ECE 595: Foundations of Blockchain Systems **October 14, 2020**

Lecture 5

Lecturer: Sreeram Kannan *Scribe: Yunguo Cai*

In this lecture we will first further discover the Bitcoin system including transaction formats, underneath P2P layer and transaction confirmation mechanism. For the next step we will dive into elucidating how nodes behave by classifying two kinds of behaviors, adversarial and irrational, and talking about action space and simple analysis.

5.1 Layers of a Bitcoin System

When we talk about a complex system such as a physical network, it is very useful to organize them into layers so it is clear to see the different tasks performed by each layer. The analysis concept is similar in terms of blockchains. As mentioned in Lecture 1, there are primarily four layers in a blockchain, including P2P, Consensus, Scaling and Application.

5.1.1 Peer2Peer

The functionality of Peer2Peer layer is to provide a mechanism for distributing blocks while accommodating churns. The vast majority (97%) of Bitcoin nodes exhibit intermittent network connectivity (churn), and this churn results in significant numbers of unsuccessful compact blocks, roughly twice the figure for continuously connected nodes. If a certain node wants to join in, the network has to expand to accommodate and similarly if a node wants to leave, the network also has to readjust to accommodate, which is called churn in peer-to-peer network. The P2P layer, also known as a network layer, is the one that is responsible for internode communication. Each computer in a P2P network is called a node. This layer can also be termed as propagation layer. This P2P layer ensures that nodes can discover each other and can communicate, propagate and synchronize with each other to maintain valid current state of the blockchain network.

Every P2P protocol requires an bootstrap node to usher you into the network and help you initialize your peer list. This bootstrap node is your entrance point into the P2P network, from which point you can then organically find new peers. The danger with a bootstrap node is that if it is not authenticated, it could be malicious and perform a man-in-the-middle or eclipse attack. In the canonical Bitcoin implementation, these bootstrap nodes are hard-coded as trusted DNS servers maintained by the core developers.

One of the weaknesses of P2P protocols is quality control. Bitcoin uses a reputation system to deal with this problem. Say you're a Bitcoin node and you've just bootstrapped your peer list. You start by assigning each of your peers a spam score of 0. Any time that peer misbehaves toward you, their spam score goes up. Once a peer accumulates 100 points, your client automatically bans them for 24 hours and stops gossiping to them. Though these spam scores are not actually propagated in the protocol across peers, this system still serves as a decent defense against misbehaving nodes.

To achieve better privacy, Bitcoin use the method called diffusion to propagate gossip messages. In diffusion, instead of immediately flooding to each peer, the client waits a random exponential delay before gossiping to each of its peers. This has the effect of obscuring the P2P message graph,

making it harder to observe where the ”message wave” originated from. However, diffusion still leaks quite a bit of information to a passive adversary. Furthermore, P2P messages are not bilaterally encrypted, so passive packet sniffers can easily snoop on any Bitcoin traffic.[1]

5.1.2 Consensus

Consensus is actually the agreement different nodes come into on what the current ledger is. Consensus protocols (algorithms) create an irrefutable set of agreements between nodes across the distributed P2P network. Consensus ensures that power remains distributed and decentralized. No single entity can control the entire blockchain network. Bitcoin is a permissionless blockchain network and the consensus used is known as probabilistic consensus. Such a consensus guarantees consistency of the ledger, though there is a possibility that various participants have different views of the blocks. This means that they remain vulnerable to ledger forks (also known as divergent ledgers). The consensus of bitcoin is reached by the combination of PoW (Proof of Work) mechanism, the longest chain protocol and the k-deep confirmation rule. K-deep confirmation rule basically means that if you take the current longest chain and remove the last k blocks, all of the other blocks are thought of as confirmed.

5.1.3 Scaling

Scaling is some version of resource allocation in that different nodes do different tasks and the resource is thus allocated in the distributed manner. Bitcoin does not have a scaling layer because bitcoin is based on full replication. Full replication is that every node receives every block right in the ledger, every node stores every block and every node validates every block, which means that it computes all the account balances if you execute this block. Since every node receives every block, it is communication inefficient. Since every node stores every block, it is storage inefficient. As more nodes join the system, there is no improvement in performance because every node needs to do all the tasks in a replicated way. The only improvement is really in security. For example, we've been talking about that Bitcoin is secure when 50% of the computational power is held by honest users. If you have more nodes, then that means there is a lot more computing power in terms of honest users. So you get improvement in security, but not really improvement in performance. Performance is divided into three things - communication, storage and validation (namely computation). Since every node validates every block, it is computation inefficient.

