

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

Lecture - 26

Welcome back to Blockchain Technology and Applications. So last session, we talked about the IOTA and the Tangle blockchain. Now as I mentioned before, the other blockchain that I want to cover is the Corda blockchain, which was created by a company named R3. And it kind of originated in the space of Finance.

So various finance companies, they actually have different types of contracts which are actually written in legal languages. And these contracts actually are such that over time various events, especially time driven events happen, when certain activities are executed based on what is written in the contract.

So as a result, what happens is that over time the contract state changes from just inked contract to maybe some part execution of the contract, and then another part execution of the contract and so on. And eventually, the, these activities may repeat for a while or the contract state might reach maturity.

Whatever happens all this information currently, well before Corda or similar type of solutions came in, came into existence was kept in each individual contracting parties, maybe two banks or one financial institute and a company or individual borrower or lender and a bank, etc. These contract are kept in each of these entities individual IT system.

And they might have very different understanding of what the contract really implies or what activities need to take place or even about when an activity has taken place, whether it has taken place properly or whether one entity owes something to the other entity. This is this has to be reconciled manually, in order for them to come to a shared truth perception.

So in order to make these things more autonomous or and more smooth this notion of a blockchain based solution came in. And the idea was that the existing blockchain

solutions such as bitcoin or Ethereum, or even Hyperledger was not too generic for this specific application they had in mind.

So therefore, Corda from the very start was very much customized for this kind of financial legal contracts, their execution and keeping each party involved in such contract and contract execution on the same page without having to do any kind of manual reconciliation. Now, if now financial Institutes they actually enter into similar contracts with multiple different other financial institutes.

And therefore they have many different contracts ongoing at the same time, but not every institute that they interact with are party to each and every contract right? So contracts maybe a two party contract or maybe three party contract, but whatever it is not every other entity for a particular specific contract would be interested to know the terms of the contract, nor do the individual parties involved in that contract for the sake of confidentiality want them to know about it.

So therefore, a completely public ledger or completely shared ledger will not solve the need of this particular type of application. So therefore, the design of Corda was quite different from the blockchains that we have seen before. It is highly customized. It is a permissioned blockchain and it is also you know, a need to know basis information sharing. And consensus is also limited to the parties that need to consent on any transaction.

And the transactions in this case would involve execution of a particular activity envisioned in the contract and corresponding change in state of the contract.

(Refer Slide Time: 06:41)

Acknowledgements

- Richard Brown, R3
- Razi Rais, Microsoft
- Corda Whitepaper: <https://www.corda.net/content/corda-platform-whitepaper.pdf>
- Corda Technical Whitepaper: <https://www.r3.com/wp-content/uploads/2019/08/corda-technical-whitepaper-August-29-2019.pdf>

So with that in mind, let us look at Corda. So most of the material in this you can actually download this to whitepapers. One is a very high level whitepaper for Corda and then the other one is the technical whitepaper. And then the Richard Brown and Razi Rais they have written number of blogs, which actually explain this whitepapers and other related technological details in various blogs which also you can find online.

However, we are not going into as much depth about Corda as we have done in case of bitcoin Ethereum etc. But just to give you a flavor of a very customized blockchain for a very specific niche application, we are going to discuss Corda.

(Refer Slide Time: 07:50)

What is Corda? Is it a Blockchain?

- Corda has no unnecessary global sharing of data: only those parties with a legitimate need to know can see the data within an agreement
- Corda choreographs workflow between firms without a central controller
- Corda achieves consensus between firms at the level of individual deals, not the level of the system
- Corda's design directly enables regulatory and supervisory observer nodes
- Corda transactions are validated by parties to the transaction rather than a broader pool of unrelated validators
- Corda supports a variety of consensus mechanisms
- Corda records an explicit link between human-language legal prose documents and smart contract code
- Corda is built on industry-standard tools
- Corda has no native cryptocurrency

So first question that one would like to ask is, what exactly is Corda? And is it a blockchain? So there are many differences between Corda and a blockchain like Ethereum or Hyperledger, etc. So first thing that would jump out of the whitepapers is that Corda has no global sharing of data, because not everybody or every party involved in this data sharing platform would be interested in what is happening between parties, which are engaged in a legal contract regarding some financial activities.

So only those parties with legitimate need to know can see the data within an agreement. Now Corda has another concept called a flow. Basically, when you talk about a transaction, even in case of bitcoin or Ethereum you talk about a transaction which involves different entities and which has a workflow.

So for example in bitcoin if you are making a payment transaction, so the workflow would be actually to first consider one of your UTXOs from which you will be used as input to your transaction. Then you have to create two recipients or one recipient or multiple recipients based on how you want to utilize the unused coins that you have that you are possessing by virtue of that UTXO.

And then you will also create the corresponding bitcoin script with the scripting language and then you will broadcast that transaction. Then each node which sees this broadcast transaction will actually execute the bitcoin script by taking a part of the script from the UTXO transaction and part of the script from this transaction and execute it and if there is a flawless execution, then it will correspondingly consider the transaction to be valid.

But that is not the end of the transaction being permanent or somebody else the one of the recipients to spend the coin that you have given that person, because this has to now make into a block and then all the nodes will make their own blocks and some of them will take this transaction into their block. Some may not take them into their block and then what will happen is that all of them will engage in solving the proof of work puzzle.

And then whoever wins will then broadcast it and then others may also solve the puzzle and they might also broadcast it and then for a while there will be a rest condition. Then whichever part of the if this block in which your transaction has been included, has been there added to the blockchain and then maybe six more blocks have been built on top of it, only then the other party can start using that coin that he or she received from this transaction.

So that is the entire workflow for exchanging or transferring some money from one entity to another in bitcoin. Similarly, in case of Ethereum, you have a workflow that your smart contract you know, in your transaction, you invoke some function in a smart contract and then it goes to everybody and then everybody executes the smart contract, you know, as per required for this transaction.

Then they change the state information and then it becomes it may become part of the next block etc. So there is always a transaction is not just a single just you know you just send a transaction broadcast a transaction and money changes hand or our information gets recorded. No it does not happen like that, there is a flow. So Corda uses this.

So flow is some kind of a first class concept in Corda and so the flow is choreographed and this flow is choreographed without a central controller as required by any blockchain platform. Now consensus normally in bitcoin or Ethereum happens by participation of everybody who wants to get involved in the consensus by the proof of work.

And the reason why a proof of work is required is because we do not we cannot trust any single entity that is participating in the blockchain. But we have seen that in Hyperledger, the consensus actually is much simpler, because we have, it is a permission blockchain, everybody has a digital certificate which binds that participant's real world, real world identity to its digital identity.

