EE 374: Blockchain Foundations **Stanford, Spring 2022**

Lecture 5: Blocks

2022/4/13

Lecturer: Dr. Dionysis Zindros
Scribe: Kachachan Chotitamnavee, Schwinn Saereesitthipitak, Alan Zhang

1 Protecting against Double Spend Attacks

1.1 Network Delay

To ensure complete security of the network, all nodes must have ledgers that completely agree on the set of transactions that has happened[1]. Otherwise, if two nodes have ledgers that disagree than one another, then there may be disagreements on what transactions are valid and what transactions are not. For instance, if Alice believes that she has 10 dollars left, whereas Bob believes that she only has 3, then if Alice makes a transaction with a value of 5, she will think that her own transaction is valid, whereas Bob will not, leading to a disagreement.

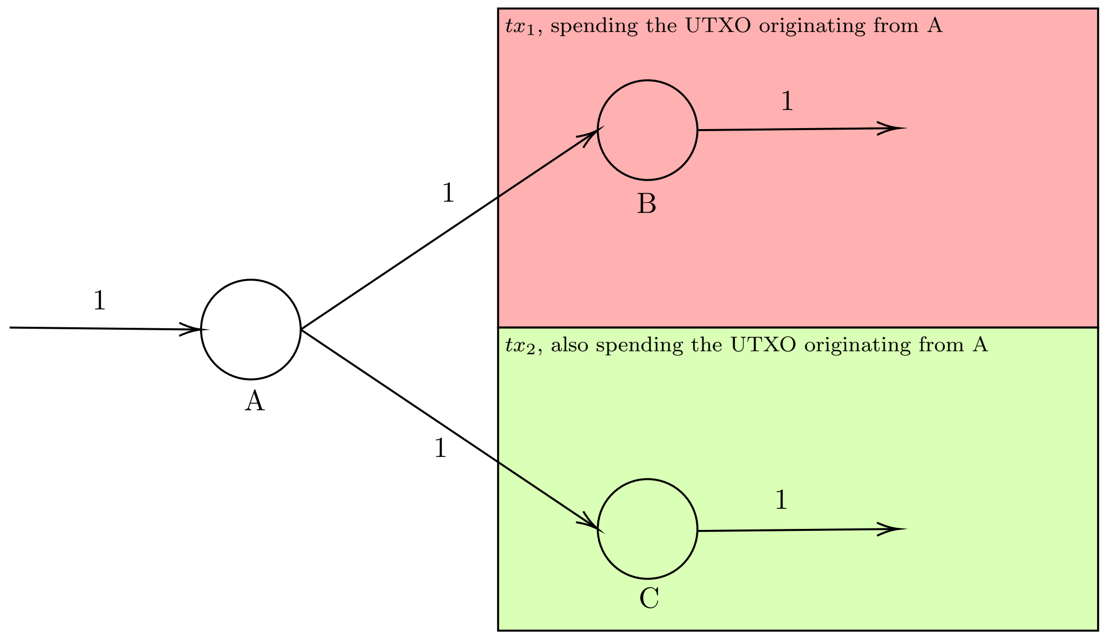
Using the model we have developed in class so far, it is impossible for there to be complete consensus among the ledgers. This is because we assume every network has a network delay. Network delay, or Δ, is defined as the maximum time needed for a message to reach from one honest party to every other honest party. In other words, when a honest party makes a transaction, it may take up to Δ time interval for the transaction to propagate to every other honest party. The existence of network delays allows adversaries to make a double spend attack, which is when an adversary can spend the same UTXO more than once.

1.2 Double Spend

The adversary can achieve double spending by signing and broadcasting two different transaction that spends the same UTXO. Let's call these tx_1 and tx_2 (See Figure 1). Now, notice that if it takes a maximum of Δ for a transaction to be broadcasted to every other honest note, then the honest nodes could receive these two transactions in different orders, as long as both transactions are broadcasted within a time span Δ. For example, honest node B could receive tx_1 first and validate that transaction, and then reject the tx_2 that comes later because an outpoint of tx_2 is no longer in the UTXO set. On the other hand, honest node C could receive the transactions in the opposite order, and validate tx_2 whilst rejecting tx_1 as invalid. As a result, the ledgers $L_B \neq L_C$, and there will be disagreement about ownership.

