces a fork.

It's only when the set of blocks are accepted or wouldn't be accepted are actually changed.

Yes?

AUDIENCE: How often do updates get pushed on the network?

NEHA NARULA: Great question.

How often do updates get pushed on the network?

So Bitcoin is at version 0.16 right now.

And 0.16 came out like a week or two ago.

And I think it's sort of been on the order of a couple months between different versions of Bitcoin.

Other cryptocurrencies release more frequently.

Some release less frequently.

Some do forks very, very regularly.

And we'll speak a little bit about that.

But the software is pretty much constantly getting updated. Yeah?

AUDIENCE: So in practice, people are just updating their versions?

NEHA NARULA: Yeah, so in Bitcoin in particular, Bitcoin does not have an auto update function.

So in order to run the new version of the software, you have to decide that you're going to run the new version of the software.

Or there might be a way to decide to turn auto update on.

I can't remember.

But the point is that they try to leave it in their user's hands.

Yes?

AUDIENCE: How are versions created?

So if you like on the GitHub, there are a ton of features, and sometimes they're pushed forward.

Is there a subreddit, where they're like, OK, version? NEHA NARULA: OK, how are versions created?

It's actually very similar to any other software project, where the people who work on the software project get together, and decide which pull requests are going to be part of the new version.

So, yes, at some point, they're just like, new version, release.

Yeah? AUDIENCE: Do software get released as updated versions?

NEHA NARULA: Yes, soft forks are definitely released as updated versions.

So that's a great question.

I think not every version has a soft fork in it.

But if there is a soft fork, it will definitely be a new version.

Yes?

AUDIENCE: Why can't this black miner not just look at the version in the blue block and decide not to?

NEHA NARULA: Because it's not a blue block.

So you're talking about here, right? OK, so the nodes are running different software.

And then a block comes along that-- ideally, yes the block would have a version number in it that specifies which version of the software.

And this is a very useful field in the block header.

But that's not really-- so what it has a different version?

AUDIENCE: If it's lower than mine-- sorry, higher than mine, then I don't do it.

Then I won't update first.

NEHA NARULA: What do you mean by don't do it?

You mean, you ignore it?

AUDIENCE: No, I [INAUDIBLE] myself, because I see that a higher version is in circulation.

NEHA NARULA: So that would be one way of running the network, which is if I ever see anything that indicates that there's newer software out there, I just realize I'm old, and that I need to update. I think that Bitcoin simply made a different design decision, which was that older nodes should still be able to-- because think about nodes that were sort of set it and forget it, they're out there running.

One tenant that is absolutely true is that users will not upgrade.

They will not all upgrade.

It just won't happen.

You will have old versions of software running.

And given that you will have old versions of software running, I think it's a question as to whether or not those nodes should still be a part of the network, or whether you just want to slice them off because they're not running the latest version.

And I think in Bitcoin, they want those nodes to still be a part of the network.

They want people to be able to make the active decision not to upgrade.

Tadge actually, I think is still running 0.13, or some older version of Bitcoin.

Still works great.

He can still validate transactions totally fine.

OK, so we just talked about soft forks.

And I think the important thing to remember about soft works is that soft works are backwards compatible.

We also talked about an example that is not a hard fork, which is if we decided that we wanted to allow blocks that were less than two megabytes, instead of less than one megabyte.

And so a 1.75 megabyte of block appears.

That is not a soft work.

What if we changed the rule to say that, instead of only accepting blocks that are less than one megabyte, we're

only going to accept blocks that are less than half a megabyte.

Is that a soft work or hard fork? AUDIENCE: Soft fork.

NEHA NARULA: That's a soft work.

The reason that that's a soft fork, is because a block that is less than half a megabyte was also still less than one megabyte.

So it's still valid under the old set of rules.

So if the change made here was now we want smaller blocks.

So we're only going to take blocks that are less than half a megabyte.

That's fine.

Everyone will still see this chain as valid.

This is still backwards compatible.

Just because your blocks are 300 kilobytes now doesn't mean that they violate a rule.