5.1.4 Application

Application is to do something useful like building a payment system, a kind of cryptocurrency or a social network and so on. It aims to conduct interaction through blockchain applications. The application layer is comprised of smart contracts, chaincode, and dApps. Application layer can be further divided into two sub-layers –application layer and execution layer. Application layer has the applications that are used by end users to interact with the blockchain network. It comprises of scripts, APIs, user interfaces, frameworks. For these applications, blockchain network is the back-end system and they often connect with blockchain network via APIs. Execution layer is the sublayer which constitutes of smart contracts, underlying rules and chaincode. This sublayer has the actual code that gets executed and rules that are executed. A transaction propagates from application layer to execution layer, however the transaction is validated and executed at the semantic layer (smart contracts and rules). Applications sends instructions to execution layer

(chaincode), which performs the execution of transactions and ensure the deterministic nature of the blockchain. DApps is a distributed application that runs on top of a distributed technology. It's a decentralized application that leverages smart contracts or chaincode. DApps can be considered a web application that interacts with the smart contract or chaincode; however, the dApps are not controlled by a single entity or an organization. Once deployed, they belong to the blockchain network.DApps are user-friendly applications, which business users can use to transact onto a blockchain network. Smart contracts allow you to connect to blockchains, whereas dApps allows you to connect to a smart contract or chaincode. For example, if you go to LinkedIn, the web page calls APIs, which gather data from a database. However, in dApps and the smart contract world, dApps are API-based web applications that connect with smart contracts, which in turn execute transactions on the ledger.

5.2 Application layer of bitcoin

This lecture talks mainly about the P2P layer and the application layer. We first start from the application layer. As far as Bitcoin is concerned, the application that we're running is just payments. A typical payment ledger is composed of transactions. The norm procedure should be that if someone wants to send a transaction, he/she needs to sign for it.

5.2.1 Types of Transaction Systems

There exist two types of transaction systems.

1. Account-Based System
   In the account-based system, there are user accounts and the balance of the corresponding user account increases/decreases as the transaction occurs.

2. UTXO-based System
   UTXO, which basically means unspent transaction output, is used by Bitcoin. In cryptocurrencies, an unspent transaction output (UTXO) is an abstraction of electronic money. Each UTXO represents a chain of ownership implemented as a chain of digital signatures where the owner signs a transaction transferring ownership of their UTXO to the receiver's public key. The total UTXOs present in a blockchain represent a set, every transaction thus consumes elements from this set and creates new ones that are added to the set. The set thus represents all the coins in the system. A UTXO defines an output of a blockchain transaction that has not been spent, i.e. used as an input in a new transaction. In the case of a valid blockchain transaction, unspent outputs (and only unspent outputs) may be used to effect further transactions. The requirement that only unspent outputs may be used in further transactions is necessary to prevent double spending and fraud. For this reason, inputs on a blockchain are deleted when a transaction occurs, whilst at the same time, outputs are created in the form of UTXOs. A user's wallet keeps track of a list of unspent transactions associated with all addresses owned by the user, and the balance of the wallet is calculated as the sum of those unspent transactions.[7]

   Let's take a look at a simplified example of how the UTXO model works in Bitcoin transactions:

(a) Alice gains 25 bitcoins through mining. Alice's wallet is associated with one UTXO record of 25 bitcoins.

(b) Alice wants to give Bob 17 bitcoins. Alice's wallet first unlocks her UTXO of 25 bitcoins and uses this whole 25 bitcoins as input to the transaction. This transaction sends 17 bitcoin to Bob's address and the reminder of 8 bitcoins is sent back to Alice in the form of a new UTXO to a newly-created address (owned by Alice).

(c) Bob then wants to give Charlie 7 bitcoins. Bob's wallet first unlocks his UTXO of 17 bitcoins from Alice and uses this whole 17 bitcoins as input to the transaction. This transaction sends 7 bitcoin to Charlie's address and the reminder of 10 bitcoins is sent back to Bob in the form of a new UTXO to a newly-created address (owned by Bob).

(d) If now Alice wants to send 25 bitcoins to David and she uses the 25 bitcoins she gained through mining in step (a) as input, this transaction will be regarded as invalid since the transaction output has already been spent(used as input) in step 2.

(e) Say there was another UTXO of 2 bitcoins associated with Bob prior to step (c), Bob's wallet now shows that his balance is 12 bitcoins. Bob's wallet now keeps track of two UTXOs: one from before and another from the transaction in step (c). Each UTXO needs to be unlocked if Bob wishes to spend them.

