

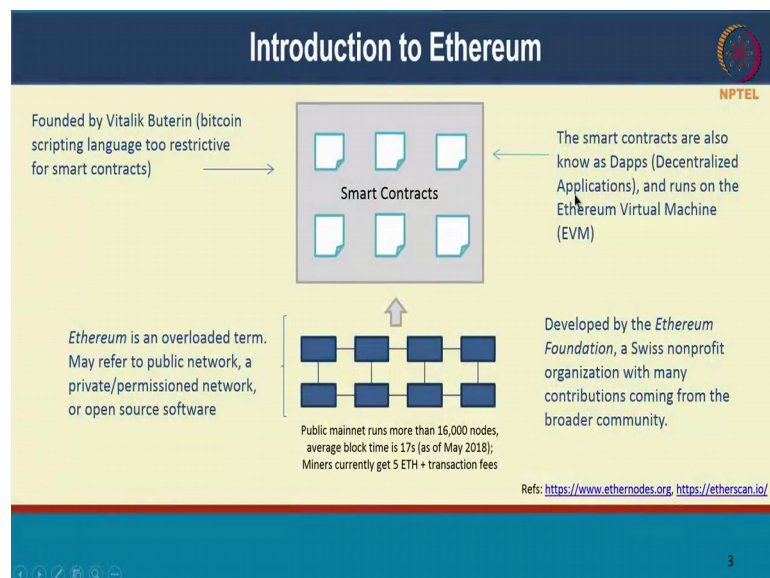
Blockchains Architecture, Design and Use Cases
Prof. Sandip Chakraborty
Prof. Praveen Jayachandran
Department of Computer Science and Engineering
IBM Research.
Indian Institute of Technology, Kharagpur

Lecture – 56
Comparing Ecosystems – Ethereum

Hello everyone, welcome back we are now into the last week of lectures for this course and I am very glad that you have stuck on with this course for this long and I am hoping that you are learning a good bit from this course. So, this week is going, to be about other alternate platforms. So, we have looked at hyper ledger fabric and hyper ledger composer earlier in this course, but we are going to look at a few other platforms, this week we are going to look at specifically Ethereum, quorum and corda.

So, I am not going to spend too much time on these, just one lecture each corda will spend a little more time, 2 lectures on corda. So, we look at some of these platforms, some of their pros and cons different design way ways in which they are defining or designing primitives for smart contract use and we will see what their, what some of that advantages disadvantages are.

(Refer Slide Time: 01:04)



The slide, titled "Introduction to Ethereum", features a central diagram and several text boxes. The diagram shows a central box labeled "Smart Contracts" containing six document icons. To its left, text states: "Founded by Vitalik Buterin (bitcoin scripting language too restrictive for smart contracts)". To its right, text states: "The smart contracts are also know as Dapps (Decentralized Applications), and runs on the Ethereum Virtual Machine (EVM)". Below the "Smart Contracts" box is a network diagram of eight nodes connected in a grid, with an upward arrow pointing to the "Smart Contracts" box. Text below the network diagram reads: "Public mainnet runs more than 16,000 nodes, average block time is 17s (as of May 2018); Miners currently get 5 ETH + transaction fees". To the left of the network diagram, text states: "Ethereum is an overloaded term. May refer to public network, a private/permissioned network, or open source software". To the right, text states: "Developed by the Ethereum Foundation, a Swiss nonprofit organization with many contributions coming from the broader community." At the bottom right, a reference is given: "Refs: <https://www.ethernodes.org>, <https://etherscan.io/>". The slide also includes an NPTEL logo in the top right corner and a navigation bar at the bottom.

So, this lecture is we are going to talk about Ethereum. So, Ethereum is is one of the hottest blockchain platforms. Today, it was founded by Vitalik Buterin and it was back in 2013 and he felt that the bitcoin scripting language was way to restrictive for smart contracts. So, it was primarily intended as a means for executing smart contracts in a decentralized fashion. So, that is how it started and in even today that is one of the biggest use cases, uses of Ethereum.

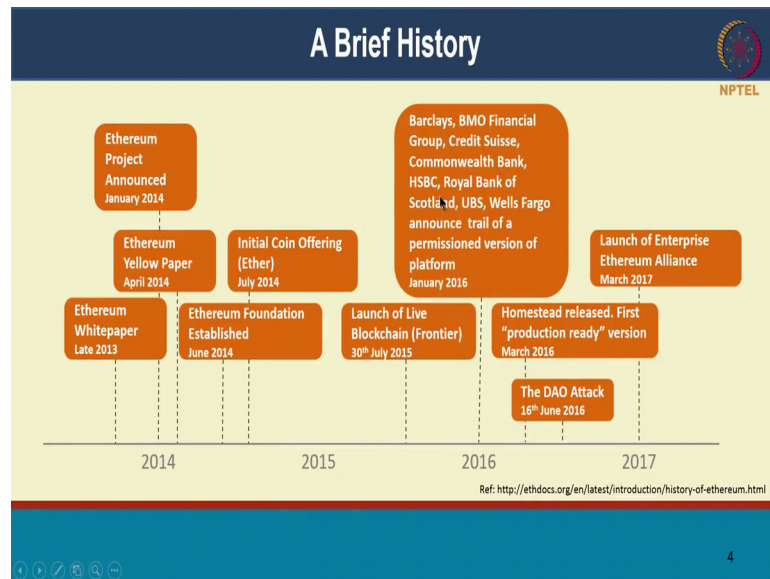
The smart contracts are known as daps or decentralized application and they run on a Ethereum virtual machine. So, we will see some details of this, Ethereum itself is an overloaded term right. So, it refers to the public network is similar to bitcoin, there is a public Ethereum network that runs and there is a set of peers that run this network. It can also be a private permission version of Ethereum.