AUDIENCE: Can I ask one last question in relation to that?

So what happens if I want to do something with the old version, and [INAUDIBLE]..

And they realize, wow, I love [INAUDIBLE].. NEHA NARULA: Right, so what happens to all the previous stuff in my chain, right?

Now, ideally, the way that the rules would be setup, the new rules, is that they would take effect sometime in the future.

So you wouldn't apply those rules retroactively to the things that you already put into the chain, because then you just throw away all history.

And that would be kind of a bummer.

Instead, you say, after this block, we are only going to accept blocks that are less than half a megabyte.

AUDIENCE: So what happens if after that block, so a majority of miners are running old software.

NEHA NARULA: Great question.

AUDIENCE: [INAUDIBLE] NEHA NARULA: We'll get into that.

We'll get into what happens when the majority, versus minority, of different parts of the ecosystem decide to adopt the change, versus not adopt the change.

But first, we're going to talk about hard forks.

What is a hard fork?

A hard fork is not backwards compatible.

And so a hard fork is a fork where, if this was the old set of rules, under the new set of rules you actually accept more blocks than you would have accepted before. So just to be clear, under the new rules, you might produce a block, which a node running under the old rules will reject.

This is not backwards compatible, because any chain that has this new type of block in it will not be accepted by any of the old nodes, full stop.

It doesn't matter about proof of work at all. OK, so the thing with hard forks is that because they're not backwards compatible, they're a little crazier and a little bit sort of harder to manage in your network.

Because if you do a hard fork and not everyone upgrades, you might end up with two chains, possibly forever. If we assume that those nodes are not going to upgrade-- and let's say they're not upgrading because there's no one really maintaining them.

Then they're not going to upgrade in the future.

And they're not going to upgrade to a change that might fix what's going on here.

They're just going to keep doing their old time thing.

Then with a hard fork, these nodes are never going to accept the new blocks.

It doesn't matter what the proof of work is on that chain.

So hard forks versus soft works-- hard forks, as I keep stressing, are not backwards compatible.

Guess what?

A fork can be a hard fork and a soft work at the same time.

Yay.

Super confusing, actually, because you can both restrict the set of rules and expand the set of rules at the same time.

Basically, it means it's a new, overlapping, but not entirely inclusive or not inclusive, set of rules, very unfortunate.

In fact, most things should actually be both, to avoid some of the things that I'm going to show you in a minute.

Yes?

AUDIENCE: But then for all useful purposes, isn't it just a hard fork, because that means some nodes won't be inside the blocks.

NEHA NARULA: For all useful purposes, it's not just a hard fork.

Let me see if I can-- The reason that it's not just a hard fork-- this is something that requires a little bit more thought.

It's not just a hard fork.

But I will think about how to explain this, and perhaps say something on this.

Or, yeah?

AUDIENCE: [INAUDIBLE] NEHA NARULA: It is.

But he's asking, isn't that just a hard fork?

And it's not just a hard fork, because I think old nodes will do something different.

Ah, OK, so there's two reasons why it's not just a hard fork.

One is because of reorgs.

So different things will happen as the majority of hash power starts to accept or not accept the change, whether or not all nodes will switch over or not.

But if it's both a hard and a soft work, they will never switch over.

But I think the question is whether old nodes can even still function.

So I think one way to use combination hard and soft works is to actually stop the old nodes from being able to have blocks at all.

Like, you design things in such a way, that given a majority of hash power moves over, they're just going to end up with nothing.

But this is a really good question.

And we should go into more detail on this.

And I will try to figure out a way to get that to you guys.

OK, so who controls forks? Yes? Sorry?

AUDIENCE: The developers.

NEHA NARULA: OK, the developers, which I didn't even write down as a note here.

OK, do the Bitcoin developers control forks?

Who thinks yes?

You're not even raising your hand.

Who thinks no? Determines whether or not a fork is going to go through.

There's the Bitcoin developers, who are actually writing the software, and who also have a lot of influence in the system, because presumably they understand it really well if they're writing software.