And therefore, we basically make the consensus very simple by having a set of nodes or a single node even in the case of solo where they actually do the ordering of the transactions. And as far as consensus is concerned, the consensus about the ordering

is what Hyperledger cares about and it is done by kind of like offloading that service to the consensus service, which may be implemented as solo as raft which is crash fault tolerant or by some kind of a Byzantine fault tolerant mechanism.

So Hyperledger is the consensus is not at the level of the system, but more at the level of the transactions or individual deals. So same thing in Corda. In Corda the consensus happens between the parties that are involved in that transaction and nobody else, right. So and in Corda also there is a it is a permission blockchain and every participating entity has a strong identity.

And therefore, although one can say that through some mechanism of certification, one can actually make these entities anonymous, but they are, that is not the main purpose of Corda. So Corda is also designed in such a way, so that there may be observer or regulatory nodes, which have visibility to all the transactions, all the activities, and therefore, and we know that banking and finance is a very highly regulated industry.

So therefore, regulators need to look at the activities. It has to be transparent to the regulators. It does not have to be transparent to everybody else, but the regulators might want to see that the transactions have no irregularity and so on. So Corda is designed in such a way so there may be observer nodes and regulatory nodes which are looking at all the transactions and also be able to do auditing.

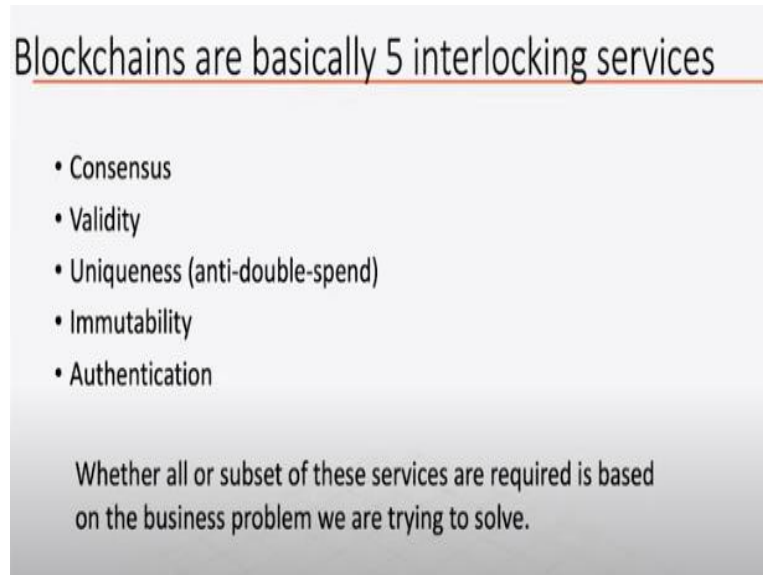
So transactions are validated also by parties to the transaction rather than from any random participants in the blockchain. Corda also like Hyperledger does not make consensus as completely monolithically entangled into the platform. So therefore, Corda also supports a variety of consensus mechanism. It records obviously, now the most of the finance contracts, they are normally written as human language like English language, legal prose documents.

But when the terms of the contract are to be executed, then that has to be done, you know, using some kind of program, and that is the smart contract. So smart contract and legal prose document they are kept in link. So we will see that you know in Corda

the records there are explicit link between the code that is written in programming language for smart contract and the legal prose of the contract.

Corda does not create its own programming environment and other things. It is built on industry standard tool, mostly it is based on Java technology. It uses messaging technology, all that stuff is already existent as industry standard. And Corda has no native cryptocurrency because that was not the purpose of creating Corda.

(Refer Slide Time: 19:17)



Blockchains are basically 5 interlocking services

- Consensus
- Validity
- Uniqueness (anti-double-spend)
- Immutability
- Authentication

Whether all or subset of these services are required is based on the business problem we are trying to solve.

So the Corda, if you look at their whitepaper, they think of a blockchain as five different interdependent services. Obviously, there has to be a consensus for consistency, there has to be validity for a transaction to be valid. And that concept of validity depends on the semantics of the particular application in hand. For example, in case of bitcoin, the transaction is valid if the signatures match of the person who is spending the coins and if the if this is an unspent coin.

In case of Hyperledger, the validity is based on the peers checking that the transactions are valid in the they actually simulate the execution of the transaction and also you know create the result of the transaction as read/write sets and at the end when after the ordering service the nodes before they persist the transaction into the blockchain they actually check about the dates of the transactions.

Of the various pieces of data that is used for a transaction execution and if there is some, you know inconsistency then they also declare a transaction to be invalid. So

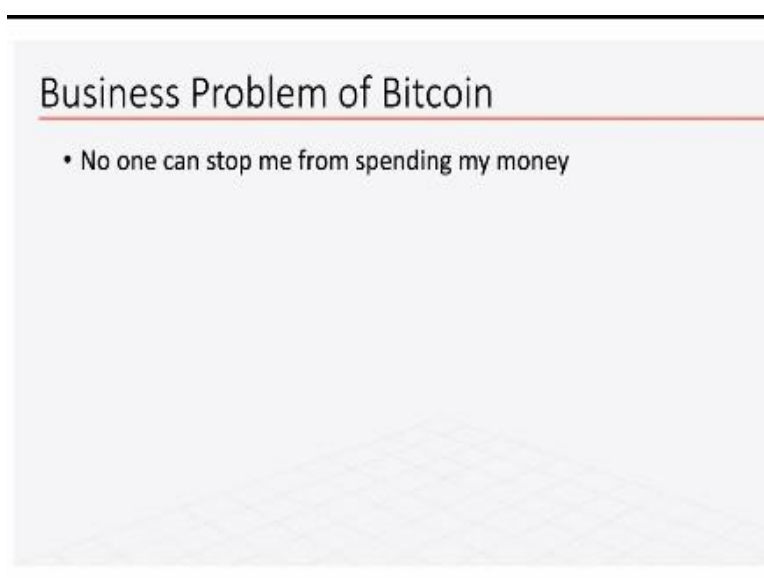
that is the different notions of validity based on the application semantics. Then there is the shake of uniqueness. And this is very important for checking for double spent. Now you might say that if you do not have a cryptocurrency, then what is doubled spent?

Now if you are having a legal service, right, and a financial legal contract. And let us say you are obliged to pay a certain amount to an entity for you know, at certain time, and let us say you by mistake or for other reason, you execute it twice. And then you are actually making a mistake because you already have, you know fulfilled your obligation but if you try to do the same obligation twice, it will be kind of like a double spend and you do not want that to happen.

Of course the records have to be immutable because that is the whole point of you know data integrity that one of the major attraction of blockchain. And in case of as in case in most blockchain this immutability is actually guaranteed by cryptographic hash function usage in a clever way in terms of hash chains basically. And then authentication is an important part of the process.