What are some simple solutions that could solve this double spend problem? One idea is to just cancel double spends, since no honest node would ever double spend. We would cancel both transactions if double spend occurs. However, consider the scenario where an adversary first buys

[1]Interestingly, it is not necessary to require that every node agree on the order of transactions, because there is only one way to draw a construct a UTXO graph out of any valid set of transactions anyway.



tx_1, spending the UTXO originating from A
tx_2, also spending the UTXO originating from A
1
1
1
1
1
A
B
C

Figure 1: Under our current scheme, the honest node cannot distinguish whether tx_1 or tx_2 is valid, if both are broadcasted within the network delay Δ

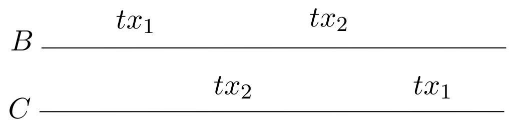


Figure 2: A possible configuration of transaction arrival times that make the protocol vulnerable to double spends, assuming both transactions reach both parties within time Δ; B and C disagree whether tx_1 or tx_2 came first, i.e. which one is valid.

an item in one transaction, and goes on to double spend the money used in that transaction after every honest node receives the first transaction. When both transactions are later cancelled, the adversary still owns that item, so the seller loses money in that case.

A second idea is to accept the first transaction that is received and cancel the second transaction. However, as previously discussed, different nodes may receive the transactions in different orders if they are broadcasted within the time span Δ. As such, each node will believe that the first transaction they received is valid, and since the order of transactions cannot be guaranteed, the ledgers disagree in this case as well.

A third idea is to set up a time window. If both transactions are received within the window, then we cancel both transactions. Otherwise, we accept the first transaction and cancel the second transaction that lies outside of the time window. Ideally, the time window would allow all honest nodes to receive the first transaction, before the second transaction is then broadcasted. However, consider the scenario where the adversary broadcast the second transaction tx_2 at time t just before the end of the time window, such that $t + \Delta$ lies outside of the time window. In this case, some nodes would recieve tx_2 within that time window, and other nodes would receive it outside of the time window. Hence, there will be disagreements once again.

It is clear that none of our preliminary ideas work, and to adequately defend against a double spend attack, we need to introduce the idea of a Block.

2 Blocks and Blockchains

2.1 Virtues of Ledgers

There are two virtues of ledgers that should be satisfied:

- **Safety:** honest parties should agree and achieve consensus.
- **Liveness:** transactions created by honest parties are added to the ledgers of honest parties "soon".

As observed from previous examples, double spends occur because network delays can cause disagreements in the arrival time of different transactions. If somehow transactions could only be created in intervals greater than the network delay parameter Δ, there would be no disagreements because each transaction would have already been broadcasted to all honest parties before the next one gets broadcasted. This gives rise to safety. However, requiring such a delay between transactions means that there will be less transactions being broadcasted every time period, and as such it may be the case that the liveness property is not satisfied. This presents a tradeoff between safety and liveness: as Δ increases, there is less certainty for liveness to be guaranteed.

2.2 Blocks

One way to satisfy both the virtues of safety and liveness is to group multiple transactions together and then broadcast them as "blocks", rather than individually. Each block, denoted $\bar{x} = (tx_1, tx_2, \dots)$, contains a sequence of transactions in any order that respects the topological sorting of the corresponding UTXO graph.

In this model, double spends can only occur if the two transactions that creates the double spend exist in 2 different blocks that arrives in different order to different honest parties, similar to the previous example with transactions. If the two double spending transactions appeared within the same block, then there is no valid UTXO graph for the transactions of that block, and so the entire block is invalid. If the two double spending transactions appear in two blocks that are further than Δ apart, then the first transaction would have been validated by all honest parties, and so the second transaction fails to double spend.

Hence, all we need to do to ensure protection against double spending is to ensure that each **block**—rather than each transaction—is rare, which preserves safety while ensuring a higher level of liveness.