So, it is possible to take the Ethereum codebase, run it as a private network or a permission network with just a few nodes. So, that is also possible and Ethereum is referred to it to in that way also, not just a public network, it can also be a permission network and it is also the open source software behind Ethereum is also referred to as Ethereum. So, it is really over overloaded in that sense.

It does have a native crypto currency called ether similar to bitcoin rate. So, bitcoin is also an overloaded overloaded term Ethereum has ether as native crypto currency and today as I checked just a few days back, there are the main net for Ethereum has over 16000 nodes and I am not sure, how many are really full nodes. But there are a or very large number of nodes, now running the public Ethereum network and the average block formation time is about 17 seconds, which is significantly better than bitcoin, which is about 10 minutes today. So, bit coin takes, 10 minutes for each block Ethereum takes roughly about 15 to 17 seconds and miners today get 5, 5 ether plus all the transaction fees for the transactions in the block the mine.

So, it also uses proof of work similar to bitcoin, but we will talk about some of that in due course um. So, there is a reward for mining which is 5 ether and the miner who finds the proof of work challenges solves the proof of work challenge first gets also the transaction fee.

(Refer Slide Time: 03:48)



So now, this is Ethereum is being developed by Ethereum foundation, which is a Swiss non profit organization, but it has contributions coming from a much broader community. So, there is a big community behind Ethereum that is contributing to that code.

So, just a very brief history of how Ethereum has evolved? So, the Ethereum white paper came out in late 2013 and the project itself was announced in 20 2014 then there was a yellow paper with more details that came out in mid 2014 and then the Ethereum foundation was established.

And in July 2014 was the first initial coin offering. So, Ethereum was perhaps the you are probably hearing about initial coin offerings ICOS, they are they are really a rage today with people starting us creating a start up and coming out with their own crypto currencies or coins for various blockchain use cases and Ethereum or ether was the first initial coin offering back in back in July 2014. So, it had this good technical backing yellow paper, white paper and Ethereum foundation before they came out with the ICO right.

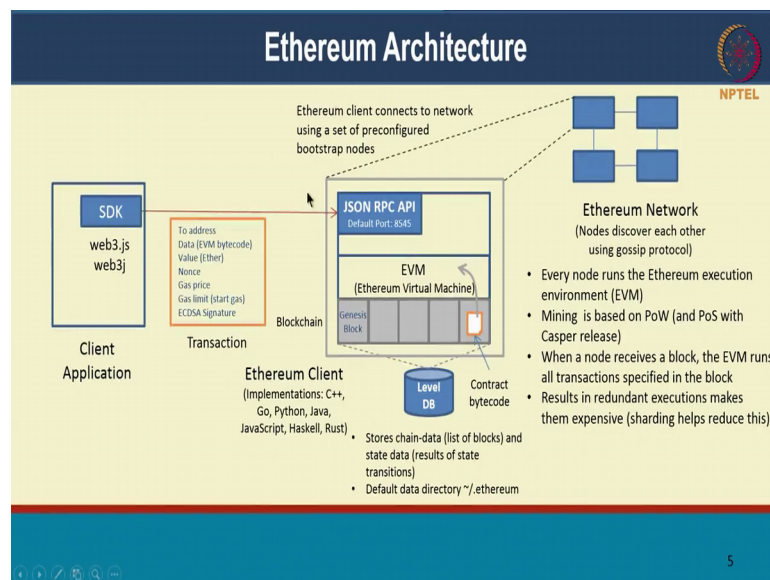
Ah the live blockchain came out a year later and since then, this it is actually seen a rapid growth in the network itself in the valuation of ether. It is really grown multi folds and fold since then and several large financial institutions have looked at variants of ether for permissioned versions for enterprise use cases and that is also become quite popular and we will look at one such variant in the next lecture, which is quorum and we look at some of the properties in in quorum, first production ready version was released in march

2016. So, roughly a little over 2 years back, there have been various attacks on I would not say, Ethereum itself, the Ethereum itself has been fairly solid the blockchain platform has been solid, but on the smart contracts running on etheria, Ethereum that have been various attacks on them. The most notable of them was the dau attack, which caused several millions of dollars to be siphoned off from the smart contract that held those that value and as a result that actually ended up in a hard fork of Ethereum.