So therefore, you know of course different blockchains have these five interlocking services in different ways. Some of them mix them in the same functionality and some of them have them separated. So in Corda, we have all of these. But, you know as we will see they are used in they are organized in a certain way.

(Refer Slide Time: 23:28)



So according to the originators of the Corda, the question always is, should be that what business problem I am solving by inventing a new blockchain platform or using the blockchain some particular specific blockchain platform. So if you think about bitcoin, the idea is that the business problem we are trying to solve there is that no one should be able to stop me from spending my money, right? So that is the idea of bitcoin.

(Refer Slide Time: 24:12)

Business Problem of Financial Institutions

"The financial industry is pretty much *defined* by the agreements that exist between its firms and these firms share a common problem: the agreement is typically recorded by *both* parties, in *different* systems and **very large amounts of cost are caused by the need to fix things when these different systems end up believing different things.**"

Imagine we had a system for recording and managing financial agreements that was *shared* across firms, that recorded the agreement consistently and identically, that was visible to the appropriate regulators and which was built on industry-standard tools, with a focus on interoperability and incremental deployment and which didn't leak confidential information to third parties. A system where one firm could look at its set of agreements with a counterpart and know for sure that:

"What I see is what you see and we both know that we see the same thing and we both know that this is what has been reported to the regulator"

So the business problem that and I would leave it to you as an exercise to figure out how this business problem is solved by the bitcoin blockchain. So the business problem of the financial institutions, and this is something that I took from, I believe in from the one of the whitepapers, but this is this prose is quite, quite verbose. But I thought that this would be very important to communicate to you.

The so let us read what they have said. The financial industry is pretty much defined by the agreements that exist between its firms. And these firms share a common problem. The common problem is the agreement is typically recorded by both parties in their different IP systems. And in the in earlier days, it used to be paper files. And it turns out that oftentimes, this information as recorded by the different parties differ substantially in terms of semantics.

And therefore, a very large amount of cost are caused by the need to fix things when the different systems end up believing different things. So we have different parties, party to the same legal agreement, but they record it in their own systems and when

they come to reconcile they if they seem to believe that they are they have a different interpretation or inconsistency then there is a usually most of the time, there will be arbitration, legal challenges and so on so forth.

And there is a lot of cost associated. So the question is, how can we solve this? So imagine we had a system for recording and managing financial agreements that was shared across firms that recorded the agreement consistently and identically. That was visible to the appropriate regulators and which was built on industry standard tools, with the with a focus on interoperability, and incremental deployment, and which did not leak confidential information to third parties.

So you want a system which will record and manage financial agreements. And this information this recording should be shared across firms. So no longer the farms will have their own IT systems storing this records. They will share the record and the evolution of the record. And this record as they become part of the record, as it become part of the system shared by all the parties, there has to be a consensus before they are put into the system, right. Because everybody is looking at it.

Everybody who is party to this agreement will have to look at it and agree that the interpretation is consistent for all the parties. And so the and this record this will be recorded consistently and identically and it will be visible to regulators as well not only the parties involved but regulators. And it has to be built on industry standard tools that we can trust to be working and functional and error free and effective etc.

And this has to be interoperable between different firms. And it has to be designed in such a way so that the legacy systems are not completely it does not have to be completely replaced by this. So there will be incremental deployment and more important and most importantly, it should not leak confidential information like the contract information between two parties to third parties.

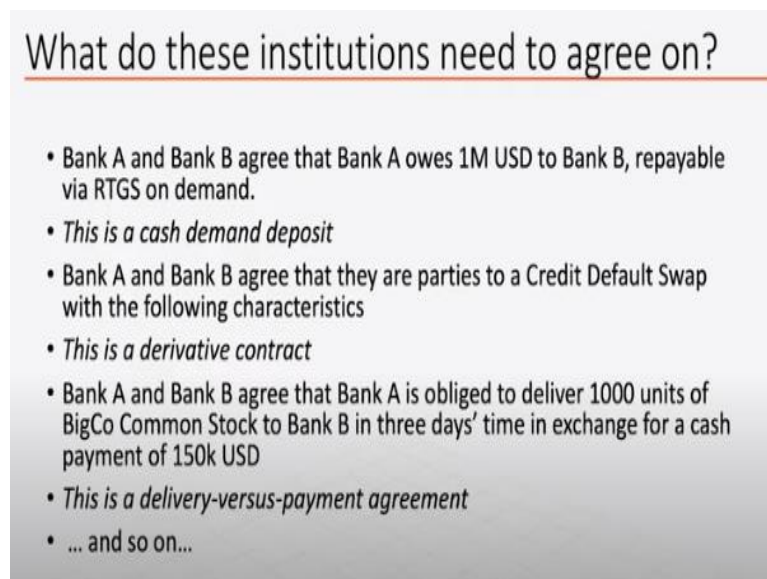
So a system where one firm could look at it a set of agreements with a counterparty and know for sure that what I see is what you see. And we both know that we see the same thing. And we both know that this is what has been reported to the regulators.

So all parties see the same thing. They also know that all parties are seeing the same thing and they also know that this has been reported this way to the regulator.

So there will be no inconsistency. Otherwise, the report by two parties will have different things about from the regulator. For example, if you are paying tax, and if your employer sends TDS information to the tax authorities, which does not match the one that you reported in your tax return then it costs the tax authorities to do audit and then you know penalize or whatever. So it basically costs a lot of thing.

So if we could have a shared information system in which tax authority, employer and employee all are looking at the same shared system, and then there cannot be any sense of different difference in interpretation, difference in the state of the system, and that would basically make things much more efficient and cheaper.

(Refer Slide Time: 30:46)



What do these institutions need to agree on?

- Bank A and Bank B agree that Bank A owes 1M USD to Bank B, repayable via RTGS on demand.
This is a cash demand deposit
- Bank A and Bank B agree that they are parties to a Credit Default Swap with the following characteristics
This is a derivative contract
- Bank A and Bank B agree that Bank A is obliged to deliver 1000 units of BigCo Common Stock to Bank B in three days' time in exchange for a cash payment of 150k USD
This is a delivery-versus-payment agreement
- ... and so on...

So what are the kind of what are the kind of things that the financial institutes agree on? So here are some examples. So Bank A And bank B agree that bank A owes 1 million US dollars to bank B repayable via RTGS on demand. So this is the kind of agreement. So here you see that this agreement requires some action on some event. So the event would be a demand from Bank B and then the action from Bank A should be an RTGS based transfer of \$1 million.

So this is the evolution of the contract. Once this is done, this contract can be closed because it is already the obligations in the contract has been discharged. So this is a

state evolution of the contract. Similarly, so this is a what is called a cash demand deposit So another example would be bank A and bank B agree that they are parties to a credit default swap with the following characteristics.

