I have tried to organize what I am doing into sections , with interactions possible between sections to result in a potential end product .

1 Definitions

1.1 Blockchain

1.1.1 Why ?

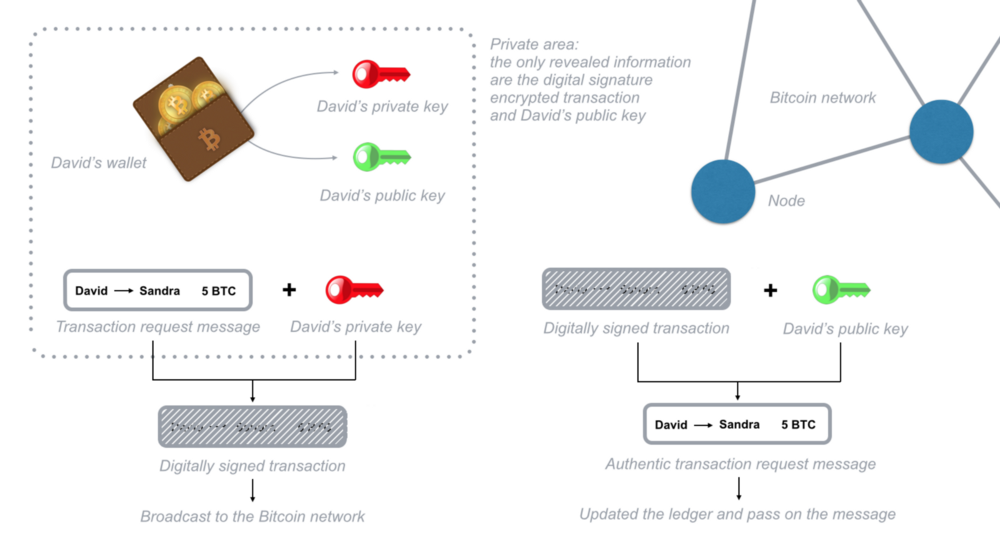
> When humanity started, trade was conducted using the barter system ( I give you 10 oranges , you give me 5 apples ) .Over time , humanity came up with better ideas - to use a piece of metal / paper as an intermediary for providing goods and services. The only reason this metal / paper had any value was because it was backed by a king or a bank, which is how it had any kind of value.   
> Currency had value because people trusted a central authority to back them up ( ie , provide an equitable amount of resources if the users took the currency to the king and asked for goods in exchange for the currency). Today , the same system continues.Most organizations operate on some level of trust that exists between users and an organization ( organization here can mean banks, governments , supermarkets etc ) .

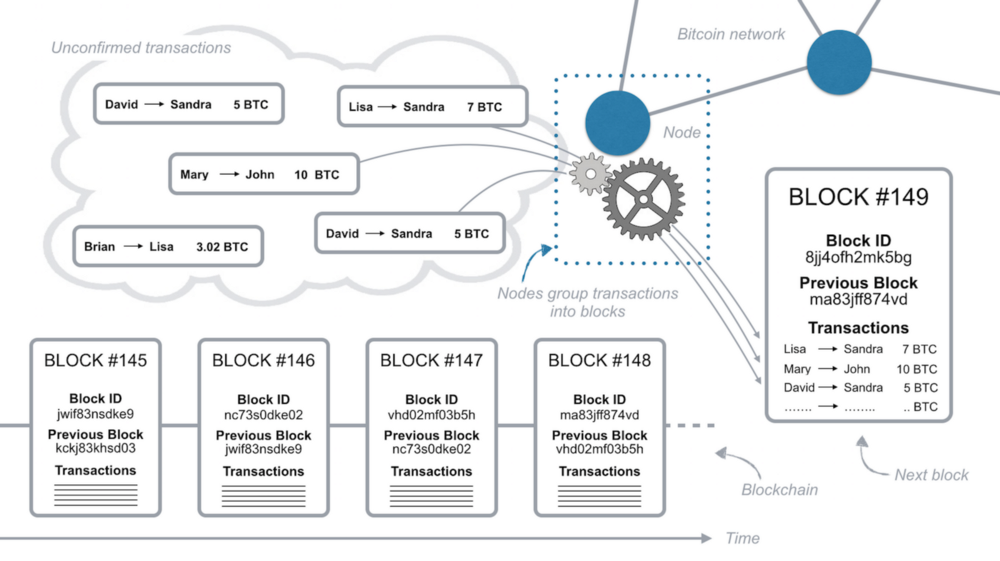
> The problem is that since all users are depending on a central authority , the chance of damage is high ( malicious users disruption , the bank manipulating your money, governments deciding on economic policies which you have no control over etc… in other words , consensus is decided by a few people. )