So, people actually ended up reversing certain blocks in the blockchain, going back in the past and then creating a fork from there to do not have this hacker take away all their money right. So, that was a very momentous occasion, where people actually undid something on the blockchain, the community decided they were going to go back.

But there have been a lot of security attacks on Ethereum smart contracts over the last couple of years and last year in march 2017, the enterprise Ethereum alliance was formed and this was again looking at how Ethereum can be used for enterprise use cases and how some of the shortcomings of Ethereum for enterprise use can be overcome.

(Refer Slide Time: 06:46)



So, the just a high level view of the Ethereum architecture, what are some of the click on strux. So, the Ethereum client a trans so every Ethereum node runs an Ethereum virtual machine and all the smart contracts run within this virtual machine and clients can connect to this the Ethereum client applications can connect through the Ethereum client

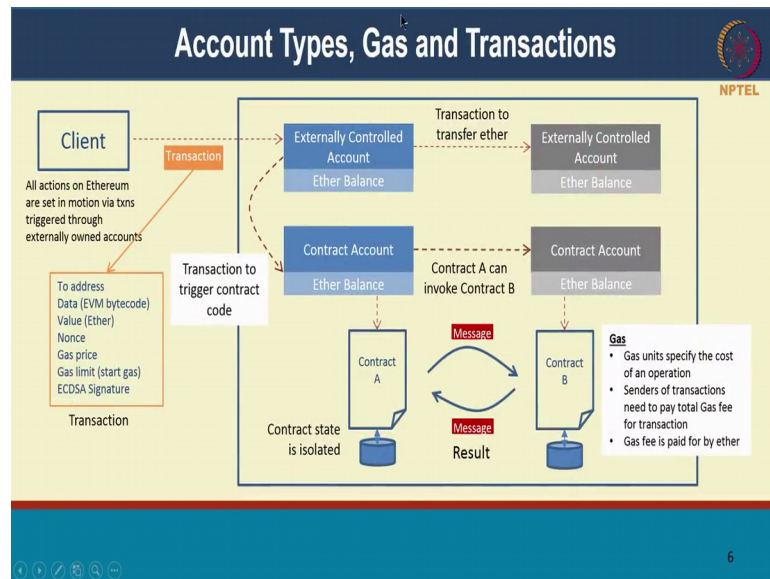
using an SDK there is a web 3 j SDK for it, transactions have a particular format of course, it has a 2 address who are you transacting this with.

So, this just like bitcoin, it is a a particular recipient is mentioned, there is data pertaining to the smart contracts thats also mentioned there, what is the value of ether, you are going to transfer to the recipient. There is a nonce and there is a notion of gas, which we she see shortly and of course, there is a signature by the transactor. So, the user who is sending this transaction is going to sign the transaction and send it to blockchain.

So, note that Ethereum actually does 2 functions, it is actually going to transfer value or ether in some sense and it is also going to execute. It can also execute a smart contract function and that smart contract can hold data on the blockchain and there is a a Ethereum network of many such nodes, all interacting with each other and these nodes are the ones, which did the which runs consensus and determine, what block of transactions should be added next to the blockchain. And this consensus is through proof of work similar to how bitcoin did it there are alternatives being considered as well, but for now proof of work is, what is being used by the Ethereum blockchain.

And there is a proof of stake algorithm, that is going to come out later in the casper release, when a node receives a block, what the client with Ethereum client will do is, it is going to execute all the transactions in the block in sequence and it will then update it is ledger. Once the block has once it knows their block has been added, it is possible that because, it is getting this block from multiple people redundant executions are possible, but they are hoping that the notion of sharding will help reduce this. So, we look at sharding right at the end of this lecture very briefly.

(Refer Slide Time: 09:15)



So the underlying database that Ethereum uses is a level DB. So, it just has a basic key value store available to smart contracts to store a, state information and the contract byte code is also stored on chain. The list of blocks and the state data, this is similar to fabric both the list of blocks and the state data is stored on the blockchain.

So now, let us look at a few notions right, what are accounts of what is gas and the notion of transactions in in Ethereum? Right. So, Ethereum really has 2 kinds of counts, one is called an externally controlled account. So, this is like my I as a user of Ethereum, I can have a wallet on account and have a certain balance in that account. So, that Ethereum ether balance is held in that externally controlled account. So, there is a user who is who owns this account in some sense and from it is possible to transfer ether from one externally controlled account to another externally controlled account.

So, this transaction is pretty much just identical to what bitcoin does. So, you are going to have a certain set of inputs, a certain set of outputs and you are going to show that the inputs are all unspent. So, it is a very it is just is basically a UTXO model and you can transfer ether from one account to another account very similar to bitcoin.

But that is this one part of the story, the other kind of account that we have in Ethereum is a called a contract account. So, this is really for smart contracts. So, every smart contract will have a contract account and it will have an address very similar look, it is a similar looking address to an externally controlled account. So, both of them have addresses, but a contract account specifically refers to a smart contract and it does not

refer to a particular individual or an externally controlled account. and what is also possible is for transactions to both ; however, externally refer to an externally controlled account as well to a contract account.