So characteristics are not described here. So credit default swap is a very complex financial instrument that banks often do. And I cannot tell you exactly at some point, I used to know what credit default swap means, especially in 2009 when this credit default swap was, had got a really bad name because of the banking crisis that came about in the US and other places in the world in 2008.

But the point here is that this is some kind of a contract and that requires some action at some future time. Another example is that bank A and bank B agree that Bank A is obliged to deliver 1000 units of BigCo common stock to bank B in three days' time in exchange for a cash payment of 150,000 US dollars. This is a delivery versus payment agreement.

So in this agreement, you see that again there is a time bound within which certain action has to be taken and the cash payment has to also be made. So this is how the contract will evolve and once this is done, this also this contract can also be disposed of. So this is these are examples of the legal contracts that these institutions get into.

(Refer Slide Time: 33:57)



The slide features the word "Corda" at the top left, underlined with a red line. Below it is a bulleted list of differences between Corda and typical blockchain technology.

- Differences with typical blockchain
 - Consensus occurs between parties to deals, not between all participants.
 - Corda lets users write their validation logic in industry-standard tools and we define who needs to be in agreement on a transaction's validity on a contract-by-contract basis.
 - Brewer's CAP Theorem
 - Data is not broadcast to every one – only to stakeholders who "need to know"

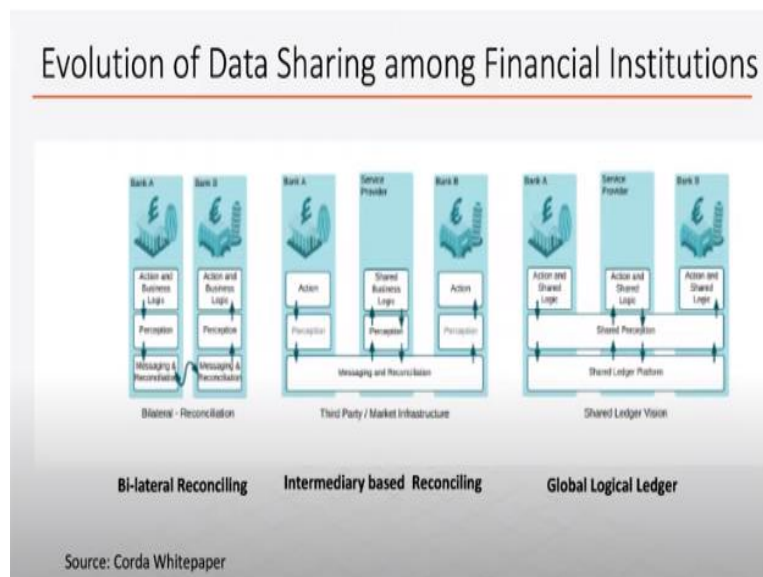
So we want a system through which all the parties involved in the contract will have this contract and have the corresponding programs or smart contracts that will be that

will be triggered based on events based on time bounds, etc. And the contract will evolve in its state. So Corda, as I said before, the difference with other blockchain is that consensus occurs between parties to deal between the parties that are involved in a deal and not between all participants.

Corda lets users write their validation logic in industry standard tools. And they define lets us define who needs to be in agreement on a transactions validity on a contract by contract basis. So a transaction validity is has to be agreed on by the parties involved in the deal and this agreement logic or validation logic has to be can be written in programming languages mostly in case of Corda in Java.

Also Corda also wants to satisfy the CAP theorem that is consistency has to be maintained, availability has to be maintained and partition tolerance has to be maintained. Data is not broadcast to everyone and only to stakeholders who need to know. So this we already discussed that these are some of the differences between Corda and any other blockchain.

(Refer Slide Time: 36:00)



So here is a little bit of history about data sharing among financial institutions. So there is something called bilateral reconciling. So here what is happening let us say we have a bank A and bank B. So Bank A does some action and execute some business logic. Then it has a certain perception about the state of the contract or whatever the deal is.

And then it sends a message to the other bank that I have done this, I have paid \$150,000 and my perception is that I should now have 101,000 common stock option for Big company. And bank B at this time has not seen did not have that perception or maybe it has the perception that it has to do it after three days and therefore, it has not done it.

So there has to be a reconciliation or there has to be negotiation that well, if you should, if the bank A was expecting immediately the stock options should be given as soon as it pays rather than waiting for three days. And the bank B thinks that no matter when the fund is paid, it has to be as long as it does it within three days it should be fine. Then there has to be some reconciliation. And that has to be done.

And another interpretation of reconciliation is the reconciliation about their individual ledgers, right. So in my ledger, I have I subtract \$150,000. So I should have in my credit 1000 stocks, and this guy should have the opposite and that has to be reconciled. So now, the other possibility is the intermediary based reconciling. So where two parties do not have to directly work with that. So it is kind of like an escrow service.

So the banks basically do the reconciliation through the middleman or a some kind of a intermediary service provider. Now this also has to be done through messaging and reconciliation. Now in the shared ledger vision, the third one where everything is actually there have there may be a service provider, there is bank one bank, A, bank B.

But every time is shared, they put everything on a shared ledger platform rather than having the perception and the ledger individually and then doing a reconciliation letter, they actually have everything in a shared ledger. And therefore, things become much more straightforward.

(Refer Slide Time: 39:22)

Principal Features of Corda (1)

- Recording and managing the evolution of financial agreements and other shared data between two or more identifiable parties in a way that is grounded in existing legal constructs and compatible with existing and emerging regulation
- Choreographing workflow between firms without a central controller.
- Supporting consensus between firms at the level of individual deals, not a global system.
- Supporting the inclusion of regulatory and supervisory observer nodes.

So principal features of Corda. So recording and managing the evolution of financial agreements and other shared data between the two or more parties in a way that is grounded in existing legal constructs and compatible with existing and emerging regulation. So you want to record and manage the evolution of the financial agreement. So you have the financial agreement, then evolution means that the obligations are discharged in various stages.

And so the financial agreement is going from one state to the other. And this is done in such a way so that all the parties involved will be able to see this in a shared ledger. And it has to be compliant to any legal or regulatory requirements, legal in the sense whatever is written in the document in the agreement document.

And regulatory means that agreement itself and the discharge of the obligations under the agreement are not violating any regulation. So then, the other thing that it does is that it choreographs the workflow between the firms without a central controller. And that is done through something called a flow which are programmed in usually in Java or Kotlin. And this is something that makes the flow a first class construct.