So they can write some software, and decide that they want to do a fork.

Does that mean that the fork is going to go through?

No, certainly not, everyone can ignore them.

And there have been situations in the past where everyone ignored them for quite a long time, and argued about it, and didn't push a fork through.

Well, miners are the ones who actually create the blocks.

Are they the ones who control forks?

I see a very strong head shake no.

Why do you shake your head?

AUDIENCE: So if the miners decide to change what types of blocks they're putting out, they can do that.

But the nodes that are actually validating it, no one can say they're not going to accept it.

NEHA NARULA: Very true.

So as we saw before, there are times when nodes will not accept a chain no matter how much proof of work it has on it.

I think that's a very common misunderstanding.

The nodes run their own validation rules.

And so if I'm running a node, I can run whatever validation rules I want.

And I can decide what chain is the one that aligns with my interests.

Yes? AUDIENCE: If you're doing a hard fork, and you are accepting new blocks, and they just never mined any blocks with those new rules.

NEHA NARULA: Exactly.

So there's certainly incentives in many different directions.

If all the miners decided to go to one side, then I'm a full node with my side of validation rules who has no new blocks.

Great, what am I going to do?

How am I going to make transactions?

How am I going to spend my money?

Also, let's say all the miners don't go to one side, but most of them do, and most of the nodes go to the other side.

Then what am I going to do?

Who's going to listen to me?

Who's going to trade with me?

Who's going to take the coins on my side of the fork?

So I guess that's just to say that it's complicated.

There's the developers.

There's the miners.

There's nodes.

There's the users who determine the economic value of these tokens, and which side they think is more valuable or less valuable.

And we've seen this happen with lots of cryptocurrencies multiple times.

And it's kind of exciting.

So Bitcoin has forked multiple times.

And every time it's forked so far, it has created a whole new version of the coin.

So it's been like a pretty serious hard fork.

And so now, in addition to Bitcoin, there's Bitcoin Cash, Bitcoin Gold, Bitcoin Diamond.

Ethereum also has fought.

In Ethereum's case, they tried to take along everyone with them.

And so the hard fork was supposed to be an upgrade to Ethereum.

But some people decided they didn't want to go along with it.

So there's Ethereum Classic.

And there's Ethereum. Both of them have value.

So it's really unclear who controls forks.

So now, let's talk about hash rate and forks.

So nodes are pretty-- OK, what nodes will do is they have a set of validation rules.

If a block validates under their set of validation rules, they will accept it.

And if all the blocks validate under their validation rules, they will accept the chain that those blocks are on.

And if there are two different chains where all the blocks validate under their validation rules, they'll take the one with the heaviest proof of work.

That's what nodes do.

So heaviest proof of work is clearly kind of important here, if both sides of a fork validate.

So if both sides of a fork validate-- so we have to think about this from the perspective of nodes who upgrade to the new software, and nodes who don't upgrade to new software.

So it might be worth it to try to kind of write this down a little bit. So we have the old rule nodes, and the new rule nodes. And let's just talk about one kind of work for now.

Now, let's say less than 50% of the miners upgrade. And let's say greater than 50% of the miners upgrade.

OK, so let's use this chart, as we're kind of trying to reason about what's going to happen in the ecosystem for soft works and hard works.

So what happens if a soft fork does not obtain more than 50% of the hash rate? So we've got the old rules. We've got nodes.

Some nodes have upgraded.

Some nodes haven't upgraded.

Less than 50% of the miners have upgraded.

So the old rule chain is going to be longer, because they're going to continue mining on the old rules.

By the way, sometimes they don't ignore things with the new rules.

Sometimes they do.

It's very complicated.

But let's just pretend right now that they are ignoring things with the new rules, because they haven't upgraded, and they don't see that. Or, sorry, they would see the new rules as valid.

But the majority of the hashing power is over here.

So what's going to happen? Yes?

AUDIENCE: The benefits of the fork aren't going to really be useful much, because be there'll be some blocks with the new rules.

But the majority is just not going to change.