So, you can, I can transfer some ethers from one account to another, I can also invoke a particular smart contract specified at a particular address and that smart contract will of course, run a particular function defined in the in the contract and that function can update state information on blockchain. So, that is the distinction from bitcoin itself, where it is just not bitcoins being transferred from person A to person B. In addition, you can have a smart contract function that operates on arbitrary state information and the state information is basically a key value store.

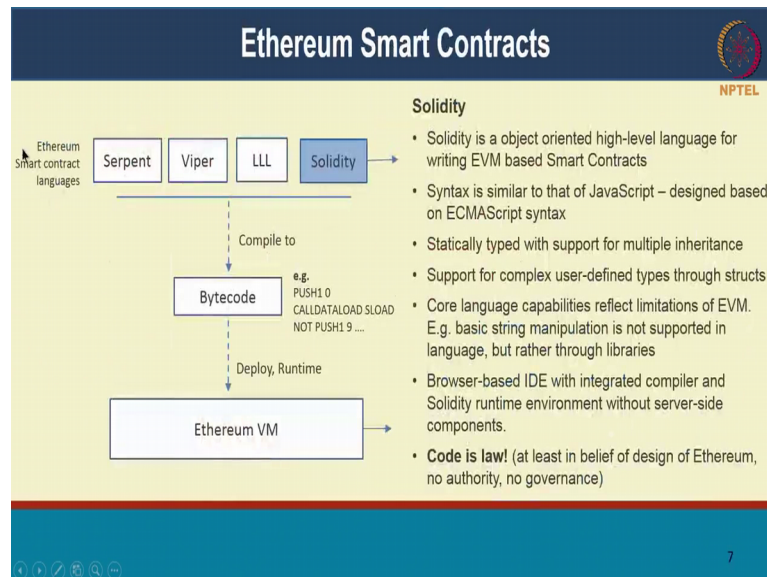
So, you can define your own keys and have values for those and apart from invoking one contract, these contracts can also invoke other contracts. So, similar to fabric where one chain code can invoke another chain code. One contract can call another contract and again there can be a transfer of balances between contracts as well.

Now, all of this invocation of transactions or invocation of smart contract functions, actually incurs a transaction cost and this is really to ensure that people are not, because there is a public permission less system, we want to ensure that people are not randomly spamming the network with invalid or wrong transactions right. So, people just do not want to take away, maybe it could be a denial of service attack also right, I could just randomly access transactions, access call functions and sub attach is a network right.

To prevent that Ethereum has the notion of gas. So, for each transaction you will have to spend some gas. So, this is the fuel for the transaction fuel and unless you pay in terms of this gas price unless you pay for the transaction, you cannot execute the transaction.

So, each transaction is also going to have a a particular gas price. So, I will say that for this invoking this transaction, this is the gas, I am going to pay and there is you can also set a limit, because you do not want your balance to be depleted like, what is the contract really runs for a long time. First for whatever reason and you lose a large amount of money right. So, you do not want to do that.

(Refer Slide Time: 13:29)



So, the gas has both a price and a limit and the gas fee is really paid using ether right, how much ever is the gas, there is a certain price. So, there is a mapping from gas to ether and you have to pay that much ether to invoke transaction. So, this is really the the gas price is really borne by the sender of the transaction.

Looking at the smart contracts themselves Ethereum allows or has multiple languages in which these smart contracts can be written with many languages, serpent, viper, etcetera but the one that is most popular is solidity. Solidity is an object oriented high level language, it is very similar to to Java script and it is executed inside an Ethereum virtual machine; so, any of these languages, what they do? Is they compiled into a standardised bytecode? So, this looks something like assembly right.

So, pushed 10 and and it you could load data things like that right. So, it is really close to something, it is really bytecode and this bytecode runs in a Ethereum virtual machine. So, this bytecode is what is deployed onto the blockchain network and this compilation is is kind of outside the syntax as I said is similar to Java script. If you if you familiar with Java script, it is statically typed and it supports multiple inheritance. Yeah it also supports complex user data types through structs. So, all this is very standard object oriented programming principles. The core languages that have been defined have all been defined for you are keeping in mind with the Ethereum virtual machine.

And Ethereum virtual machine itself is a bit locked down in some sense. So, you can only perform a certain set of things, we will come to it in the next slide. The Ethereum

virtual machine allows only has certain limitations and and deliberately, I has certain limitations and the languages also reflect that right there are only few things you can do. For instance, even basic string manipulation is not supported, but it is only supported through libraries.

So, there will be other libraries that you can use for string manipulation, but the languages themselves to not support it. There is a, browser based IDE or solidity solidity and it has that browser has an integrated compiler and the solidity runtime environment can run without any server side competence.

An important very important philosophical notion for Ethereum smart contracts is that the belief is that code is really long right, whatever is in the code is the right thing right. So, there is no auditability or governance, there is no other authority that is going to govern, this piece of code right. Whatever this piece of code is is going to be executed in a decentralized fashion.