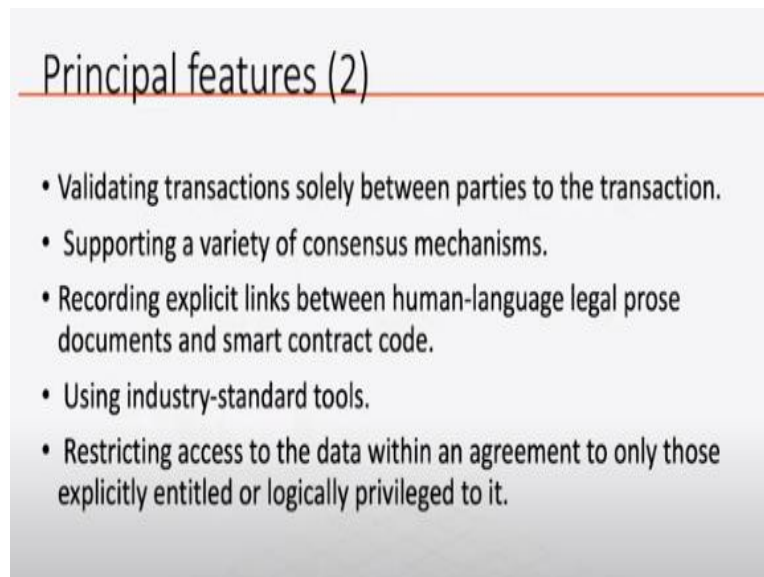
So any transaction has to basically invoke a flow that will allow the transaction to complete. And that flow might include the validation, the consensus etc. The consensus actually between firms is obviously happens at the level of individual deals and not globally. Obviously, say I am having an agreement with you to exchange something for some money.

Then we need to have a consensus between us about the interpretation or perception of what we are supposed to do and then do it as events trigger us to do it. And that we need to have consensus about that. We cannot, we do not need to invoke anybody else for the consensus. And then we have to support the inclusion of regulatory and supervisory observer nodes. So this is something that is very different.

This can be also done in case of Hyperledger. But Hyperledger does not need this. It in fact, actually Hyperledger can actually do everything that Corda does, but it has to be, you know done by you know molding a particular application for financial agreements and instruments in Hyperledger. And there I can also include a peer which is basically regulatory and supervisory observer.

But it would not be the same as Corda because of certain other things that Corda by virtue of its design provides switch in case of Hyperledger will require a lot more twisting of the Hyperledger concepts.

(Refer Slide Time: 43:23)



Principal features (2)

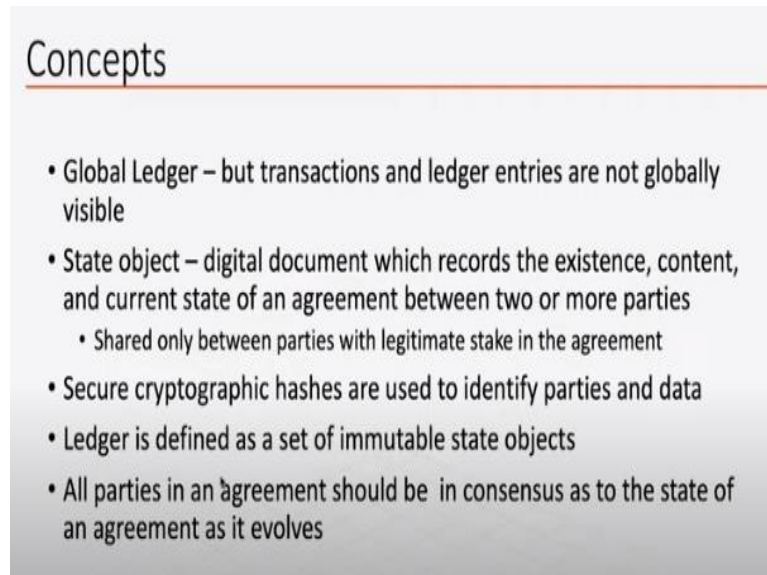
- Validating transactions solely between parties to the transaction.
- Supporting a variety of consensus mechanisms.
- Recording explicit links between human-language legal prose documents and smart contract code.
- Using industry-standard tools.
- Restricting access to the data within an agreement to only those explicitly entitled or logically privileged to it.

So also again validation of transaction is also between the parties to the transaction and nobody else needs to do it supporting. So code also supports a variety of consensus mechanisms. You know, it can be solo or it could be some kind of a crash tolerant consensus or it may be some kind of a notary, etc. So recording explicit links between human language legal prose documents and smart contract code is something

that is kept in Corda. Use of industry standard tools instead of inventing its own contract, smart contract language etc.

And then it also restricts access to the data within an agreement to only those explicitly entitled or logically privileged to do it. So only the parties involved and maybe the regulatory observer node etc., they can see the data.

(Refer Slide Time: 44:34)



Concepts

- Global Ledger – but transactions and ledger entries are not globally visible
- State object – digital document which records the existence, content, and current state of an agreement between two or more parties
 - Shared only between parties with legitimate stake in the agreement
- Secure cryptographic hashes are used to identify parties and data
- Ledger is defined as a set of immutable state objects
- All parties in an agreement should be in consensus as to the state of an agreement as it evolves

So again, Corda maintains a global ledger. But like the channels in case of Hyperledger, that people who are part of a particular channel can only see whatever happens between those parties who are part of the channel. Here is similar, but not exactly because here in case of a channel, each channel has its own ledger. Here we have a single global ledger, but the visibility of entries in the ledger are not done in such a way so that only those who need to see those entries can see the entries.

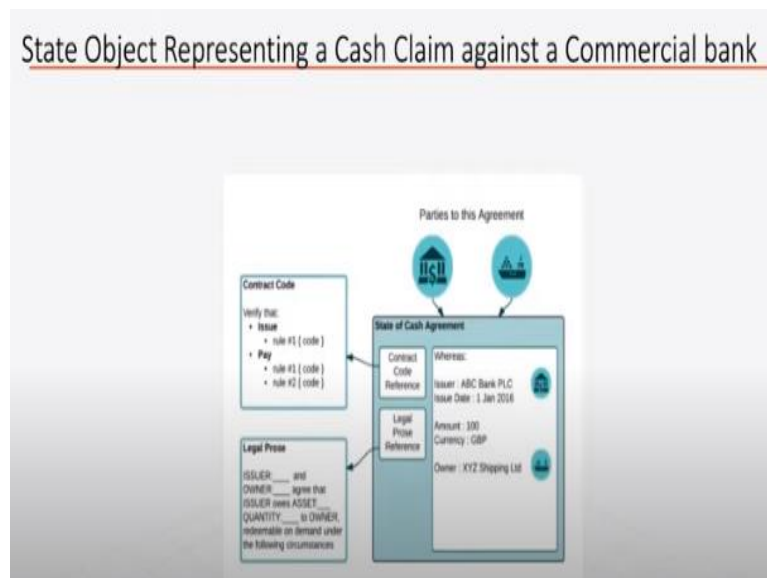
So the what is put into the ledger in case of Corda are called the state objects. So state objects basically are digital document, which records the existence the content and the current state of an agreement between two or more parties. So as I have been mentioning again and again when you have an agreement, that agreement goes through various states based on the execution after different obligations under the agreement.