> Why is consensus important for achieving value? An example that is often used in the blockchain context is called the Byzantine General’s problem ( <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.126.9525&rep=rep1&type=pdf> ) . The explanation given by Satoshi Nakamoto , who designed bitcoin, is a good example of how this is solved (<https://www.mail-archive.com/cryptography@metzdowd.com/msg09997.html>) .

> An alternative to this approach is blockchain , where consensus is derived from users of a system , and not from a central authority . For this to work , the users of the system have to agree on three things about the currency that is using blockchain - that is an accurate measure of value ( ie , can be exchanged for a fair amount of goods and services when needed) , that faith in the demand for the currency persists , and that holding the currency will not lead to a massive fluctuation in value.

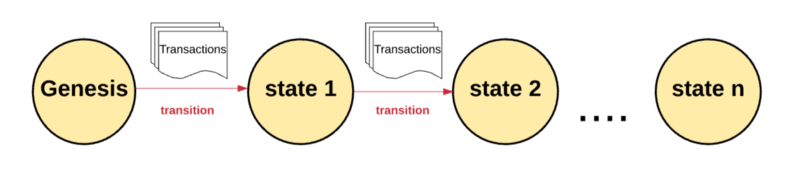
1.1.2 How ?

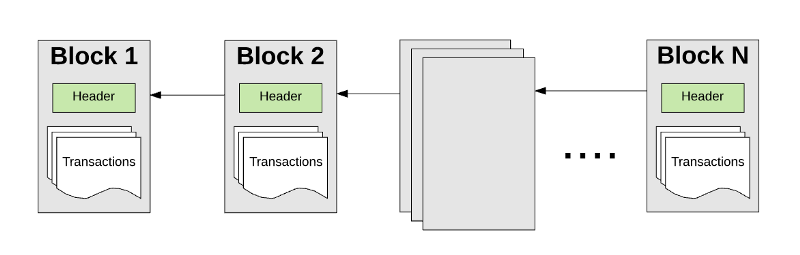
> I will try to explain how the blockchain architecture works using bitcoin as an example. Note : Bitcoin is NOT the same as blockchain. Blockchain is the technology which powers bitcoin , while bitcoin in a cybercurrency which can be traded , similar to other currencies like USD or GBP.   
> Three key terms here - cryptographically secure ( hard to crack ) , transactional singleton machine ( at any time , there is a single instance of the machine responsible for all transactions created and stored in the system ) and shared state ( transactions available to everyone ) define blockchain.   
> The blockchain uses a ledger ( a list of transactions ) , which is not stored in a single place but is distributed via a network of devices that store and execute computational data and functionality. Each device is called a node. When David wants to transfer 2 bitcoins to Sandra , he broadcasts a message stating that his balance goes down by 2 bitcoins and Sandra’s balance should go up by 2 bitcoins. All nodes in the network receive this message .   
> Transactions are performed through a wallet , which is a program that allows you to transact bitcoins. Each wallet has a public and a private key. When David wants to send Bitcoins, he needs to broadcast a message encrypted with the private key of his wallet, so he and only him can spend the Bitcoins he owns as David is the only one to know his own private key necessary to unlock his wallet. Each node in the network can cross check that the transaction request is coming from David by decrypting the transaction request message with the public key of his wallet. This process is propagated through all nodes in the network.   




Transactions are grouped into blocks, which has current and previous block header information.