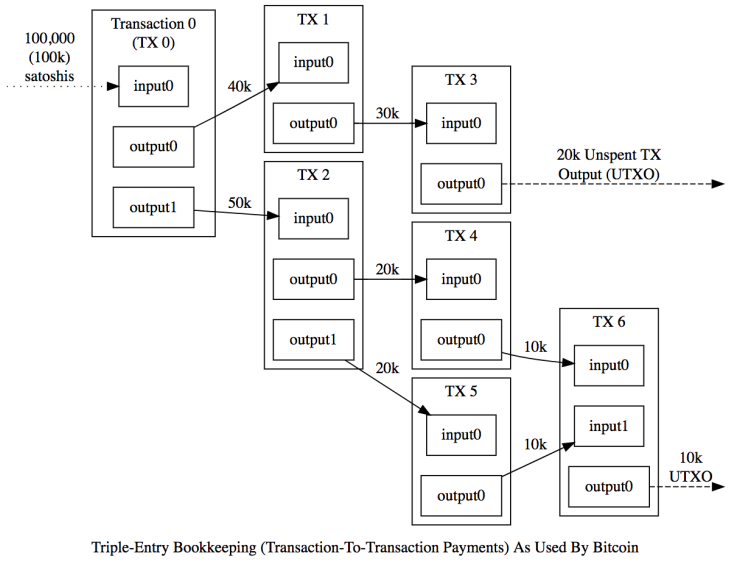

Figure 5.1: Transaction-To-Transaction Payments Used by Bitcoin

Both models achieve the same goal of keeping track of account balances in a consensus system. The benefits of the UTXO Model are:[4]

- Scalability — Since it is possible to process multiple UTXOs at the same time, it enables parallel transactions and encourages scalability innovation.
- Privacy — Even Bitcoin is not a completely anonymous system, but UTXO provides a higher level of privacy, as long as the users use new addresses for each transaction. If there is a need for enhanced privacy, more complex schemes, such as ring signatures, can be considered.

The benefit of the account/balance model are:

- Simplicity — Ethereum opted for a more intuitive model for the benefit of developers of complex smart contracts, especially those that require state information or involve multiple parties. An example is a smart contract that keeps track of states to perform different tasks based on them. UTXO's stateless model would force transactions to include state information, and this unnecessarily complicates the design of the contracts.
- Efficiency — In addition to simplicity, the Account/Balance Model is more efficient, as each transaction only needs to validate that the sending account has enough balance to pay for the transaction.

An important aspect of Bitcoin is called coupled validity. When creating a block, you can determine whether each transaction is valid because when creating a block you have fixed the parent and then this blog is going to be in a certain position that you know which transactions are valid. So in Bitcoin each block is valid only if all transactions are valid. A block is considered valid only if all transactions in the block are valid relative to its history. So when you mine a block, you want to mine only valid blocks because all of the nodes are going to immediately check the validity before accepting it when they receive the block. Bitcoin is forcing a coupling between application layer and consensus because it is the longest valid chain and what is valid depends on the application layer to bitcoin in a vertically integrated system. Actually for the blockchain, the right thing to do is to break this coupling. Consensus is not required to be only on solid blocks. Bitcoin is for whatever simple purpose but after that we will have been trying to do scaling and very complicated applications. The existence of the coupling prevents many things from carrying forward.

It doesn't help to get rid of the double spending attack because what it helps is to make sure that if your transaction is well embedded into the ledger, you can immediately infer that it's valid. Validity can be inferred just by its presence. People don't have to actually go and check all the history and everything like the previous accounts. It doesn't say anything about double-spending because double-spending happens on distinct chains of the block.

5.2.2 Types of Transaction

We've already known that Bitcoin use the UTXO transaction system. Now we will talk about the types of transaction. There are two types of transaction in Bitcoin.

1. Pay-to-Public-Key[2]
   Pay-to-PubKey-Hash (Pay-to-Public-Key-Hash, P2PKH) is the basic form of making a transaction and is the most common form of transaction on the Bitcoin network. Transactions that pay to a Bitcoin address contain P2PKH scripts that are resolved by sending the public key and a digital signature created by the corresponding private key. Two types of payment are referred as P2PK (pay to public key) and P2PKH (pay to public key hash). Satoshi actually decided to use P2PKH instead of P2PK. A Bitcoin address is only a hash, so the sender can't provide a full public key in scriptPubKey. When redeeming coins that have been sent to a Bitcoin address, the recipient provides both the signature and the public key. The script verifies that the provided public key does hash to the hash in scriptPubKey, and then it also checks the signature against the public key.