So those basically are represented as state objects. So in the beginning you are in a particular state, then when some event happens and then you execute something, it

goes to a different state and so on and so forth. And this basically going from one state to the other is basically done by doing transactions. So secure cryptographic hashes are used to identify parties and data.

And this is also you know, required for data for integrity reasons. Ledger is defined as a set of immutable state objects. And all parties in an agreement should be in consensus as to the state of an agreement as it evolves, right. So that is the notion of consensus here.

(Refer Slide Time: 46:54)

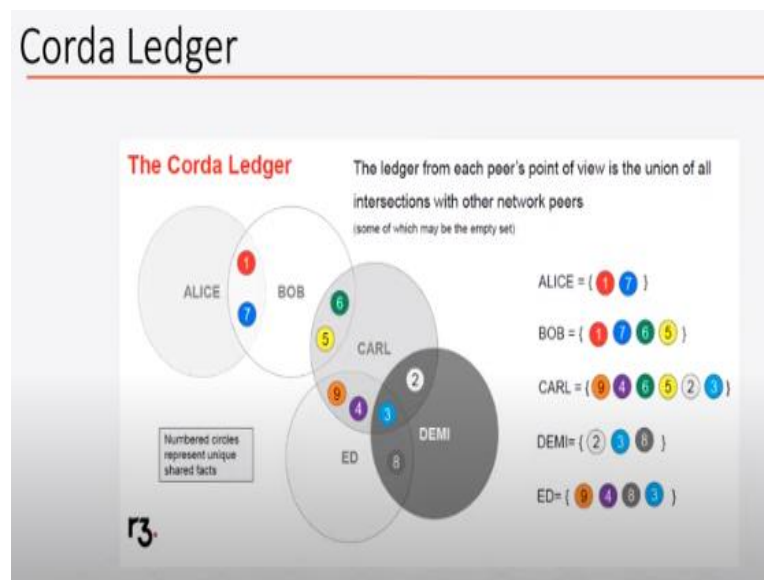


So here is an example of a State object representing a cash claim. Remember cash claim is something like Bank A says that on so and so date I this guy will have to give me this amount of cash, right. So here is an example that there are two parties. Here is maybe a bank and this may be a shipping company. And what it says is that, so it has a contract code, which is basically a programming is a program and here is the legal prose.

So the contract has an issuer and owner. So issuer is the one that issues the cash claim and the owner of the cash claim and the so for example, the shipping company is the issuer in this case. And the owner is the bank. And the issuer has to pay certain quantity to the owner of the cash claim and then it can be done any time it demands, right. So this is some borrowing mechanism by which this is done.

And this contract has multiple different functions like for example, there will be a contract issue function by which a contract can be issued to a party and also pay function which would have may have multiple different rules, maybe partial payment, maybe total payment and so on. So as we said before the so this is the state object at the beginning and then, as the state changes, the state object will change.

(Refer Slide Time: 48:59)



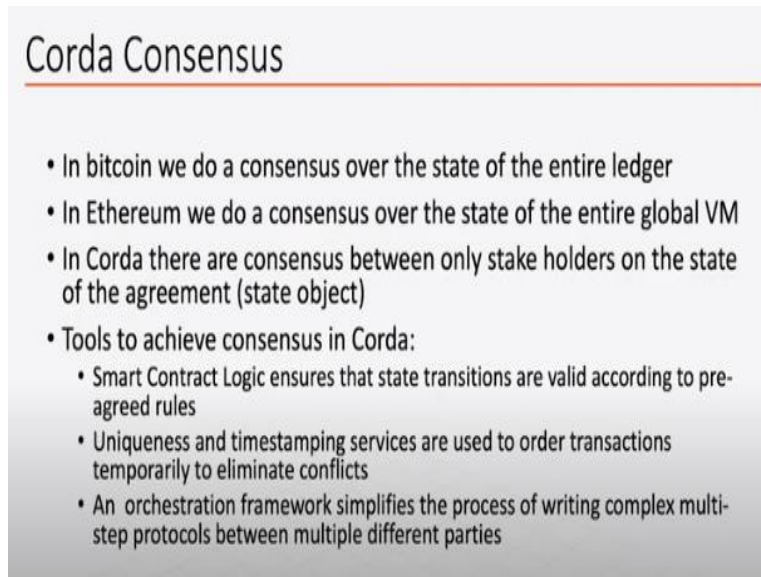
Now here is a pictorial representation of Corda ledger. So here, multiple different this colored small balls, they represent state objects. So Alice for example, has two state objects, basically it is involved in two contracts 1 and 7. Bob is the is also party to 1 and 7 with Alice. But it is also party to 5 and 6 with Carl. But Carl is actually quite active it seems and it has a 2 and 3 that it is involved with Demi and it is also involved with Ed on 3, 4 and 9.

And Ed is involved with Demi on 3 and 8. So 3 is certainly a tripartite agreement. Whereas 4, 9 are bipartite between Ed and Carl. 2 is a bipartite agreement with Demi, but 3 is a tripartite between Demi, Ed, and Carl. And 8 is a bipartite agreement between Ed and Demi. Similarly, Alice and Bob has two different agreements. So this is how the this represents the way the ledger has all these state objects.

So this state object 1 and 7 is only visible to Alice and Bob and maybe a regulatory node. Similarly, 5 and 6 is only visible to Bob and Carl, but nobody else and maybe the regulator node. Whereas 3 is actually visible to all three Carl, Ed and Demi. But 8

is only visible to Ed and Demi. So this is the idea of having a global ledger with **very**, very you know customized visibility.

(Refer Slide Time: 51:08)



Corda Consensus

- In bitcoin we do a consensus over the state of the entire ledger
- In Ethereum we do a consensus over the state of the entire global VM
- In Corda there are consensus between only stake holders on the state of the agreement (state object)
- Tools to achieve consensus in Corda:
 - Smart Contract Logic ensures that state transitions are valid according to pre-agreed rules
 - Uniqueness and timestamping services are used to order transactions temporarily to eliminate conflicts
 - An orchestration framework simplifies the process of writing complex multi-step protocols between multiple different parties

So consensus in Corda. In bitcoin we do consensus over the state of the entire ledger. In Ethereum we do consensus over the state of the entire global virtual machine EVM. In Corda the consensus happens between the stakeholders and the state of the agreement. And the tools to achieve this consensus in Corda is, first of all the smart contract logic would ensure that the state transitions are valid according to pre-agreed rules.

Remember, the smart contract is part of the state object. And the two parties have already agreed on that smart contract as well because it agreed to the state object between the two. The uniqueness and timestamping services are used to order transactions temporarily to eliminate conflict. So their uniqueness service and timestamping service.

