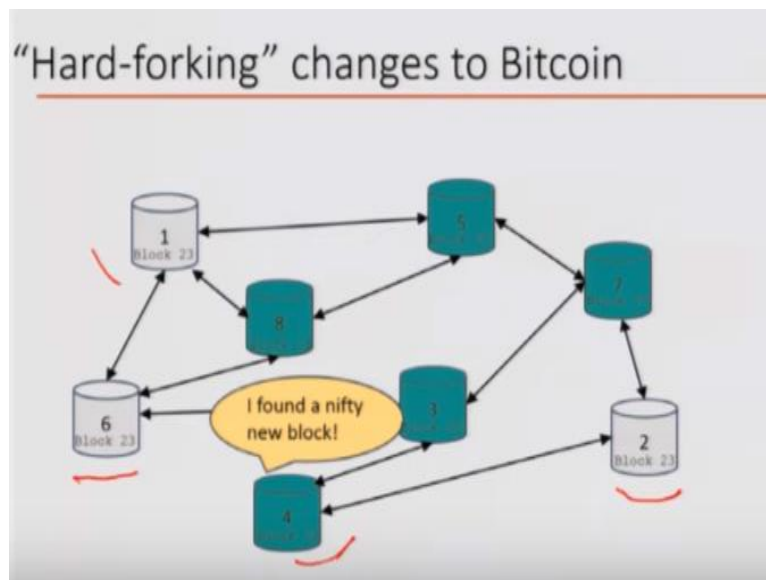


Introduction to Blockchain Technology & Applications
Prof. Sandeep Shukla
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture - 12

Welcome back. So we have been talking about the bitcoin and it has certain limits, hardcoded limits and we were discussing that you know in the future, if we want to change these limits, what are the process or procedure for doing so.

(Refer Slide Time: 00:40)



So it turns out that since all the activities on bitcoin blockchain depends on the mining process, whether they will be accepted or rejected, and they will be made permanent it is not that easy to actually change the software or to change the limits, which are hardcoded or change the mining process or change the opcodes.

For example, in the scripting language, or to actually do a different kinds of hashing algorithm or signature algorithms, because it would require some work. So it turns out that what happens, so we already know that when a block is mined, the one solves a puzzle. But at the same time, multiple people, multiple nodes might actually solve the puzzle at the same time.

And everybody who solves a puzzle, which is computationally very expensive, they expect that that their mind block will be made permanent in the blockchain, so they broadcast their block. Now depending on where you are located with respect to the

miners, your block versus somebody else's block might first make it to one of the miners. And that miner might actually attach that block to the blockchain.

Whereas some miner on the other side of the world, who gets another block, which also has solved the puzzle, and add that to as the next block to his copy of the blockchain, and that is what is called a forking. Now fortunately, over time, these forks usually converge. And so one of these two blocks will actually survive in the longest chain of the network. So this is not a deterministic guarantee.

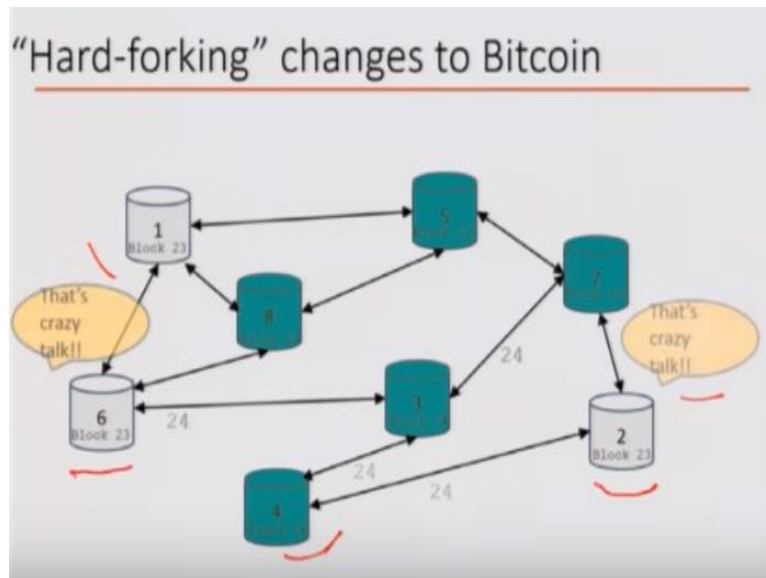
But this is a probabilistic guarantee with a high probability. So that is how a single chain which we call the consensus chain survives, and whatever is on the consensus chain is considered the permanent blockchain any transactions on any of the blocks on that consensus chain is considered permanent and but forking is a natural process of the natural phenomena that happens.