NEHA NARULA: Right, so it kind of depends on the soft fork.

So it depends on what the new rule nodes think of old blocks.

If they think that old blocks are still all valid, then they're going to build upon old blocks just fine, because they think they're valid.

If they think that old blocks are no longer valid, then they're going to try to do their own chain. But the majority of the hashing power is with the old side.

So they're just going to get reorged out.

Yes?

AUDIENCE: If it's a soft fork, but they also think all the blocks are valid, how is it a fork at all? NEHA NARULA: So an example of this is taking what was an unused op code, and now making it meaningful.

So it just so happens that none of the old rule nodes ever used this silly unused op code.

Because they never used it, they never had any invalid uses of the op code.

So the new nodes will still accept old blocks, because none of them use this op code anyway.

But if they were to use the new op code, and they used it as a no-op op code, instead of what the new rules are then it would be an invalid block.

Does that make sense? So you have to think really carefully about how you are designing these upgrades to your software.

You have to think very carefully what will happen to the old nodes.

What happens if 50% of the miners take this thing?

Well, under the old rules, was anyone actually using this thing, that we are now maintaining must be used in a new way?

Or pay to script hash is a good example of this as well.

Pay to script hash, technically, looks like you're just paying to a hash.

So old nodes will see it, and say, OK, if you can produce a signature that shows that this is valid, blah, blah, whatever, whatever, that's fine.

But nobody ever did that.

Nobody ever did that in a way that was meaningful under the new rules.

No one ever used it.

So all the old things will still validate, simply because no one ever did anything that was under the subset of the new rules.

Sorry, this is very kind of complicated.

So what we have here is that if old rule blocks are still valid, the software will get reorged out.

So no fork there, no fork there.

You're just not really going to end up with stuff in the blockchain.

If old rule blocks are now invalid, then we're going to end up with two chains.

Do people see why this is?

Because the new rule people are not going to accept any chain that has an invalid thing in it.

So they're not going to reorg out.

They are going to continue.

And they are going to have their own chain.

And they're going to ignore all of the blocks on the other chain, which they deem to be the invalid chain.

Yes?

AUDIENCE: Is this the case for Bitcoin Cash?

NEHA NARULA: No, it's not the case for Bitcoin Cash.

It's actually the case for something called a user-activated soft fork, which the bitcoin community threatened to do when the miners wouldn't do a soft fork they wanted them to do.

They said, fine, you miners won't do the software that we want you to do.

So, normally, people don't like the situation.

So you don't want this to happen.

So, normally, soft forks are written such that if you don't get 95% of the miners to sign, on the soft fork never activates.

So that's the way that, traditionally, soft works are done in the Bitcoin ecosystem.

What does that mean?

It means 5.001% of the hash power can stop a soft fork from going through. So usually, things are written in such a way that you really want to guarantee this two chain thing never happens.

We see hash power is kind of voting.

And if you can get 95% of the hash power to agree with the soft fork, then only will the soft fork activate.

And the way that miners do this is they signal in their block headers that they are going to adopt the soft fork.

And then the software is usually programmed in such a way that it looks at some sliding window of block headers to determine whether 95% of the hash power has decided that they're going to go along with it.

And it only activates once that's the case.

So doing protocol upgrades is really hard.

It's not a trivial matter.

You don't just push some code.

So usually, it's the case that you want to wait for a significant majority of the mining power to indicate support for the soft fork.

And, by the way, they can indicate support without actually running the soft fork.

This happens all the time.

But you want to wait for significant support from the mining community before actually activating the soft fork.

What happened recently with something called Segregated Witness, which we'll talk about, which was a soft fork to Bitcoin to introduce new functionality that the miners opposed, was the community actually said, forget it.

We are going to start running the new rules.

We're going to ignore you.

We don't care what the majority of the hash power does.

We are going to go ahead and create this two chain situation, and follow the chain with the new rules.

Interestingly, this actually worked.

I think it was like 80 plus percent of the hash power was not in favor of the soft fork.