2. Pay-to-Script[5]
   Pay-to-script is to release the money to the node if the certain programmatic condition you set up is satisfied, which is the basis of smart contract. It is an advanced type of transaction used in Bitcoin and other similar cryptocurrencies. Unlike P2PKH, it allows sender to

commit funds to a hash of an arbitrary valid script. It basically allows you to create your own custom "redeem scripts", but still be able to share them easily with other people. Before P2SH, if you wanted a complex locking script (e.g. P2MS) placed on your bitcoins, you would have to give the person "sending" you those bitcoins the entire locking script. But with P2SH, instead of giving someone an entire locking script, you can essentially just give them a hash of your script instead. As a result, the sender is no longer burdened with the size (or the details) of your locking script.

As mentioned, P2S is the basis of smart contract. We'll talk a little bit more about smart contract and the script programming language. A smart contract is a computer program or a transaction protocol which is intended to automatically execute, control or document legally relevant events and actions according to the terms of a contract or an agreement. Typical examples of smart contracts on Bitcoin include not allowing 0.1 BTC to be spent until 2021, or requiring more than one person to sign off on a transaction before the money can actually move. Thus far, Bitcoin Script is the language that makes these contracts possible. The problem is it's tricky to work with Bitcoin Script. It is unlike other, more popular programming languages developers are used to, making it harder to wrap their heads around and compose in. This lack of understanding also makes it easier to make a mistake, potentially putting Bitcoin at risk. It turns out that the Bitcoin smart contract language is not generally enough because it will only put in to satisfy certain basic conditions and later Ethereum as another blockchain basically built a system that can handle universal smart contracts.
The unwieldiness of Bitcoin Script was one of the factors that led Vitalik Buterin to design the Ethereum platform in the first place. Solidity, Ethereum's first smart-contract language, was designed to be much easier for developers to read and thus use. And it has paid off: Ethereum has grown to become the go-to platform for smart contract developers. Ethereum virtual machine, or EVM for short, is a blockchain-based software platform. It allows developers to create decentralized applications (Dapps)[3]. Programmers value them for having no downtimes and keeping all created objects safe from modifying. Instead of a distributed ledger, Ethereum is a distributed state machine. Ethereum's state is a large data structure which holds not only all accounts and balances, but a machine state, which can change from block to block according to a pre-defined set of rules, and which can execute arbitrary machine code. The specific rules of changing state from block to block are defined by the EVM. The evolution of blockchain makes Ethereum have a different application layer from Bitcoin, which we call as Bitcoin script. The programming language is referred to as Bitcoin scripting. The other popular blockchain, for example, Facebook's major blockchain effort called Libra or CaLibra has a programming language called Move and their own virtual machine associated with it.

5.2.3 Roles of nodes

There are three categories of nodes. Clients are those who have money when making transactions. Nodes are those that keep track of the full Bitcoin history, the Bitcoin ledger. Miners are those who mine blocks. Mining node means that you are not only keeping track of the current blocks, but you are also trying to make the next one. Those mining nodes will get rewarded since they are really the one providing kind of free service. The system of internal incentives make sure that the nodes actually maintain the system.

5.3 P2P Layer of Bitcoin

In the peer-to-peer network, anybody can join and leave the bitcoin network. Nodes are all connected to the Internet through an overlay network. A peer-to-peer overlay network is a computer network built on top of an existing network, usually the Internet. Peer-to-peer overlay networks enable participating peers to find the other peers not by the IP addresses but by the specific logical identifiers known to all peers so that everybody does not need to maintain the addresses of all other nodes in the Bitcoin because that;s a highly variable set and the set may keep changing. Each node connects to a small subset of neighbors which are chosen randomly (eight other nodes in Bitcoin but the number maybe different in other systems).
Peer-to-peer networks themselves have a long history and roughly there are two families of peer-to-peer networks. One is called structured and another is called unstructured. [6]