However, when you want to change something intentionally in the blockchain code or in some of the processes, then there may be two possible things. One is called the hard-forking and another is called a soft-forking. So let us first look at hard-forking. So what happens is that suppose you make a change and some of the nodes upgrade their software to reflect that change.

So that change could be that you have changed opcode or you have changed the hashing algorithm of the bitcoin by you know re-implementing one of the opcodes. And then based on that, the blocks let us say, number 4 here, this node, mines a block and he expects that when he broadcasts this block, all the nodes will actually validate the block validate all the transactions inside.

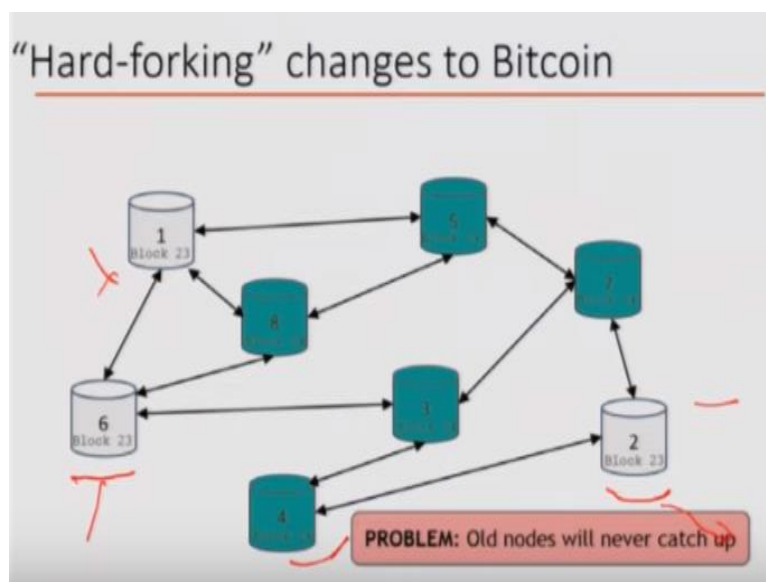
And the validate that this solution to the puzzle is in fact a solution and then they will keep attaching it into their copy of the blockchain. But now here we have an example where some of the nodes did not upgrade and therefore, either intentionally or they are not aware of the change.

(Refer Slide Time: 04:32)



So when the block goes to them, they will consider that block to be invalid and they will not accept that block and therefore, they will not attach that to their copy. And this will happen with all the nodes that do not have the new upgrade.

(Refer Slide Time: 04:50)



And this way, the copy of the blockchain that is being maintained by this nodes and what is being maintained by these green nodes will differ. And at this point, there will be two different consensus chains. And this will become suddenly two different blockchain with a shared history. So until this point when the upgrade happen the consensus chain versus a convergent chain, and then suddenly there will be two of them.

And that will require the blockchain to have two different things and accordingly the currency will also split. And this happened to bitcoin, like for example, bitcoin cash is one of the hard fork actually the original bitcoin consensus chain that continued and then the regular bitcoin is based on an upgraded version.


(Refer Slide Time: 05:45)

Soft forks

Observation: we can add new features which only *limit* the set of valid transactions

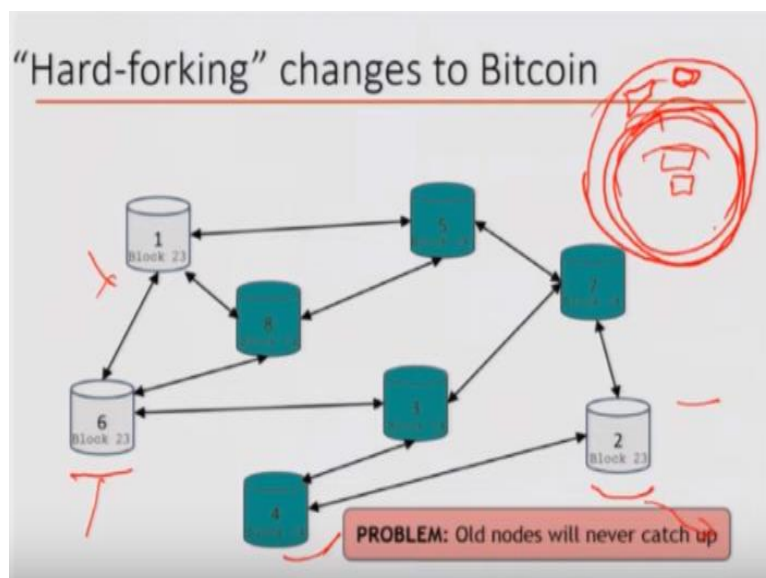
Need majority of nodes to enforce new rules