And many a time this philosophical notion, while it is create from a notion of philosophy that, there is no authority, there is no governance, while that is great to hear because; it is completely decentralized that way. In reality that becomes, problematic because what if a hacker? As has happened in the past, if a hacker comes in and steals data because, your smart contract had a vulnerability in it right, it allowed the hacker for steal the data.

Then what do you call it? Right, the code if you say the code is law then the fact that the hacker leveraged a vulnerability, the vulnerability is it then a feature right. So, that becomes a philosophical question then, the problem is that there is a specification, typically of a problem that you want to solve and there is code that tries to write a solution to that specification and typically that code is written by a human and it is fallible right.

So, people you could make mistakes when tried writing code, there are bugs for instance, in code that you write and smart contracts are going to be no different. There will probably be bugs in smart contracts and if code is really law, then you are saying buggy code is going to be believed and enforced upon all the time.

(Refer Slide Time: 17:18)

Ethereum Virtual Machine



- (quasi) Turing complete 256bit Virtual Machine
 - Computation is intrinsically bounded through a parameter, gas
- Allows execution of EVM Bytecode
- JIT interpreter that compiles the byte code into manageable data types and structures.
- Define ~70 OPCODEs
- Gas defined for each OPCODE (e.g., IF, AND, MULTIPLY, SEND), paid for using ether
- Designed for simple operations: "Roughly, a good heuristic to use is that you will not be able to do anything on the EVM that you cannot do on a smartphone from 1999."

8

So, that be causes problems and also cause for a lot of the debates, that have happened in the community whether they should really be the case or whether there should be some governance around it?

Moving on the eth coming to the Ethereum virtual machine itself, it does support turing complete language it is a 256 bit virtual machine and the computation is intrinsically bounded through gas. So, I avoid a certain amount of gas and if the gas is consumed then the contract execution, the smart contract execution will stop right.

So, that way if a transaction or a smart contract had unbounded execution for some reason, you provided certain inputs on which there was unbounded execution, by design of having this gas and having a limit on it, we will ensure that the contract does not run indefinitely. So, this allows termination to be definitive within a certain period of time and it allows the execution within the EVM bytecode.

So, the EVM defines about 70 OPCODEs and your, you are limited by that. So, that is only this the sets that are that is aloft and this these set of OPCODEs are interpreted by an interpreter and for each OPCODE each of these 70 OPCODEs, there is a specific gas value that is defined and this gas value is set based on, in some sense roughly, the difficulty of that operation and what are some of the operations it could be? For instance, arithmetic functions like add, subtract, multiply, divide. It could be logical operations like and, or kind of logical operators, it could be conditional statements like if then else, it can be send right, I can send ether to another address.

So, all of these are OPCODEs and for each OPCODE based on the computational complexity on a traditional processor, there are gas limits that are set. So, you could say that ok, if I am going to make up numbers, if cos 40 gas or and multiply operator cos 30 gas right.

So, based on the complexity of the operations gas is specified and that gas. So, based on all the operations in transactions, we say a transaction has 10 such operations. You will add up all the gas for all the operations and that is the gas necessary to invoke that transaction and that gas has to be paid for using ether, which is the inbuilt crypto currency in Ethereum and if you want to get a sense of what kind of operations, that are left think of an old smartphone right back in the early 2000s perhaps right that those old phones, whatever computation you could do on those old phones is what you could do on an EVM.

(Refer Slide Time: 20:20)

```
pragma solidity ^0.4.4;
import "KVSLock.sol";

contract CustomerManager {
    KVSLock private kvStorage;

    struct Customer {
        bytes32 customerId;
        bytes32 name;
    }
    mapping(bytes32 => Customer) customers;

    modifier customerNotExists(bytes32 customerId) {
        if (hasCustomerId(customerId)) throw;
        _;
    }

    event NewCustomer(bytes32 indexed customerId, bytes32 name);

    function createNewCustomer(bytes32 customerId, bytes32 name)
        customerNotExists(customerId) {
        //...
        customers[customerId] = Customer(customerId, name);
        NewCustomer(customerId, name);
    }

    function getCustomer(bytes32 customerId) constant
        returns (bytes32 name, bytes32 customerId) {}
}
```

For instance, I would really cringe away from doing like crop complex crypto functions inside my spot contract, in contrast with fabric we are providing you the ability to provide crypto functions within the chain call itself whereas, an EVM that will be very hard to do.