1.1.3. How are the transactions arranged ?



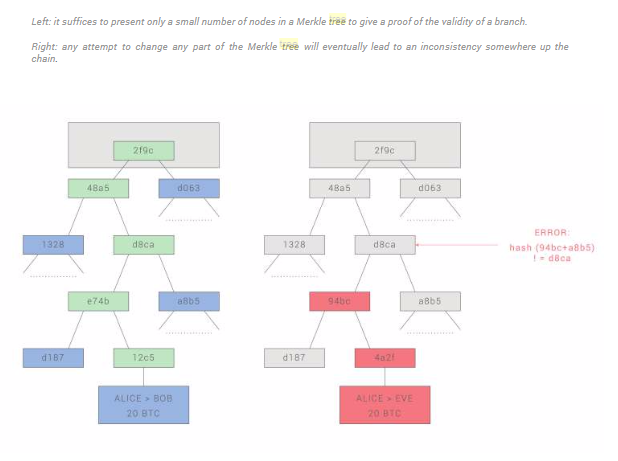
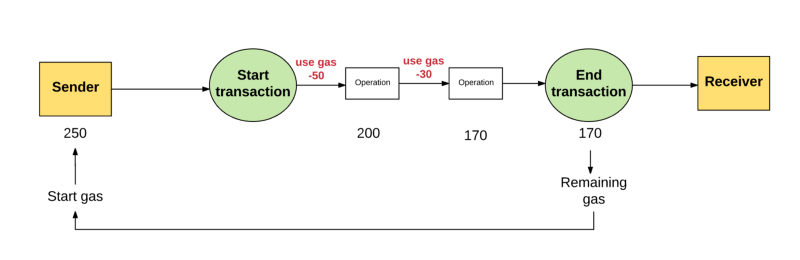
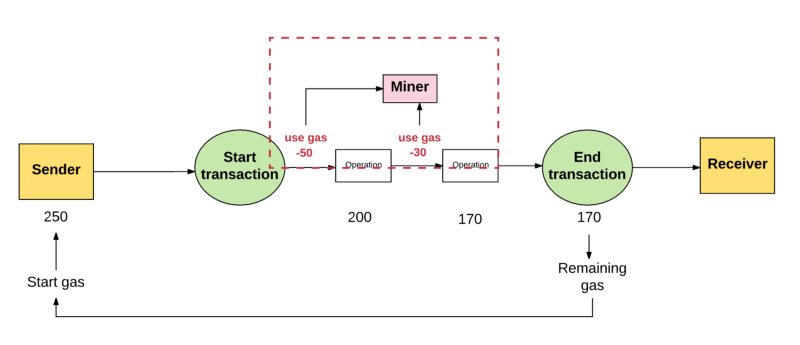
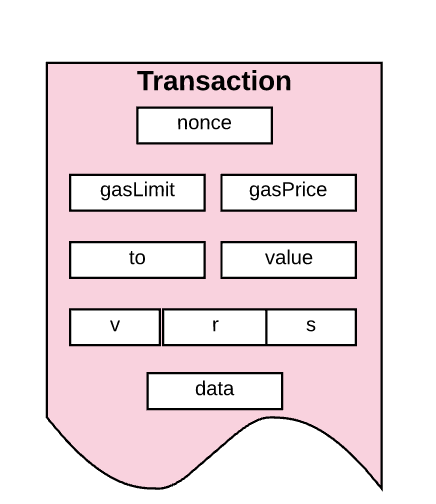
Genesis refers to the first point ( when the blockchain was created ) . After a set of transactions happen , it transitions to the next state and so on. At any given point , there is only one state that exists.   
  
  