Old nodes will approve



So one example and the soft fork would be when you add features, but so in this case, in the previous case, in the hard-forking case, what happens is that the let us say this is the set of valid blocks.

(Refer Slide Time: 05:54)



So now you are talking about set theory. So or any anything that any property that is satisfied by these blocks can be thought of as a set. You know it may be a very large

actually maybe an infinite set of all possible blocks that are valid. And then you suddenly decide that you will increase the set of valid blocks, right.

So now there are blocks here in the in your expanded set of valid blocks that are not considered valid by this other nodes which are not aware of this extension or which do not agree on this extension. And therefore, the blocks mined according to the extended rule will be attached to the chain that are maintained by these green blocks whereas, the chain maintained by this older blocks will be only having blocks that satisfy properties of this set.

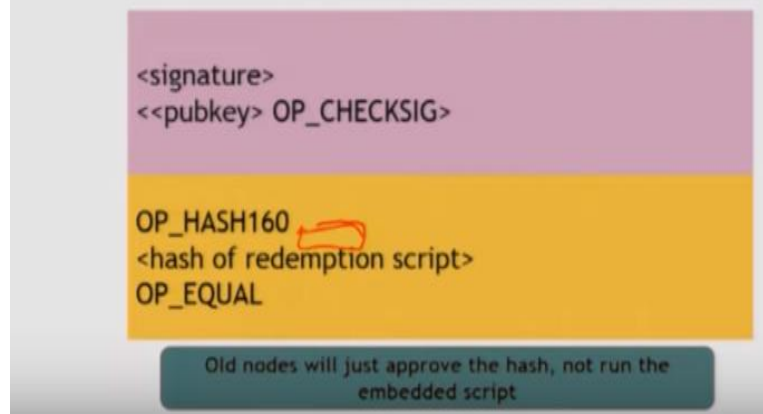
Now consider the opposite problem. Suppose you decide that I will this was my set of valid blocks. And now I decide to restrict it and say that here are some of the restriction. So I have a subset of blocks that I will consider valid. In this case, what will happen is that the blocks that are acceptable to the un-upgraded ones also have this property because this is a subset.

So any block that was here or here they are part of this larger set. So therefore, any node that was aware of this larger set will continue accepting blocks that are either here or here. However, when the nodes that are not upgraded, will start mining blocks according to the rules that only that does not put the block in this set, but rather only in this set, they will be not be accepted by others.

And very soon this nodes un-upgrade nodes will realize that the when they mine a block and they are not restricting their blocks within this set they are being rejected. So very soon they will catch up, and then they will also upgrade. So that is a soft fork, because in this case, eventually there will be a single consensus change. So it was a fork for a while and then there was a single chain.

(Refer Slide Time: 08:35)

Soft fork example: pay to script hash



So soft fork example is earlier we discussed pay to script hash, where we said that the burden of writing the script was not with the sender of the money or the initiator of the transaction, but the receiver does a complex type of receipt of the bitcoin. So they write a script and provide the hash, right. So hash is like an address of that script.

So now when that was introduced, right so any block that contains a transaction of this type can also be permitted by the old the nodes that do not know about this, because they will only check the hash part, right. So see it was an address here receiver address, that would be also a hash of 160 bits. And if it is an address of the script, that also is a 160 bit hash. Therefore, they will accept it.

And therefore, they will approve the hash, but they will not run the embedded script. So there, when they actually verify transaction, they might actually accept a particular transaction, which is not valid. But otherwise, if valid transactions are mined by the new nodes that are upgraded, they will be also permitted by the older nodes. So that is why it will be a soft fork and not a hard fork.