- Unstructured Overlay
  Unstructured overlays have been used in several widely used file-sharing systems, despite their inefficiencies. In the research community there has been much effort to study the properties of these overlays using crawlers to measure the overlay network, peer, and content properties. In addition, many improvements have been suggested to increase their performance and reduce overhead. Since structured and unstructured overlays have somewhat complementary characteristics, there are proposals to create hybrid overlays that combine both types of routing algorithm. Patterning the overlay organization on power law and social networks has also drawn a great deal of interest.
- Structured Overlay
  Structured overlays emerged to address limitations of unstructured overlays by combining a specific geometrical structure with appropriate routing and maintenance mechanisms. In the design of structured overlays, the geometry of the overlay is a key design decision. To be effective for P2P use, the geometry must meet several criteria, including: the overlay must be fully connected for resiliency to peer failure, peers throughout the peer population must have a uniform degree to avoid load imbalance, there must be support for at least one type of distributed routing function that converges, and so on. To organize the large space of structured overlays, one can use the following criteria: multihop versus O(1)-hop, logarithmic degree versus constant degree, and routing using prefix, Euclidean distance, and XOR metrics.

Bitcoin is clearly an unstructured peer-to-peer network. The only parameter that you can offer is broadcast or flooding of data. It can guarantee that you can broadcast data. If you are connected to enough number of nodes, the graph itself is a connector since there is a path from any node to another node. The graph is connected so you can send a block/transaction/any data that you want to distribute to your neighbors. They then send it to their neighbors and so on. The reason this kind of unstructured network is chosen is to be wrestling to adversaries. You increase the number of nodes you connect to and you can decrease the probability that you are not connected to any honest node. If you are not connected to any honest node, a possible attack is called an eclipse attack and then whatever you think is the longest chain may not be the longest chain. So your view may be very different from the view of the network and you may be cheated by showing you a shorter chain or things like that. To prevent this from happening, there are various mechanisms and precautions people take. We will not go into details on these mechanisms but roughly the idea is every node connects to eight random nodes and then this gives you a certain resilience because you can make sure that the probability that you are disconnected from honest nodes is small by making the number of nodes large.

P2P layer is doing two things in Bitcoin. One is that it's propagating transactions when transactions come.

5.3.1 Life cycle of a transaction

- Client Side:
  Suppose that a client has a wallet. He issues the transaction and sends it to a few miners. The miners receive the transaction and flood it to its neighbors. (In Bitcoin, you may additionally check the transaction whether it is valid before flooding).
- Miner Side:
  Then a miner mines the block with this transaction. This happens because each miner maintains a buffer of valid transactions and puts in everything that's valid when mining a block. This is the mining process and this buffer is called mempool. (The mempool is where all the valid transactions wait to be confirmed by the Bitcoin network. A high mempool size indicates more network traffic which will result in longer average confirmation time and higher priority fees.) If a miner succeeds, the miner forwards the block to all other miners (other nodes) in general. Once the block is k-deep, it is confirmed. The parameter k is not universal (not a parameter of the protocol). It's a client specific parameter which can be set in the client's wallet. If the client chooses a larger k, then it has higher latency.

You can measure transaction latency as the time from when the client issues a transaction request until the response is received, or from when the database server receives the request until it queues the response. It includes all the time from the time the transaction is issued, to the time the transaction is propagated, to the time miners mine the blocks with the transaction, and to the time the transaction is confirmed with the k-deep rule.
There is no internal incentive in the system for the miner to flood the transactions to its neighbors. They just do it due to reciprocity. The case is that if I don't send your transaction then you may also not send mine.

5.3.2 Reward Transaction

The first transaction is the reward transaction. So the input for the transaction is null and let the output be 25+3 bitcoins to Alice, for example. Here the 25 bitcoins is the block reward and the 3 bitcoins is transaction fee. Not only for the first transaction, actually all the transactions have implicit fees. Suppose that the input is the 28 bitcoins in the previous step and the output is 27.9 bitcoins sent from Alice to Bob. In Bitcoin output, if the summation of output values is greater than the summation of input values, it means that the transaction is not valid. When the summation of output is less than the summation of input, the remaining is the transaction fee. Whatever the remaining difference between the input and output is considered to be transaction fee and that goes to the miner. The miner collects all of that and put it into this first transaction. This basically gives miners a little bit of incentive, actually could be a lot of incentive, depending on how much this number is. Currently, it's a little bit.

References

[1] HASEEB QURESHI Nakamoto website. *Bitcoin's P2P Network*.

[2] Bitcoinwiki website. *Pay-to-Pubkey Hash*.

[3] Ethereum website. *ETHEREUM VIRTUAL MACHINE (EVM)*.

[4] Flora Sun website. *UTXO vs Account/Balance Model*.

[5] Learnmeabitcoin website. *P2SH*.

[6] Sciencedirect website. *Unstructured Overlay*.

[7] Wikipedia website. *Unspent transaction output*.