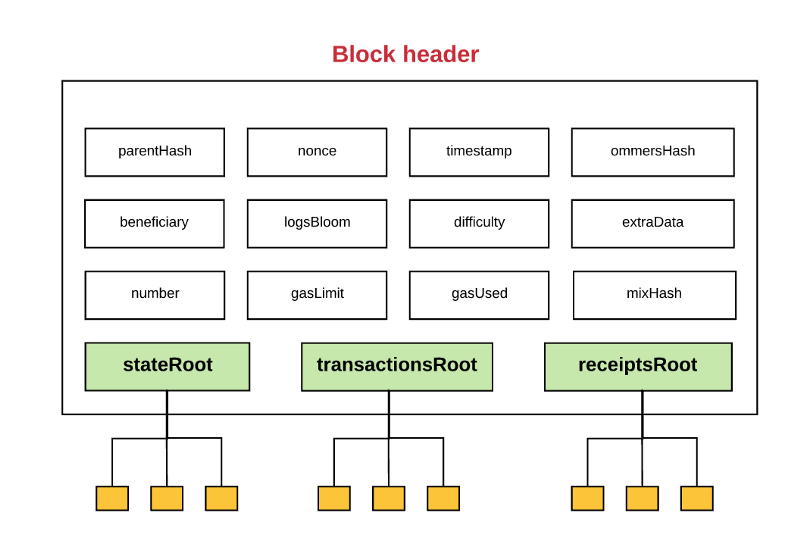
A set of transactions is grouped into a block , which has a header associated with it. The n+1th block’s header points to the nth block’s header, therefore ensuring continuity.

1.1.4. Now , how is the mathematical integrity of the transactions ensured ?   
It goes through a validation process called mining , where a set of nodes on the network identify themselves as miners and use their computational resources to create and validate all transactions in a block. The output of this is a mathematical proof (proof of work). There are many miners working on solving a single block , where speed of result decides if the computation performed by the miner is added to the chain. The miner gets paid for this in digital currency.   
  
1.1.5 How to decide path validity ( which path to take ? )   
As miners compete to add blocks to the chain, forks are possible (similar to forks in Github, where people can make changes to the main commit). There are a lot of methods for this , but in my opinion , the one which the best cohesion with the blockchain principles is the GHOST ( Greedy heaviest observed subtree ) protocol (<https://eprint.iacr.org/2016/545.pdf>). The logic is to consider the chain alternative where amount of computation done is maximum. As the length of the chain increases, more computation must be done quickly along the chain. This is a method where the input variable is a function of max(computational effort) in min(time).

2. Ethereum

For the writeup ( and possibly the paper ) , I propose a variation on the Ethereum blockchain system which is applied to transportation and financial derivatives. This is why I am explaining Ethereum in detail in this section.   
  
2.1 Key terms   
2.1.1 Accounts - Two types of accounts . a) Externally owned accounts , controlled by a private-public key combination and no code, whose working is explained in 1.1.2 . b) Contract account - bound by contract code . External accounts can communicate with each other ( to transfer tokens ) and with contract accounts to perform some functionality. A contract is like a function to which arguments are passed. Contracts can communicate among themselves, but only when initiated by an external account.   
2.1.2 State - Has four components a) nonce - basically , count(transaction) b) Balance - amount of token owned by account c) storageRoot - a hash of a Merkle Patricia tree root node ( the tree is basically a secure way to store key,value pairs like in a Python dictionary - for more : <https://github.com/ethereum/wiki/wiki/Patricia-Tree> ) . This link explains the logic behind the tree - <https://ethereum.stackexchange.com/questions/6415/eli5-how-does-a-merkle-patricia-trie-tree-work> This link is the explanation of a working example : <https://easythereentropy.wordpress.com/2014/06/04/understanding-the-ethereum-trie/> . The use of this is that each block in this structure stores hash information about the state , transactions and receipts . If a malicious user messes with the transactions , the error is generated which moves up all the way to the root node, thereby ensuring it does not happen. Also, nodes need not store all information and still ensure security ( called light nodes ) .On the other hand , nodes that download and store the entire chain are called archive nodes. I do not understand the intricacies of the tree myself , but spending time on it might yield results. d) Codehash - the hash of the Ethereum virtual machine - like an index value for when a function is executed .

  
Merkle Tree working   
   
2.1.3 Gas - value paid to computation to perform task. Includes gas price ( amount of value you are okay with spending for gas ) , gas limit ( limit set for max gas for every transaction ) . The gas money goes to the miner.   
  
How gas works   
  
How miner gets paid   
2.1.4 Fees - like gas , but for data storage. Having it ensures the network is abused.   
2.1.5 Transactions - think of it as a set of data and instructions ( validated , generated by external account ) that is serialized and then submitted for approval to the blockchain. Like making an API request with your params in there.   
  