And the users managed to create this kind of detente where they forced everyone to go along with what they wanted, super interesting. They had no actual way to vote.

It was entirely sort of via Twitter.

AUDIENCE: [INAUDIBLE] NEHA NARULA: Yes, so this is one of the major reasons, and I think some of the strongest evidence, that miners have nowhere near as much power as you actually think they do in these ecosystems.

Certainly have power, and part of the power they have is to do what's called an evil soft fork, which I'm not going to talk about in this class, but I think we probably should talk about at some point in time.

OK, great.

Let's see.

OK, if the soft fork is greater than 50%, old rule blocks will follow the new rule blocks automatically.

So we get soft fork successful.

We do not have two chains, because these will simply be orphaned and abandoned.

And, in fact, there's actually pressure for the miners to move over to the new rules so that their blocks aren't abandoned. Yes?

AUDIENCE: So [INAUDIBLE] making it easier mine, are those considered types of forks?

NEHA NARULA: It depends.

So the question is around difficulty, and the number of leading zeros required in the proof of work.

That changes all the time.

There's a formula which determines what that number should be.

So, no, it doesn't require a fork.

But if you want to change the formula, that would be a fork.

Requiring more proof of work would be a soft work.

Requiring less proof of work would be a hard fork. OK, now let's talk about hard forks.

So what happens if a hard fork doesn't obtain 50% of the hash rate? AUDIENCE: [INAUDIBLE] NEHA NARULA: It depends, again.

It depends on if the old rule blocks are still valid.

So we're doing a hard fork.

We're changing the set of valid blocks.

If blocks are still valid under the old rules, then we're going to end up with a situation like this.

If old rule blocks are still valid according to the new rule nodes, then the new rule nodes will just follow along.

OK, so basically no work happening here, because the new rule nodes are going to produce blocks.

They're not going to have enough of the hash power.

And so they're not going to be considered valid by the other set.

So they're going to end up following along with the old rule nodes.

Yes?

AUDIENCE: Again, isn't the whole point of a hard fork that you're expanding the set of things that are valid?

So aren't all old blocks going to be valid in a hard fork?

NEHA NARULA: Not if it's a combination hard fork, soft fork.

So maybe I should split this up into whether or not it's a combination hard fork, soft fork, or not.

OK, so what happens if a hard fork doesn't obtain greater than 50% of the hash rate?

What's going to happen then? AUDIENCE: We'll be mining to the old rules.

NEHA NARULA: Yes, we're going to end up with two chains.

This is going to have chain-- whatever, the point is, there's going to be an old chain and a new chain.

The old nodes are never going to accept the new nodes blocks.

And we're going to end up with two chains forever.

And this has happened multiple times. Are there questions about hard forks? Yes?

AUDIENCE: Are there ways for the hard fork to make life really difficult for the smaller chain?

NEHA NARULA: Are there ways for a hard work to make life really difficult for the smaller chain?

That is a good question.

I think it depends. So there's multiple ways to make life difficult.

I think what you're asking is not, what if the hard fork becomes more popular and more people decide to move over to it, and abandon the smaller chain?

That's one way in which life could be more difficult.

But another way that life could be more difficult is if the hard fork somehow made the previous chain no longer a

productive chain to be on.

And I think that the primary way that that would happen would not actually be a hard fork, but would be a combination hard fork, soft fork.

But there are definitely ways for miners to play with hard forks and soft forks in such a way that-- see, this is actually kind of ideal in some ways.

This is like, OK, we disagree on what the rules should be.

I want some new set of rules.

You don't want some new set of rules.

Fine, let's just split the ecosystem.

You guys go your way.

I'll go my way.

We'll have two different coins.

Life will be great.

What's kind of not so nice is when one side tries to coerce the other side to go along, which is kind of what a greater than 50% soft fork is.

You're kind of coercing the other side to go along, because they have no way of opting out.

And there are ways that miners can make the side of the chain with less harsh power basically useless. This is totally something miners can do.

People talk about it occasionally.

It just doesn't seem like they figured out that they can do it, or they haven't tried to do it.