So this is how the pay to script hash was introduced in blockchain after the initial you know bitcoin was introduced.

(Refer Slide Time: 10:15)

Soft fork possibilities

- New signature schemes
- Extra per-block metadata
 - Shove in the coinbase parameter
 - Commit to UTXO tree in each block

So certain other things that will not require a hard fork or new signature scheme. So if you introduce a new signature scheme that could be fine. And if you start to require that in a block, there should be some extra metadata. That could be also be fine. You can actually put it into the coinbase parameter or you can commit it to the UTXO tree, the Merkle tree that you maintain, so it should not be a hard fork.

(Refer Slide Time: 10:45)

Hard forks

- New op codes
- Changes to size limits
- Changes to mining rate
- Many small bug fixes

UTXO

Proof of work

Proof of stake

Proof of Authority

Ethereum

So what are the cases when hard fork is required? When new opcodes are introduced. So you have a new opcode so that this script cannot be run by nodes that have not upgraded because they have not they would not recognize the opcode. If you change the size limits of the blocks, right so then you need a hard fork because the large let us say you have changed it to a larger size, then you will the nodes that are not upgraded will start rejecting the blocks.

And so if you change the mining rate, or if you make bug fixes, remember that we talked about a bug in the multisig transactions, where it actually pops an extra data. So the data has to be replicated intentionally by the script writer. So if you try to fix that bug, there will be a hard fork and that is why it has not been fixed yet.

So what we understand from all this is that changing something in the blockchain in the block chain's code or block chain's, you know engine that, you know that executes the bitcoin scripts or the hardcoded limits, you have to go through a forking process. And that is quite expensive. Now this is true for you know most proof of work based blockchain or even in case of a proof of stake or proof of authority blockchain.

So let me quickly tell you what these other two means. So proof of stake means that it could mean a few different things depending on what is meant by stake. So in some cases, if you have a the who is chosen as the next block maker, right. So in case of proof of work, you make it very democratic.

Well, I would not say democratic but you make it such that your chance of being the next block maker is proportional to the or your probability of being the next block maker is proportional to your computation power right your hash power. Now if you do not go by that, we saw that this is a very expensive process in terms of computation, in terms of energy. So let us say you do not want to do this.

So you may choose to have a different way of choosing the next block maker. So one possibility is that you make it proportional to the amount of coin holding by the different users. Now in bitcoin that is not possible because in bitcoin, we do not have a notion of accounts. So every time you make a transaction, you can use a different address and so all the information in the state of the blockchain is maintained in terms of unspent transaction or UTXO's right.

So you cannot do a proof of stake based selection of nodes based on how much they hold, because it is not clear how much the node holds because a node might be operating many public addresses. And you do not know that they belong to that

particular node. But in case of Ethereum, as we will see, in this session as we start looking into Ethereum, there is a notion of accounts and therefore each user has an account.

And all the coins that they actually own is actually consolidated in the accounts. Well, one can actually create multiple accounts with different addresses that is true. But then the stake will be based on the accounts. So now the question is that if you decide stake on how much money they have, then one that has more money will always be selected and that will not be fair, because then it will become a centralized system.

So people have devised other ways like for example, they have considered a mixture of how much money and what is the age of the coin like for how long they have been holding the coin, because one that is holding the coin for a longer time probably has more interest in keeping the money for a longer long-term investment. So that kind of thing is used as a stake.

Proof of authority is slightly different and this cannot be also done in a blockchain that does not have a proper notion of accounts or proper notion of some kind of a identity because proof of authority basically makes the chooses the next block maker from a set of privileged or authorized nodes. So not every node there are can be a miner like in bitcoin or in a proof of blockchain but nodes that are already chosen by some authority or some mechanism may be a voting process.

But some authority, so among those set of nodes which have the authority, there could be an algorithm like a round robin algorithm or some random choice algorithm or a mixture of stake and authority could also be used. So these are the different types of blockchains. Now certainly as we see here is that the proof of work is the most expensive one and also to make changes in the blockchain code or in the blockchain hardcoded numbers, proof of work will actually require a fork.

In case of proof of stake, it may or may not need a fork, but certainly in proof of authority, you can actually consolidate the changes in only in the authorized nodes and therefore, it may not require a fork. So but we will we will discuss that later if

there is time, if we get the chance to actually compare these different ways of deciding we will mine we will do the next block or we will create the next block.