A transaction   
  
2.1.6 Blocks - a set of transactions that are grouped together. Has a block header , transaction information and other block headers for the current block’s ommer. An ommer is a block whose parent is equal to the current block’s parent’s parent. Ethereum has a block time which is 40 times greater than Bitcoin, so transactions happen fast. Since >1 miners are competing for a block , ommers ensure that the failed computations ( not fast enough , took more time ) are also rewarded with a small value. Not all computations are rewarded , only the top few are.   
The block header has the following information :

  
2.1.7 Transaction execution - for a transaction to execute , first , info about whether you have gas , transactional signature etc are checked . Minimum value constraints are also imposed. If yes , execution starts. During execution , Ethereum keeps track of the substate, a way to record information when transaction is being processed and is needed after transaction is completed. Substate is running , then remaining value ( if any ) is given back to sender . The output is a new state and a set of log information. Pretty much like a credit card transaction.   
2.1.8 Contract - a piece of functionality is written. Once the \_init code for that function is executed, it uses gas. If function uses more gas than available, an exception is called , sender is NOT refunded gas. However , value is refunded. Functions can call other functions by giving return value of first function as args for second function.

2.1.9 EVM - Ethereum Virtual Machine - a stack based VM limited by gas , with volatile memory , non volatile storage . Converts code written in Solidity to EVM-bytecode ( much like literally any other machine ) .

2.1.9 Mining - it is a security mechanism ( strongest and therefore the best survives ) , ensures wealth distribution . Issue : single node might own > 51% of total computational power , making network believe that that node can lead to the heaviest computational chain. Solved by making proof of work algorithm sequentially memory hard ( a nonce needs memory AND bandwidth both ) . Reduces chances of centralization.   
2.1.10 Proof of work - Ethereum’s proof of work algorithm is the Dagger Hashimoto algorithm (more info here : <https://github.com/ethereum/wiki/blob/master/Dagger-Hashimoto.md> ). I do not fully understand it.

3. Transportation

3.1 Why do people need transportation ?

a) To move people from one place to another   
b) To move objects from one place to another   
c) To derive some form of satisfaction ( you drive around in your Porsche because it makes you feel cool , you drive around at nights because it relaxes you etc )   
d) To derive value from owning assets ( Enterprise and Hertz own cars to make money off of them )  
  
3.2 What’s in it ?

At its very core , transportation has a few essential components :   
a) A device ( car , plane , skateboard etc )

b) some kind of fuel   
c ) A person / code ( for driverless vehicles ) running the device   
d) An object in the device   
  
The output is some use ( used to move people , goods , achieve happiness by driving around ...whatever )

3.3 Rolling up

Now , let us further consolidate 3.1 and 3.2 into something very basic - a collection of resources and functionality to produce some kind of output.  
The resources here refer to physical objects that have mass ( this is a bad way of putting it across , but the point I wish to convey is that resources are a car , a person , a fuel , an object ...in other words , ).   
Functionality here refers to some kind of task that is being performed. For example , the task being performed can be moving a package from one place to another , moving people from one place to another , having the expectation of deriving happiness from driving etc.   
Output here refers to what happens after the functionality is executed . For example , when an object or a person is delivered from one place to another , the output is the money you make for delivering the object or person.