I don't know.

Maybe they don't want to do it.

I'm not sure.

It would be pretty disruptive to the ecosystem. OK, so question-- we've talked about what happens here from the perspective of full nodes, and from the perspective of miners.

What happens from the perspective of people with wallets who might be running SPV? What do we think? Depends on which node they're connected to.

So one problem with SPV is that they don't actually get the blocks.

So they don't actually validate blocks.

All they validate is proof of work.

Some changes that make a block invalid might show up in the header.

But a lot of changes that might make a block invalid, or new blocks valid, don't show up in the header. So given what we've learned about how SPV works, what happens when there's a fork?

So let's go back and take a look at what SPV wallets see.

They see the block headers.

You have to download all the headers, which includes the nines for the proof of work, the Merkle root for the transaction, timestamp, stuff like that.

They also get Merkle paths for the transactions that they've told the node they're interested in. So what would happen if there were a soft work or hard fork?

What would happen to an SPV? They don't really have a say in the situation.

They aren't really participating in validation.

They're not participating in mining.

And they're kind of at the mercy of whatever node that they are talking to.

And, in fact, if they try to talk to multiple nodes, they might get multiple different answers.

And they don't really know which answer is the right answer.

And this could get really tricky and complicated.

Now, I think probably it's fair to say that the majority of the Bitcoin ecosystem, in terms of users, actually does not

run their own full nodes.

They interact with the Bitcoin network, at best, through SPV, and at worst through a trusted third party like Coinbase.

So what does the ecosystem have to say or do about whether or not forks are accepted or not accepted? Yeah?

AUDIENCE: They don't really have to.

NEHA NARULA: They don't really have a say.

They kind of I would say at like a meta level, in that users can pressure Coinbase to start following Bitcoin Cash which is a hard fork of bitcoin.

They kind of have a say at an economic level, in that if Coinbase starts accepting Bitcoin Cash, then the price of Bitcoin Cash will go up.

And so that's kind of a big deal.

But in terms of like a protocol level, in terms of accepting or rejecting transactions, they don't really have a say because they can't see enough to determine whether or not they're interested in following a fork or not following a fork.

In fact, it's even worse than that.

There were multiple situations where we ended up with two chains.

So an SPV wallet might think that they received money on one chain, and they didn't receive money on the other chain.

So very bad things can happen to wallets and to SPV clients during a fork.

And it's incredibly disruptive.

And people who create software on the behalf of users-- so not even running SPV, but like the Coinbase's of the world, and the exchanges of the world-- they have to do a fair amount of work to make sure that they aren't going to lose users' money when there's a fork.

And the exchanges, and the Coinbase's of the world, have screwed this up in the past.

There's been money lost.

There are ways to safely design your fork so that it's less likely that wallets will do the wrong thing.

But I think we're kind of just figuring out how to really do that effectively.

AUDIENCE: [INAUDIBLE] NEHA NARULA: Yeah, so this is kind of pretty bad, because if Bitcoin is really going to undergo a hard fork or soft fork, everybody is paying attention.

If Ethereum is going to do it, everyone's paying attention.

But there's so many altcoins out there.

And, yeah, there's so much opportunity for people to take advantage of the disarray that happens when a fork happens.

So I think we're right in the middle of trying to figure out best practices for handling forks safely, and also, giving users some way of choosing what side of a fork that they want, because there's this sort of thought that eventually there will be economic pressure to-- for a long time, people thought that if Bitcoin hard forked, one side would win, and the other side would go to zero.

And now, we've seen that that is absolutely not true.

In fact, when Bitcoin hard forked, sometimes the total market cap of one side plus the other side is bigger than the total market cap of what Bitcoin used to be.

So money is literally created out of nowhere.

So hard forking is very lucrative.

It literally gives every Bitcoin user more money.

So if you own Bitcoin within the past-- like, if you owned Bitcoin a year ago, you probably have like four other coins right now.

I don't know.

You should go sell them, or sell the Bitcoin, and buy the other coins, or whatever you want to do.