And we will see that in nonpublic blockchains like hyper ledger and a few others would for example, none of this things are done. And instead there is a Byzantine fault tolerant consensus mechanism that is quite different from what these things are. Now that is where we will draw the line about bitcoin blockchain. We may come back to it once in a while to compare with the other blockchains.

But we have gone through the basics of bitcoin blockchain, the concept of transactions, concept of unspent transactions, the concept of consensus through proof of work or mining, the concept of blocks and concepts the cryptographic concepts required to make the to show the authority or authenticity of the transaction sender. We also saw the notion of hashing and hash pointers to ensure the tamper resistance properties.

And we also saw the execution of scripts to actually make transactions which are more complex than ordinary transaction of sending just a few coins from one sender to a receiver. And we said that those are kind of smart contract but not exactly smart contracts. Now we what we will do is that we will go to our next example of a blockchain and which is the Ethereum, which was originally conceived with the hindsight of what is missing in the bitcoin blockchain.

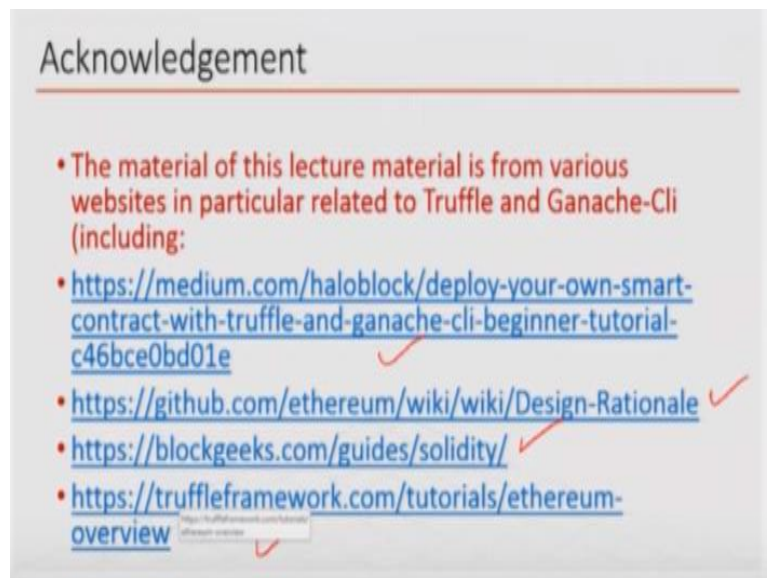
And then it got its own different ideas. One of the ideas inventors of Ethereum worried about is that a bitcoin blockchain is purely for cryptocurrency and purely for transactions. So if you want to store data permanently for example, let us say you want to store grades or you want to store medical information of patients or you want to store land record information, then we saw there are some ways like you can put them in the coinbase transaction as a coinbase parameter.

But or you can use a, you know proof of burn type of transaction. But it is not the most convenient and actually, bitcoin wants to discourage that kind of use of their blockchain. And that Ethereum wanted to address. So Ethereum wanted to create a blockchain infrastructure in which you can do all kinds of computation, not just

necessarily coin transactions or coin mining, although they also do coin transaction and coin mining.

But you can do all kinds of smart things with Ethereum. So let us now go and look at the situation or the concepts that are there in the in this Ethereum blockchain. So we will start in this session, a little bit about Ethereum. And then we will continue in the next session. So first of all, let me tell you, where you can find more material. So obviously in lectures you cannot learn everything.

(Refer Slide Time: 20:53)



And so let me first say that we collected a number of concepts from various documents and blogs that we are going to present first about Ethereum and then we will go a little bit deeper and compare it with bitcoin, then move to private blockchain because in bitcoin there is no private blockchain concept but in Ethereum you can create your own private Ethereum blockchain.

And from there we will jump to more commonly used private blockchains. So these are the material that you should also read, because when we talk about them, a lot more detailed explanation of what we are talking about could be found in this documents.