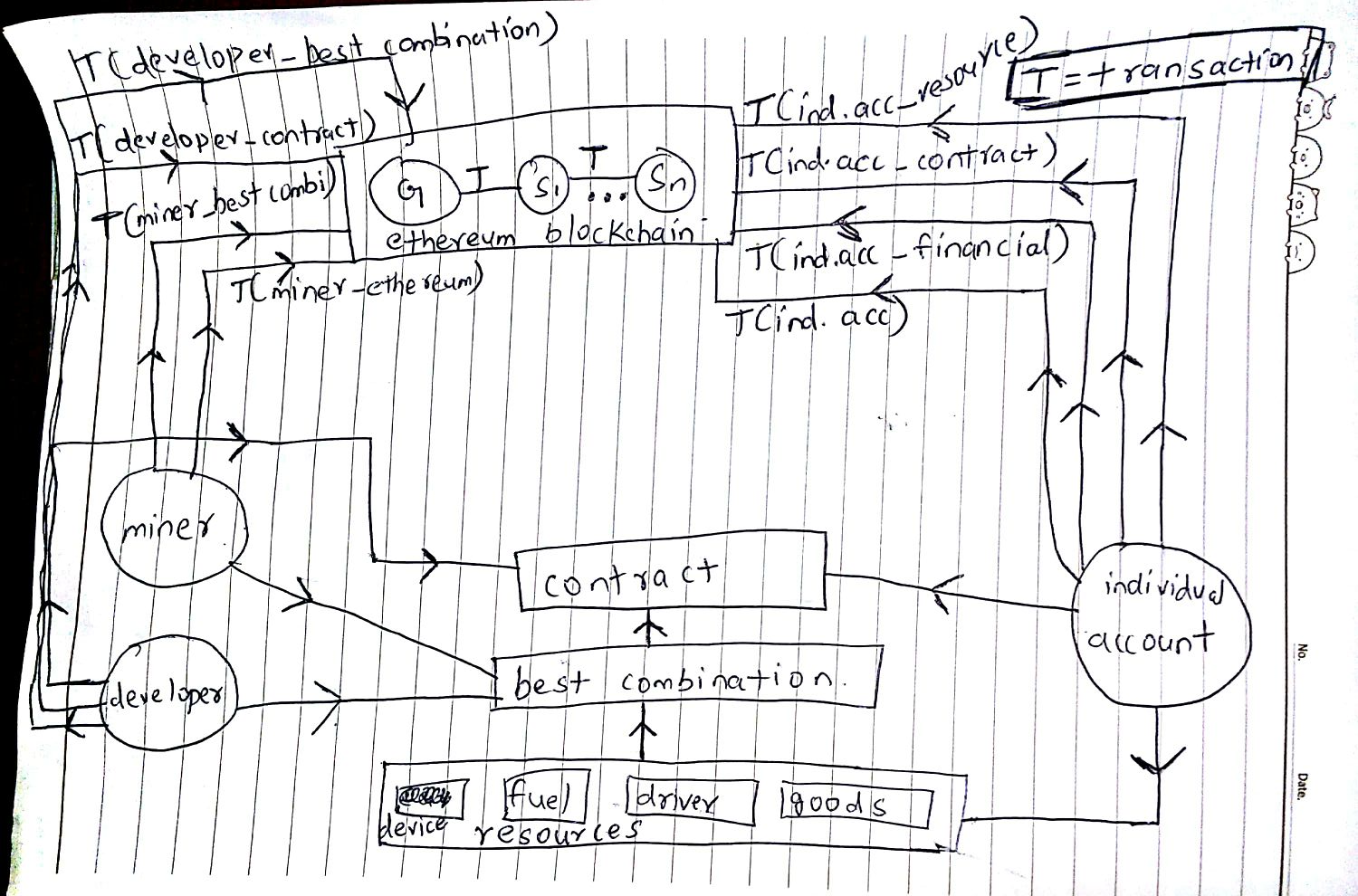
Simply put, the resource executes some functionality to get an output.

3.4 Ownership   
  
Today , transportation device ownership follows a one to one or a one to many relationship between person/company and device. For this paper , we are considering a many to many relationship between person/company and device. For example , consider this :   
A group of 5 people/companies ( say x1:x5) own 5 devices (d1:d5) . In the current scenario , x1 can own d1, d2,d3 . x2 can own d4 and x5 can own d5. X1:x5 now have to make sure they have ownership of the car , fill fuel in the car , take care of maintenance and basically own a device that is in essence a depreciating asset over time and lies unused > 50 % of the time that it is owned.   
Now , consider this scenario : d1 is owned 30% by x1 20 % by x2 10 % by x3 25% by x4 and 15 % by x5 . d2 is owned 70% by x2 and 30 % by x3 . d3 is owned 100 % by x4 , with d4 and d5 following a similar logic. Additionally, people/companies and devices are located at different geographical locations.