But the point is, you have more money than you might have even known that you had, because someone has done these forks.

Don't even like get started about the tax consequences of this.

It's very confusing, extraordinarily confusing.

And people are trying to figure out what to do, because you didn't ask for this fork.

Like, I as a holder of Bitcoin, didn't decide that I wanted Bitcoin Cash to be a thing.

I didn't accept Bitcoin Cash.

But now, the same private keys I used to control my Bitcoin also control the same amount of Bitcoin Cash.

It's kind of crazy.

Let's talk about forks in practice.

So soft forks in practice-- there have been lots of soft forks.

Soft forks are very useful.

Soft forks are helpful.

This is the major way that people upgrade software.

And, again, the way that they do it is they try to make sure 95%, or some huge majority of the miners, are on board.

And that way, it's sort of like as non-disruptive to the ecosystem as possible.

So pay to script has was a soft fork.

The introduction of Segwit was a soft fork.

In operation, check sequence verify was added as a soft fork.

So that's kind of why they're all these like unused ops out there.

In case we want to use one of them for something new, we can do that with a soft fork.

Hard forks in practice-- there's lots of new Bitcoins, Bitcoin Cash, Gold, Diamond, like I said.

Ethereum went through a really interesting hard fork.

And there's some cryptocurrencies that actually hard fork quite frequently.

For instance, I learned the Monero hard forks about every six months.

Vertcoin, how many hard forks have you guys done?

Three?

Three hard forks.

So, despite the way that I said that hard forks are very disruptive, if you can get all the nodes to upgrade, it's not really a big problem.

So for cryptocurrencies where the nodes are really well connected, or they auto update, or things like that, hard forks are much less disruptive.

So let's talk about the Ethereum hard fork for a moment.

So this happened two years ago-- a year ago?

I can't remember when this happened, two years ago, I think, almost.

And so what happened was we haven't talked a lot about Ethereum, so we'll be coming to explain Ethereum in much greater detail.

But the point is Ethereum had a smart contract which had a bug in it. And someone hacked the contract, and started siphoning off ether from the contract.

So the Ethereum team all got together, and decided, we're going to rectify this wrong.

Now, rectifying the wrong-- I mean, I already said it's a hard fork.

But why is rectifying this wrong a hard fork not a soft fork?

AUDIENCE: Because you want to change history.

NEHA NARULA: Because you want to change history.

That's an interesting way of putting it.

And the way that we change history is not by actually going back in time and editing history, unless you are Accenture I guess, which creates editable blockchains.

But the way that we edit history, is we add something to the blockchain which indicates that we're going to undo, or change, or rollback or somehow create an operation, that does something that effectively undoes what happened in the past. So the Ethereum developer got together.

And they originally tried to do this as a soft fork, actually.

So it could have been a soft fork.

And the way that the soft work would have worked is that it just would have ignored all of the transactions that were sort of-- I can't remember exactly how it worked.

But, basically, the general idea was, ignore all of the transactions that might spend the hacker's money.

Ignore the transaction where people gave money to the hacker, what ultimately became the hacker, which is a new rule.

Now, you're shrinking the set of acceptable blocks.

And the problem was they realized that if they did this as a soft fork, then they were inadvertently creating an attack vector, because people could now spam the network and potentially waste the resources of the miners-- very challenging to do soft forks and hard forks correctly.

So they decided, OK, you know what?

We're just going to do it as a hard fork.

When this block comes about, block 1,920,000, we are going to simply transfer around 12 million either from the hacker's account into this other account, this other contract, which is going to give the money back, where people can reclaim their money.

So literally just hardcoded in this transfer.

85% of the hash power in Ethereum went along with it.

So that means 15% didn't.

And that means that there are two currencies now.

There is Ethereum, and Ethereum Classic.

And they persist to this day, almost two years later.

And when I looked this morning, it was about a 30 to 1 ratio between Ethereum and Ethereum Classic.

Yes?

AUDIENCE: So Ethereum Classic says, we are not OK with you guys undoing this hack.