So timestamping service is invoked to timestamp transactions and uniqueness service we will have some more words on that later. The uniqueness service is the idea that I am going to have the transaction I am going to do consumes some state object and it produces new state objects. But if a state object has already been consumed by another transaction, then I am unnecessarily I am working on stale state object if I am also trying to invoke a transaction with that state object.

And therefore it should not be done. So that uniqueness service basically is used to ensure that whenever I invoke a transaction, my input state objects are unique in the sense that it has not occurred in an already happened transaction as inputs. Then the uniqueness is violated, which means that I which basically would mean that I am working on stale data.

Remember this is the same kind of idea that we have used in Hyperledger when we did the validation by looking at the timestamps on the read/write sets the read set of a transaction and checking against the timestamp or version number in that case, not exactly a timestamp, version number on the current database that the node has. And that is how it detects whether the transaction is on stale data or on current data and accordingly it invalidates a transaction.

So same kind of thing is done by uniqueness service. An orchestration framework simplifies the process of writing complex multi-step protocol between all these different parties which are involved in executing the smart contract then going to the uniqueness service and timestamping service and come back and complete the transaction. All these activities are part of a flow.

Every transaction needs to have invoke a flow. So there is a flow library for, so commonly occurring transactions can actually use that flow library to orchestrate all the activities and their sequencing needed for a transaction to happen and the state to be properly evolved.

(Refer Slide Time: 54:50)

Corda Transactions

- In Corda, updates are applied using transactions
 - Transactions consume existing state objects
 - Output new state objects
- Two aspects of consensus
 - Transaction validity: -- check all contract codes that executes the transactions runs successfully, all required signatures are valid, and any referenced transaction is valid
 - Transaction uniqueness: -- there exists no other transaction, over which a consensus was reached, and that consumes any of the same states as this transaction
- Parties can agree on transaction validity by independently running the same contract code and validation logic
- Consensus on Transaction uniqueness requires a predetermined observer

So Corda updates are applied using for the state updates happen to transactions. So transactions consume existing state objects and outputs new state objects. So we already discussed that. So two aspects of the consensus, one is the transaction validity. So I have to check all contract codes that executes the transaction runs successfully, all required signatures are valid and any reference transaction is valid, right.

So that is the validity. So that can be checked by the node that is executing it. But the uniqueness has to be checked through the uniqueness service. So there exist no other transaction over which a consensus was reached and that consumed any of the same states as this transaction. So this is what is called the uniqueness. And this has to be done through a uniqueness service.

So parties can agree on transaction validity by independently running the same contract code and validation logic. But consensus and uniqueness requires a predetermined observer which records all the transaction what inputs they actually use so that when I go and ask whether the transaction I am going to do my input has been already consumed has been already consumed by another transaction it can tell me.

(Refer Slide Time: 56:23)

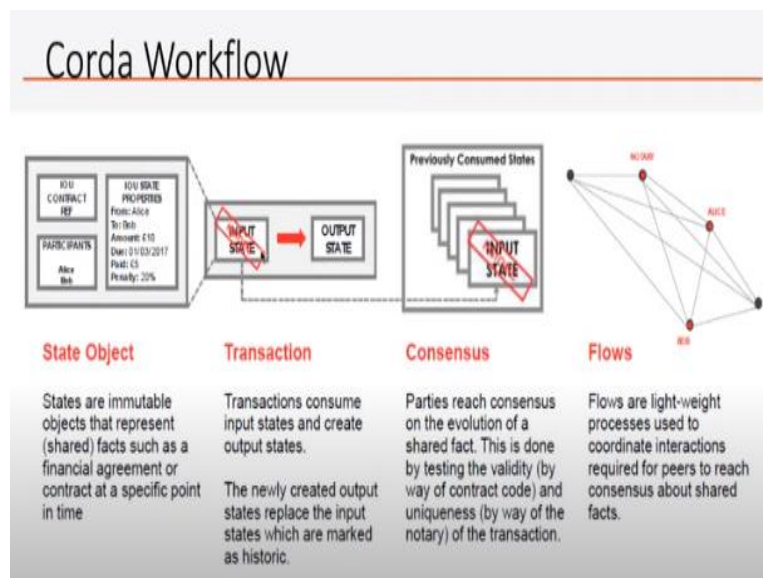
Uniqueness Consensus

- Corda has pluggable uniqueness service
 - Improves privacy, scalability, legal-system compatibility, and Algorithmic agility
- A single service may be composed of many mutually untrusting nodes coordinating via a Byzantine fault-tolerant algorithm or could be a single machine
- Uniqueness service does not check validity of transactions hence do not need to see full content of transactions -- privacy

And that uniqueness service could be actually the has to have an agreement and therefore it can be Byzantine fault tolerant or whatever I feel comfortable with in terms of the availability and fault tolerance etc., or malicious you know assumption about maliciousness node, maliciousness of some of the nodes involving in the uniqueness service. So uniqueness service does not check the validity of the transactions.

So they do not need to see the full content of the transaction, they only need to check the inputs. So therefore there is a some sort of privacy also available.

(Refer Slide Time: 57:12)



So here is a Corda workflow. So here is a state object. Let us say it is an IOU contract. IOU contract means, let us say Alice, it was issued by Alice to Bob, saying that I owe

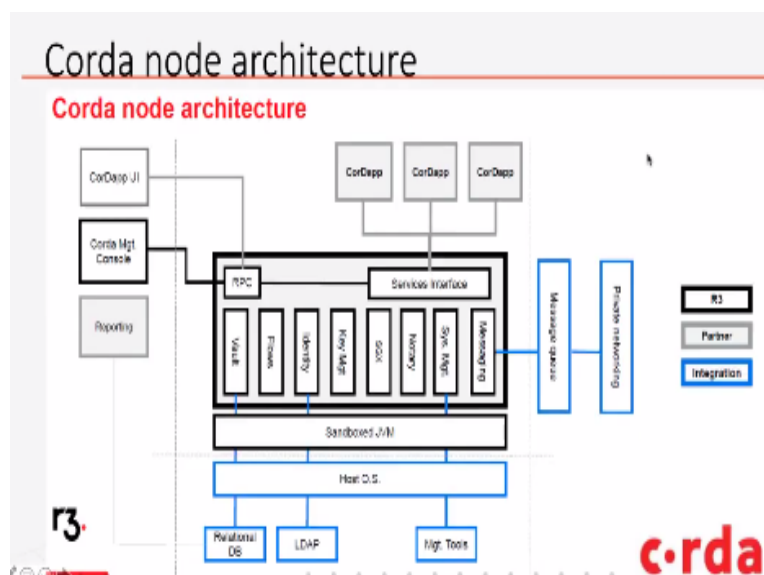
you \$10 10 pounds. And it is due on so and so date and Kevin state is that 5% has been paid, \$5 has been paid, and there is a penalty of 20% maybe for paying it halfway or something. I do not know what the penalty is for and Alice and Bob are the participants.