3.5 Attribute definition   
3.5.1 Resource   
3.5.1.1. Device - LatitudeLongitude, ObjectStorageCapacity, PeopleStorageCapacity,FuelType,FuelCapacity,CurrentFuelCount,OwnershipRecord,Availability,HasPermit,GoOnLand, GoOnWater,GoOnAir,AverageSpeed, CurrentObjectStorageCapacity,CurrentPeopleStorageCapacity  
3.5.1.2 Driver - LatitudeLongitude, HasLicense,Availability,SkillRating   
3.5.1.3 Fuel - LatitudeLongitude, FuelType,FuelSuitableFor, ExistingFuelVolume, FuelDepreciationRate  
3.5.1.4 Goods - LatitudeLongitude, GoodDimensions,GoodWeight,GoodPerishableYesNo, GoodImportance,GoodFragileYesNo,GoodCount,

4. Investment products   
Simply put , what is an investment product ? It is a product purchased at time T with the hope of making a profit in time T+n . Products include , but not limited to , stocks, options, futures, bonds, mutual funds, certificates of deposit, money market investments, ETFs and annuities.   
The basic idea is to bet on an underlying asset and make money later.

5. An Ethereum like blockchain structure for transportation including financial derivatives   
5.1 Assumptions : Any kind of transaction occurring in this system is still assumed to follow the three basic tenets of the blockchain architecture - cryptographically secure ( hard to crack ) , transactional singleton machine ( at any time , there is a single instance of the machine responsible for all transactions created and stored in the system ) and shared state ( transactions available to everyone )

5.2 Each component explanation   


Let us consider the following structure.   
The existing Ethereum components are developer , miner , ethereum blockchain , individual account , contract . The new components are best combination , resources and financial derivatives.   
In the above diagram , I have mentioned only one miner , one developer , one individual account and one contract for simplicity. In reality , the numbers are very high , but to understand the concept , a single element is mentioned.   
Developer - This is the person who develops the contracts. He/she writes code that may take in inputs and execute smart contracts.   
5.2.1 Miner - explained.

5.2.2 Ethereum blockchain - explained   
5.2.3 Individual Account - explained .

5.2.4 Contract - explained.

5.2.5 Resources - These are the “physical things”. A brief explanation of what they is above , but this section will go into detail.   
5.2.5.1 Device - a device here is something which can be used to transport goods from one place to another. Examples :include cars , ships , airplanes etc .   
5.2.5.2 Fuel - Fuel is the thing that is needed to power the devices. Examples : petrol , diesel , electricity.   
5.2.5.3 Driver - the Driver is the person / code ( in the case of self driven vehicles ) who drives the device.   
5.2.5.4 Goods - goods are basically objects that need to be transported from one place to another. It is important to understand that goods can be of different sizes and types , from fragile diamonds to huge cargo shipments.   
Best combination - The best combination is a function which identifies the ideal transaction parameters from the resources ( device , fuel , driver goods ) which can be combined to perform a contract to have a transaction that leads to the best possible exchange rate of digital value for a contract that has been executed. The best combination function can be made by developers to suit a particular scenario , or can be made as a standard in the Ethereum ecosystem. Figuring out the best combination is a computational task.

5.3 How to the different components cohese ?   
Each component in the set of resources has some attributes ( mentioned above ) .A set of individual accounts own different percentages of each item in the resource set. The manner in which this works has been explained in Section 4 . When an individual account ( which wants to get some task done for which it is willing to pay digital value ) executes a contract , best combination comes into play. The logic for best combination is written by developers ( who get paid some digital value because their contract code is being used ) and the computational task of actually figuring out what the best combination is is done by miners ( for which they get paid ). After the best combination is decided and agreed upon by the individual account resource owners and the individual account contract caller , the transaction is performed. When the transaction is finished , the transaction is added to the blockchain. Since it is the individual accounts which own the resources , they also have the option of exercising financial investment products on the resources owned by other individual accounts. The kind of financial investment products that are suitable in this scenario are stocks , options , futures , mutual funds , ETFs and annuities. This would mean that individual users have the ability to perform financial transactions on the resource set ( and since this is a zero sum game , the only people who make any digital value are the individual account resource owners , miners and developers who write contracts to figure out what the best potential return on investment at the moment is **).**  Bonds and certificates of deposits are not possible in this scenario. Now , how are the resource components organized and how do they maintain a synchronous state transfer ? I am conflicted about this and have three ideas in mind. The first is that we can consider each item in the resources to be its own blockchain. When a resource is first added to the Ethereum ecosystem , it gets its own genesis state , and after every set of transactions that it performs, the state changes. The set of transactions performed here are reflected in the ethereum blockchain as well , because the digital value is decided by the Ethereum blockchain. The advantage with this system is that the number of transactions ( and therefore the number of blocks ) for each resource item would be relatively very small , so the amount of computational power needed for this would be pretty low. The obvious issue with doing this is that over time , the asset would obviously go worthless ( old cars rust , fuel sources run out , people die and goods get decomposed) . The second method is to simply store all of this information as data , and when every time any function or user decided to use the data , they would have to pay a storage fee. The data would be updated after every transaction. The issue with this is that the amount of data stored in this kind of system would be huge.Imposing a pay as you use type of reward system ( where individual accounts pay for every instance of the data called ) would solve the problem to an extent. The third kind of system would be to have a different digital value for the resources and to have a atomic exchange mechanism to ensure that there is a balance between the Ether and the digital value for the resource ( more about atomic exchange here : <https://www.multichain.com/developers/atomic-exchange-transactions/> ) .