NEHA NARULA: Exactly, so Ethereum Classic says, we're not messing with people's accounts like that.

No, no, no.

What happened happened, fair and square.

We are not changing history.

And so in Ethereum Classic, the hacker has money.

On Ethereum, the hacker does not have money. Yes?

AUDIENCE: [INAUDIBLE] one specific contract.

So the argument is we shouldn't be fixing people's mistakes.

They should learn-- Was this analogous to the Parity?

NEHA NARULA: Yes.

So here they decided, there was a bug in someone's smart contract.

It was exploited.

We're going to give people their money back.

This happened again with a different type of smart contract in a different way a couple months ago.

And they decided-- well, they could still decide to do it.

But so far, they decided not to give everyone's money back.

So there's about 90 million or something dollars worth of ether that's locked up in these contracts, due to a bug.

And in the case of the DAO, they decided, let's do it.

In the case of this Parity bug, they decided, let's not do it.

And if you talk to the Ethereum Foundation people, they'll talk about governance.

And they'll say, well, we feel like this made sense for these reasons.

But this doesn't make sense for these reasons.

But it's still it's a set of people getting together and deciding whether or not to give money back.

Yeah?

AUDIENCE: [INAUDIBLE] NEHA NARULA: No, it's not the Ethereum Foundation.

The Ethereum Foundation was not a fan of Ethereum Classic.

It's a separate group of people.

AUDIENCE: [INAUDIBLE] NEHA NARULA: No, actually, so it turned out, interestingly enough-- and this was something that I saw at a conference that looked into who were the contributors to all of these different forks, when trying to understand governance?

It actually turns out that no one from the Ethereum Foundation works on Ethereum Classic.

And no one who used to work on Bitcoin works on Bitcoin Cash, Gold, or Diamond.

So it's entirely new sets of people.

Yes?

AUDIENCE: The former lead developer of Bitcoin core is working on Bitcoin Cash.

NEHA NARULA: He's not actually.

You're talking about Gavin Andreessen?

Yeah, I don't think he actually makes commits to Bitcoin Cash.

AUDIENCE: In a general sense, he's contributing on GitHub, not specifically to any implementations.

But to protocol, block-level exchanges.

NEHA NARULA: Interesting.

Yes?

AUDIENCE: [INAUDIBLE] NEHA NARULA: It depends on how you define contribute, certainly.

This is a really interesting example of a hard fork.

In summary, just to wrap up here, forks are extremely challenging.

Upgrading software in decentralized cryptocurrencies is extremely challenging.

And so the lesson that I want you to take from this is, unless you need what a decentralized cryptocurrency has to offer, you will probably be better off not using a decentralized cryptocurrency.

They are very hard to update.

They're very hard to upgrade.

There are frequently bugs.

And so I think that forks and the challenges around forks are an excellent reason not to do this when you don't have to.

And also, what's really interesting, I think, is that your traditional consensus systems don't address this problem at all, because your traditional distributed consensus systems don't have any notion of what nodes are going to find valid, or not valid, or changing the rules of validity.

And so these problems really only arise when you start to have these validity rules that you're dealing with, and you have nodes that might take one set of rules, and nodes that might take another set of rules.

The consensus mechanism here ultimately becomes irrelevant if you don't agree with the validity of the blocks.

It doesn't matter that a majority of hash power said that this is the blockchain.

It doesn't matter if a majority of signers say this is the blockchain.

If you don't agree that the things in that blockchain are valid, you're going to ignore that.

Majority be damned.

So it's very different than distributed consensus.

Okay, and so real quick, last thing...

on Wednesday we will have another guest lecturer. Her name is Sharon Goldberg.

She's a professor at Boston University and she's worked on a lot of really interesting things. In particular, something really cool that she's worked on is using the peer-to-peer network to create attacks in Bitcoin, and then very recently she nor co-authors released a paper on peer-to-peer network attacks in Ethereum. And so she's gonna be here to talk about the peer-to-peer networks behind these things and to share some of the research that she's done.