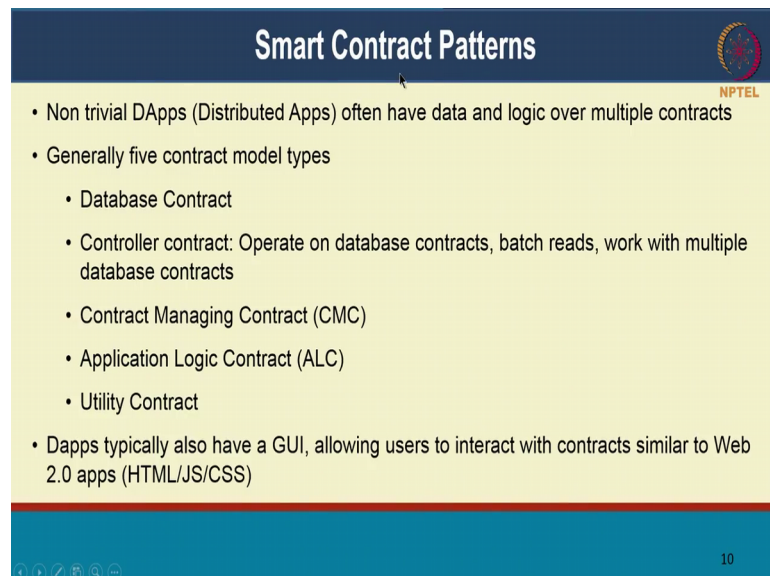
So here, is an example of a basic solidity code. So, I am not going to go into details, but just to give you a feel for what this looks like. So, standard object oriented programming similar to Java script right, a contract is defined you can import other contracts is also.

So, you can reference other smart contracts. So in this case, we are actually importing a key KV store contract.

You can have transaction events. So, these events will be emitted when the transaction executes and. So, if you this is the definition of the event and within the function, this is a smart contract function, that is going to get invoked in a transaction you can also fire these transactions. So, at different points of the function execution, you can fire transaction events.

Then you could have modifier functions, these ensure that execution proceeds only if a precondition is met, you can have the like a constant keyword added. So, this is for read only transactions. So, this is for like a query query operation and you can have return types right. So, this is roughly the structure of a of a facility contract. So there is a contract, it can have multiple functions, you are defining some structs or data objects. You can refer to other contracts; you can invoke other contracts all that are possible. So in some sense, this is or it is to curing complete, but the the sophistication is limited, it is deliberately limited because you do not want to allow non deterministic execution, you want termination to be guaranteed.

(Refer Slide Time: 22:04)



The slide is titled "Smart Contract Patterns" and features the NPTEL logo in the top right corner. The main content is a bulleted list:

- Non trivial DApps (Distributed Apps) often have data and logic over multiple contracts
- Generally five contract model types
 - Database Contract
 - Controller contract: Operate on database contracts, batch reads, work with multiple database contracts
 - Contract Managing Contract (CMC)
 - Application Logic Contract (ALC)
 - Utility Contract
- Dapps typically also have a GUI, allowing users to interact with contracts similar to Web 2.0 apps (HTML/JS/CSS)

At the bottom right of the slide, the number "10" is displayed.

So because, you want some of those properties it has been deliberately restricted ok.

What are some of the patterns? Right the non trivial decentralized, you should read decentralized apps by the way, it should not just not just it is actually decentralized not distributed. The data and logic can actually span multiple contracts typically, there could be 5 model types, one is a database contract this is going to have all the data objects on there. One is a control controller contract, which operates on the database contracts. So, this is this will be the controller on, when or what to be written on to the ledger? And one controller contract can work with multiple data database contracts, just for separation and there is a contract managing contracts. So, this manages the contract itself, there is an application logic contract and that could be utility functions defined as a separate contract.

So, collectively all of these contracts could be a decentralized app, decentralized app in Ethereum and the decentralized apps typically, also have a graphical user interface and this allows users even those who are not so familiar with blockchain or the intricacies of smart contracts.

(Refer Slide Time: 23:29)

The slide is titled "Additional Capabilities" and features the NPTEL logo in the top right corner. It lists several key capabilities of Ethereum:

- Proof of stake to save energy consumption, Casper release
- Swarm file storage
 - Decentralized file storage built into Ethereum
 - Uses contracts and ether to encourage cooperation among nodes
- Whisper messaging for secure private communications
 - Messages are encrypted and announced to the network off-chain; person-to-person and contract-to-contract
 - Inherently supports unicast, multicast and broadcast
 - Ability to use masks/filters to narrow down topics of interest without giving away what topic is being sought
 - High latency, low bandwidth
- Raiden state channels – similar to bitcoin lightning network / payment channels; off-chain transactions with proof / dispute resolution on-chain
- Sharding transaction groups – Nodes are partitioned so that nodes don't have to store and process entire blockchain (still in development)

At the bottom of the slide, there are navigation icons and the number "11".

The GUI will allow them to interact with the with the contracts, very similar to how web 2.0 apps to right. So, it gives you a HTML interface or a JS, CSS interface for interacting with smart contracts.

Ah there are a lot of additional capabilities, that are being added or have been added and are being added to Ethereum, there is a lot of innovation going on in the Ethereum

blockchain itself. So first as I mentioned, this is proof of stake for saving energy consumption. So, with Ethereum the, it currently it does use proof of work which is it consumes lots of a lot of energy, because you are doing a lot of hash computations, instead there is a proof of stake and alternative algorithm that consumes lesser energy. So, that is coming out later, it is not yet there with Ethereum, while the block formation time is significant low. This is significantly lower, the throughput of Ethereum is still fairly low, few 10s of transactions per second ok.