What do the different transactions mean :

Transactions explanation

>Developer \_ best combination - dev gets paid when his best combination code is used

>Developer \_ contract - dev gets paid when his contract code is used   
>Miner \_ best combination- miner gets paid for ensuring mathematical accuracy of best combination

>Individual accounts \_ resource - Individual accounts get paid when the resources they own make money

>Individual accounts \_ contract - Individual accounts lose when they execute a contract

>Individual accounts \_ financial - Individual accounts get paid when they perform a financial market transaction

>Individual accounts - get paid when value is transferred among them

5.4 Explaining this using the Ethereum architecture   
5.4.1 Accounts - logic remains same as Ethereum   
5.4.2 State - logic remains same as Ethereum

5.4.3 Mining - logic remains same as Ethereum

5.4.4. Proof of work - remains same as Ethereum

5.4.5 -EVM - Ethereum Virtual Machine   
5.4.6 - Contract   
5.4.7 - Gas - In the case of Ethereum , it is the value that is paid to perform a task. In this implementation , it it however not that simple. The concept of Gas is still used , but the only thing is that because it is not just computational tasks occurring for which gas is paid , but the provision of an actual service for which resources are needed , it will be prudent to account for those costs as well.   
The categories of Gas : Value for providing best combination algorithm , value for providing best combination algorithm proof of work , value for using resources , value for getting the transaction to the blockchain (paid to miners). The concept of gas has to be redefined to include not just computational , but compensation for resource usage and compensation to account for resource usage depreciation ( especially in the case of device and driver ) . The Gas value has to be divided among the miner , developer and resource owners , with the resource owners getting a larger percentage of the output value.   
5.4.8 - Fees. In the context of Ethereum , fees were charged for data storage. Here , fees will also have to be charged for providing access to the resources data as well. This will, similar to Ethereum , ensure that the network is not prone to abuse.   
5.4.9 Transactions - a lot more conditions would have to be verified when compared to the Ethereum case.   
5.4.10 Blocks : While the Ethereum blockchain would proceed as is , the relationship that exists between the resources and the Ethereum blockchain would mean that a lot more transactions would potentially be performed. The omner size would have to be reduced , especially if the case of considering each resource item as a blockchain by itself is true.   
5.4.11 - Transaction execution The substate needs to exist for a longer period of time than a normal Ethereum transaction , because the contract would typically not be executed as fast as an Ethereum contract would be. Remaining value would also have to be divided proportionally between the different users.

6 Applications - discuss during meeting

6.1 - Using this to ensure that there is a tracking mechanism for resources being moved across geographical boundaries - <http://www-03.ibm.com/press/us/en/pressrelease/50816.wss>

6.2 Predicting transaction execution leftover value based on transaction parameters to make money

6.3 Resource object flow - how to avoid bottlenecks by redistributing object assignment   
6.4 Resiliency in areas where number of users are limited

6.5 The economic side of it - Knowledge in coordination by Daniel Klein .

6.6