So this is the input state of a transaction. And then a transaction will consume this input state and let us say now Bob wants to pay Alice wants to pay another 5 pounds, but maybe she has to add the 5% 20% penalty. So she has to pay maybe \$7 to complete or discharge this contract. So the anyway so whatever the transaction is there will be an output state. And then it goes to the service, uniqueness service.

Uniqueness service has all the previous input states, which are called which are marked historic. So consensus will look that up and tell you that whether it is unique or not, because if Alice has already paid Bob once then invoking a transaction in which Bob is going to get paid again, may not be the right thing. Or even let us say Alice then bought 3 pounds at some point through a transaction.

Now she owes only 2 more pounds. But if I use this as my transaction input then I will think that Alice needs to pay me 5 pounds. So that has to be resolved through this uniqueness service. And that is how this so this entire flow happens like this. So there is Bob Alice, there may be a notary and there may be uniqueness service there may be other services.

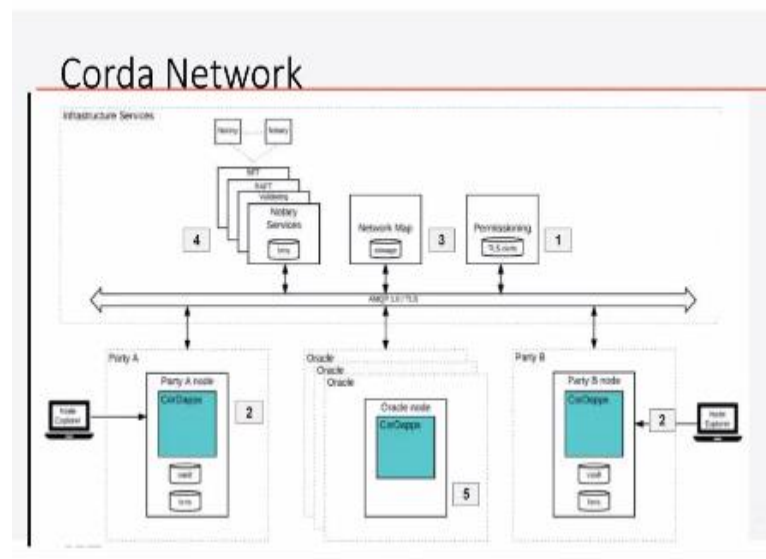
(Refer Slide Time: 59:20)



So Corda node architecture is, so code applications distributed applications written for Corda are called CorDapp. And Corda apps basically run on top of the nodes. And these nodes have various things like vaults. They have the state information, they have flows, they have an identity service, key management service. They have SGX for doing, you know crypto computation.

They have notary and system management functionalities and messaging, right. So they use a messaging service called AQMS. So which is actually a queuing based messaging service. And this there is a sandbox JVM on which all these things execute. And the host operating system may have other things like a database, the name service or some kind of LDAP, etc., etc. So this is the node architecture.

(Refer Slide Time: 1:00:30)



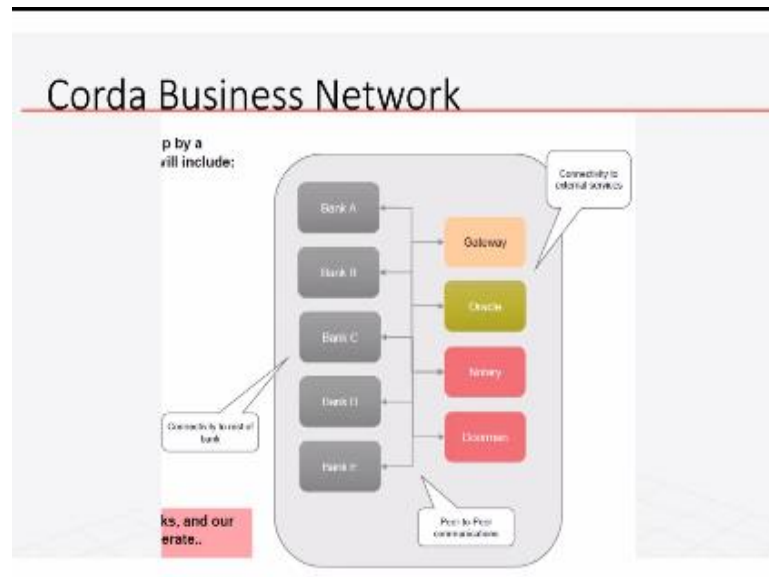
And here is you can see that this let us say you have multiple nodes. So a party A, party B. And then there is notary services, which may be based on BFT or raft or maybe just something. Then we have the network map. So network map is required for querying how to reach another node when one node wants to interact with another node. And there is a permissioning service which is basically the digital certificate service.

And then there is this message messaging system or AMQP 1.0. And then here are the Corda applications running on the various components inside the node like vaults and other things. And there is another concept called Oracle. In Corda Oracle basically is

same as similar as Oracle in Ethereum, where we have seen that Ethereum Oracles basically allows you to store real world information like a stock price etc.

And persist it so that it becomes deterministic to the blockchain and that is the Oracle service.

(Refer Slide Time: 1:02:00)



So that basically is about Corda that I want to talk about. Of course, from what we discussed, you cannot start tomorrow writing Corda applications. But it is not that difficult. So you could actually go to the Corda website and download some already canned examples and start installing and building stuff so it is not very difficult. So we can you can actually try. So this is sort of the Corda business network.

We have various entities in the network with peer to peer communication through that AQMP messaging service. And all these different things like Oracle Notary, Gateway, Doorman, etc., are concepts that are for giving digital certificates, doing Oracle service, doing uniqueness service etc. So this is more or less so but you see that in this case, this is so different from Ethereum or bitcoin blockchain.

This has come very far from bitcoin or Ethereum blockchain. Here the nodes have very different well defined purposes as compared to bitcoin and blockchain where every node is having the same type of responsibilities or same set of abilities to participate. Of course, computational power might make that little bit curtailed. But at least that is the theoretically every node can do everything.

In the Hyperledger we saw that there are special nodes for peers, there are special nodes for clients, they have special nodes for ordering service etc., but still there is not this kind of custom activities that we are seeing in this one. So this is very customized for financial type of financial, legal financial contract creation, execution and corresponding, you know various say integrity, security etc., for these financial institutions to participate in shared activities and execution of their financial obligations under the agreements etc.

So that completes our seventh week and the next week, the last week we will try to tie up the things that we talked about, talk about some applications and then give you some certain insights if you like, and that is how we will complete the course and we will see you next week.