What are some of the additional capabilities right. So, the first one is there is a swarm file storage that is basically allowing decentralized file storage to be built inside Ethereum and what it does? Is it uses contracts, themselves to encourage cooperation among nodes. So let us say, I want to store a particular file, I can store that in Ethereum and I can pay the nodes, who are storing it. So, the contracts themselves will manage both the storage as well as the payment for that story. So, that is that is being integrated into Ethereum.

The other next one is whisper messaging. So, this allows secure private communications amongst maybe a subset of entities right. So, it inherently supports unicast, multicast and broadcast and what it does? It is actually encrypts messages, where is it simple in concept right it is just encrypting messages and it is announced to the network off chain right, it allows contract to contract communication apart from this person to person communication.

So, whisper has been integrated into the Ethereum protocol, but it does because of all the encryption and messaging complexity, it does have high latency and and low bandwidth right. So, that is important to consider, it is also possible to use masks and filters to narrow down the topics of interest. So, each of these messages you can put them into topics right and you can probably, just you know filter these messages based on topics that you are interested in and for any particular user or a client, you you would not be giving away, what topics you are interested in what topics you are seeking to find out? Ok. So, that is on on the secure messaging (Refer Time: 26:01).

So, raiden is a very interesting concept, built on top of Ethereum, it gives you a notion of state channels. So, the idea here is with blockchain being rather slow, commit latency is fairly high with bitcoin it is 10 minutes with Ethereum is 17 seconds or 15 seconds, but

that is still the significant period of time. Today our digital transactions go through much quicker than that right.

So, what state channels allows you to do? It is really a payment network, but what it allows you to do? Is just have some proof on blockchain and actual payments happen on a separate off chain channel right? So, this is not the similar to channels in fabric, it is a different notion, where 2 or more parties can get into a particular channel. They can exchange messages between each other to exchange value. So, they could exchange for instance, ether amongst themselves, but they do not go to the blockchain for every one of those transactions to just give you a very vague or a very high level example let us say, I am going to go to a coffee shop, I go there every day, every morning I buy coffee. If I were to for instance, pay for that coffee and crypto currency then every day, I have to make that transaction and each of those transactions has to be recorded on blockchain.

But typically, the coffee shop and I are we kind of trust each other right, we are known we are known entities unless, there is a dispute, we do not want to overburden ourselves with going to the blockchain every time. So, what I could do? Is I I could say, I have 50 rupees in my or whatever right then let us say, I have 500 rupees in my account. I can go to the coffee shop, I can just exchange a message saying here, I have send a message to you saying, I am purchasing a coffee from you here is here is 10 rupees right, I can do that over 100 days and collectively submit that whole set of transactions onto the blockchain.

If for instance, I I try to run away with without paying for my coffee then the coffee shop will be able to go to the blockchain, prove to them that I have actually cheated. So, there are there will be penalties invoked in impost and that the sense is this incentivizes either of us from cheating. So, that is really a a payment channel that is outside the blockchain itself, there is only some minimal information recorded on the blockchain, that will be useful for dispute resolution or service a proof of transactions, but nothing more right.

So it really allows, you to batch transactions together and send them and commit them on blockchain. It is the, a very interesting notion to enhance the performance of blockchain platforms. So, you should definitely check that out the rate, rate in state channels. This is a similar to bitcoin the lightning network in bitcoin. So, you should you should check that out.

So, the next capability is that of sharding or having transaction groups. So, what is possible? it is this one is still under implementation, it is not yet out there with Ethereum, nodes are nodes can be partitioned into multiple transaction groups and each transaction group only processes certain transactions or only store certain amount of data. So, that is really similar to the sharding notion in databases, where that database can be really be a distributed database and with sharding each node in the database stores only certain partition of the data right.

Similarly, with block chains also they are introducing sharding, where certain pairs hold certain data and I do not have to hold all the data in the network, there are of course, you do this at reduced security guarantee on on how many nodes need to be attacked for you to take over at least that transaction group? So, you have to take over smaller there is a lesser security guarantee there, but there are ways to dynamically figure out, what is the trust level in the network? So, there is a lot of innovation going on in this space right, I want to figure out what is the trust level? And based on that, I create my transaction groups I do the sharding based on that. So, there are some interesting concepts going on there.

So, Ethereum does have it is own limitations, it has a nice set of features it is great for smart contracts for transactions, that involve exchange of ether between parties, but it does have limitations in it is in it is smart contract logic right, smart contracts are the state is bound to a specific contract. So, the state and the logic, if they are in the same contract it or a same contracted, it reduces maintainability, what if I want to update the contract logic right?

Then one has to manually copy over the state from the previous contract version to the new contract version. So, all that becomes problematic, the some extent, the separate separation of state contracts and the logic contracts that I talked about in dabs, which is typically followed it helps it helps the mitigate this. So, that is a design pattern, that people have found out that they want to separate the logic from the state, that way the logic can be changed and the state can be can be reused across

As I mentioned, it does not support string manipulation, it does not support returning complex data types. So, only simple data types can be returned and that becomes sometimes a cause of concern, because it does not give you the flexibility you need as a

developer. Especially working with an object oriented programming language and functions cannot contain more than 16 parameters and return values.