(Refer Slide Time: 21:42)

Revisit Blockchain

- So far, with bitcoin, we saw use of blockchain as a
 - Creation engine for digital currency
 - A distributed, tamper-resistant log of transactions in crypto-currency
 - A distributed ledger that solves heuristically Byzantine-safe distributed consensus
- Now, we look into more generic blockchain
 - That can be thought of as a distributed execution engine for consistent program execution
 - That can also support crypto-currency and in fact uses crypto-currency to solve consensus problem
 - That can be used to enforce contracts between parties through smart contracts

So to understand Ethereum we need some context right. So far, we saw bitcoin only as example of a blockchain. And if you just know bitcoin, then you get the impression that blockchain and cryptocurrency are kind of synonymous. Because the bitcoin blockchain was a creation engine for a digital currency, and it creates a tamper resistant log, which is distributed and replicated at various nodes.

But this information is majorly about cryptocurrency transactions. And then the consensus mechanism there is under the assumption that any of the nodes could be malicious or any of the nodes could fail. And then how do we make sure that you can keep the ledger consistent across all replicas and that is what the consensus problem is all about. And we saw that proof of work and expensive method is used as a heuristic for Byzantine safe or Byzantine safe distributed consensus.

So that is what we saw in bitcoin blockchain. Now blockchain in the beginning of the course and repeatedly I have been saying is not necessarily about cryptocurrency and in fact, the cryptocurrency is probably be most harmful application of blockchain. But it has a lot more generic application and it is actually a very, I would say transformational platform in which computation, storage of information safely and integrity preserving way is possible.

So what Ethereum showed is that you can think of a blockchain as a distributed program execution engine for consistent program execution. So you can actually think of this as a platform in which there are many nodes and all the nodes are capable of

executing the same program and to verify that the outputs of the program are the same. And therefore, once they have verified that they can store that as a result of those execution.

They can also store the code of the program so that in the future also they can verify that the execution results are in fact correct. But also it does support cryptocurrency, generates cryptocurrency by doing consensus, solving the consensus problem in fact using proof of work like bitcoin. And it can be used to enforce contracts between parties through smart contracts. So what is a contract?

A contract is usually an agreement between two parties, that if you do this, then I will give you this or if we do this together, we will get something from a third place together. And these contracts so far are usually enforced by a trusted third party. Otherwise nobody will agree to a contract. Either you trust each other when if you trust both sides, then you agree to a contract and you know that both sides will hold to the words of the contract.

If not, if you do not trust each other, then you have a trusted third party, who will ensure that the both sides hold the deal after the work is done, or after the conditions have been satisfied to fulfill the obligations under the contract. Here, we want to create a situation where there is no trusted third party and the platform itself enforces that the contracts are the obligations of the contracts are met by all the parties involved, while the parties do not necessarily have to trust each other.

And that is what the smart contracts are all about. So a program execution, distributed program execution engine for consistent program execution, a cryptocurrency framework, like bitcoin, and then through smart contracts create a contract enforcement mechanism where parties do not necessarily trust each other.

(Refer Slide Time: 26:21)

Agenda of this lecture

- Relook at the Blockchain technology
 - Why use a blockchain?
 - What is a blockchain?
 - How does a blockchain work?
- The Ethereum Blockchain
 - What is Ethereum? ✓
 - What is a smart contract? ✓
 - Ethereum Networks ✓
 - Distributed Applications (Dapps) ✓
- Truffle and Ganache-CLI for your first smart contract

So what are we going to do in the lecture about Ethereum? Compare or we will recall some of the things about blockchain technology. And then we will look into the Ethereum blockchain, particularly what exactly is Ethereum? What are the smart contracts? What are the Ethereum networks? And what are the Dapps or distributed application. This is a term that you will hear a lot in the context of Ethereum because distributed application is what we get on top of the smart contracts.

And then at the end, we will talk to you a little bit about some of the software that you will use like Truffle, and Ganache-CLI for writing your own first smart contract. Now remember that you can actually have a whole course on writing smart contracts, making sure smart contracts are correct, using various tools that actually check the correctness or kind of like, simulate smart contracts or even do some program analysis to check the security of smart contracts.

But that is not what we do in this course, right. So in this course, you will just get to understand what the smart contracts are? What do they look like? How do they execute? And get some basic principles in your mind and then it is up to you whether you want to run with that information and an introduction to some of the tools to self-teach yourself more about smart, especially Ethereum smart contracts which are written in a language called Solidity.

And then you can go with that and make yourself an expert in Ethereum. So when we come back, we will go on and look into blockchain technology in the context of Ethereum and then go into Ethereum.