(Refer Slide Time: 32:22)

Limitations

- Smart contract state is bound to specific contract. Storing state along with business logic reduces maintainability, since every contract 'update' requires state migration from old contract.
- Separation of state and application logic is used as a pattern to mitigate this
- Does not have native string manipulation (e.g. concatenation, find, split ...)
- Cannot return complex data types from functions, only primitives
- Functions cannot contain more than 16 parameters and returns values. Due to the lack of string handling and lack of complex type returns. we have to return multiple individual fields as results to queries. The limit of 16 returns means this model cannot scale.
- Debugging DApps is harder, since the EVM (Ethereum Virtual Machine) does not currently return meaningful error messages.

12

So, that is also tends to be a limitation, because you cannot have complex data types. So, you have to say, if you are if you do want to send that across, you have to send each element separately and there is a limit of 16 and that becomes a concern right between not being able to pass complex data types and between the fact that string date, string handling is not supported very well that becomes a limitation for a developer.

(Refer Slide Time: 32:38)

Inter-Contract Calls

- Cannot pass dynamically sized data (e.g. variable length strings, arrays) to functions called inside smart contracts.
- Calling one contract from another creates a new EVM instance for the called contract. The performance cost of this can quickly add up especially when storing complex types in a generic database contract

```
graph LR; CC[Customer Contract] -- uses --> KVC[KVStore Contract];
```

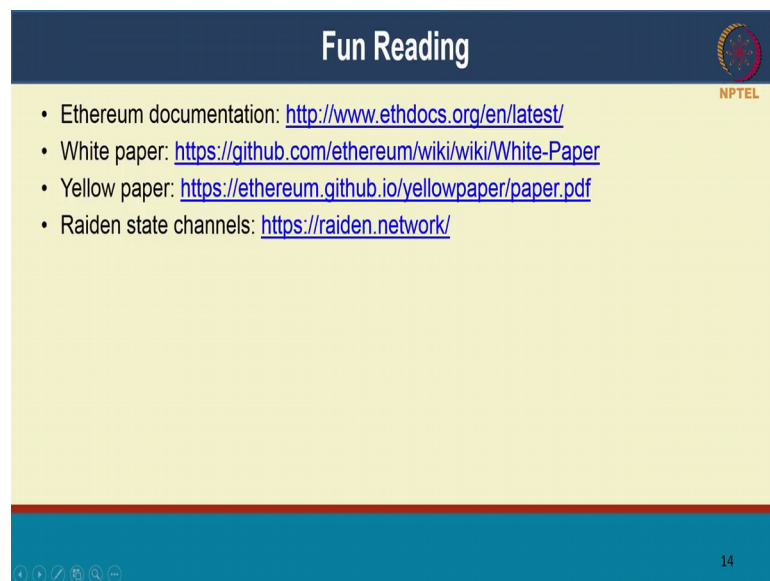
13

Debugging distributed application also tends to be harder since the EVM, today does not return very meaningful error message, it is not something at least I have personally felt using the using solidity on EVM.

And again for inter contract calls, when one contract invokes another contract again it is not possible to pass dynamically sized data. So, as to be fixed length strings and the functions calling one contract from another contract also creates a new EVM instance. So let us say, I have both of these contracts. So, contract A and contract B both of them are independently running in my in EVM instances.

Now let us say, if contract A calls contract B, it does not use the existing EVM instance. It actually creates a new altogether, new instance right. So, that way many such instances are created and if one contract is simultaneously called by many of these other contracts, then that becomes a performance bottleneck, because I am spawning so many of these instances redundant instances, I could potentially do with just 1 EVM instance.

(Refer Slide Time: 33:41)



The slide is titled "Fun Reading" and features a dark blue header with the NPTEL logo on the right. The main content area is light yellow and contains a bulleted list of links. At the bottom, there is a dark blue footer with navigation icons and the number 14.

- Ethereum documentation: <http://www.ethdocs.org/en/latest/>
- White paper: <https://github.com/ethereum/wiki/wiki/White-Paper>
- Yellow paper: <https://ethereum.github.io/yellowpaper/paper.pdf>
- Raiden state channels: <https://raiden.network/>

So, there is a performance penalty, when you are calling from you are you are making inter contract calls. So that is that is a high level overview of Ethereum, there is a good documentation that is available, there is a very large community using Ethereum today, there is a white paper and a yellow paper that, I will encourage you to read and I mentioned state channels are a very interesting notion.

So, I will I will you can check out the raiden network that is trying to improve the performance of the public Ethereum blockchain itself. Today the performance is not at a level acceptable for enterprise applications and they are doing some of these improvements to assess that, but the raiden is specifically for 2 party or mostly for 2 party contracts sorry, 2 party exchanges and mostly used as a payment exchange ok. So, that brings us to the end of the lecture on Ethereum, in the next lecture we will look at some of the Ethereum developer tools as well as as quorum right as a permissioned version of Ethereum.

Thanks a lot.