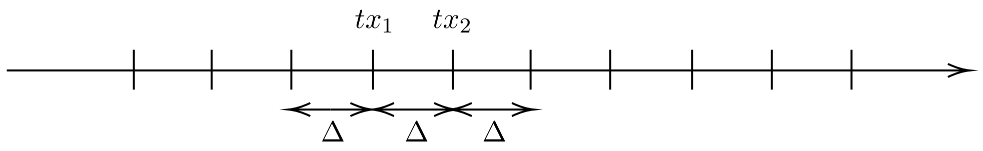


Figure 3: In an ideal scenario, each blocks are broadcasted exactly Δ away from each other, so in a double spend we can always accept tx_1 and reject tx_2

2.3 Proof of Work and Mining

To ensure that the ability to create blocks is rare (i.e. that one can only be created after Δ has passed since the last block), we devise a scheme to issue rare tickets for creating a block. To do so, we will introduce the concept of Proof of Work.

Definition 2.1. Proof-of-Work Equation. $H(B) \leq T$ [2]

Here, T is called the target. In order to solve the Proof-of-Work equation, one would have to find a value of B such that $H(B) \leq T$.

Definition 2.2. Random Oracle Assumption. Output of H is distributed uniformly at random.

Under the Random Oracle Assumption, notice that the time it would take to find a bruteforce solution to a Proof-of-Work equation depends on the value of T. When T is larger, it would be easier to find B such that $H(B) \leq T$, and vice versa. In particular, $Pr[H(B) \leq T] = \frac{T}{2^\kappa}$ for any arbitrary B.

In the network, T is fixed and hard-coded for honest parties. The value is chosen such that the time it takes for a node in the network to find a solution to the Proof-of-Work equation is around Δ.

Now, notice that if we allow honest parties to pick whatever B they want, then as soon as the first valid value of B is found, each node can just trivially reuse that value instantaneously, and safety is lost again. As such, in the network, we require that $B = \delta || \bar{x} || ctr$, where $||$ means concatenation and where δ is the hash of the previous block $H(B')$. Both δ and $\bar{x}$ are fixed, and ctr is a number to be found that solves the Proof-of-Work equation. In other words, when nodes are trying to solve the Proof-of-Work equation, they are trying different values of *ctr* until finding a satisfactory one. The time it takes the first node to find a satisfactory *ctr* for their block will be around δ. The process of finding a bruteforce solution to the Proof-of-Work equation is called mining.

2.4 Blockchains

The previous block hash, δ, is a part of B because we want to ensure "freshness", which means that only blocks that are recently created should be able to be broadcasted. Consider the scenario where an adversary mines multiple blocks, but does not broadcast them immediately. Then, at a later point in time, the adversary broadcast all of these blocks at the same time. Since there is not a Δ interval between the time these blocks are broadcasted, the problem of double spend arises again. There could be disagreements in the ledgers of the honest nodes. In order to prevent this, we require each block to point to the most recent known block. By including δ as a part of B, we ensure that blocks that were mined beforehand are no longer valid, since the adversary could not have predicted δ before the most recent block was broadcasted. As a result, we can guarantee that blocks will be roughly be broadcasted in regular time intervals. Since blocks are all connected to each other in this way, this chain of block is called the *blockchain.*

[2]This scheme was derived in 1993 by Cynthia Dwork and Moni Naor in the seminal paper "Pricing via Processing or Combatting Junk Mail".

2.5 Genesis Block

The Genesis Block is the first block in a blockchain. It contains real world data to anchor time, in order to ensure that it is impossible to pre-mine. If the hash of the Genesis Block is known before it is broadcasted, then the adversary could pre-mine multiple blocks beforehand (before the network is even available to the rest of the world). Then, when the Genesis Block is broadcasted, the adversary would be able to broadcast multiple blocks within a short time span. Hence, it is necessary for the Genesis Block to contain real world data to anchor time. For example, Bitcoin genesis block quotes the January 3 2009 The Times headline.

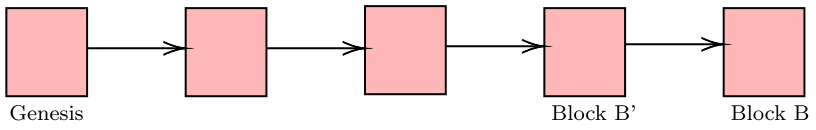


Figure 4: To mine block B, the miner runs the Proof of Work Equation with B = $H(B')||\bar{x}||ctr$

2.6 Honest Party Block Generation Algorithm

Putting it all together, an honest party (which we can assume to be every honest node in the network for now) follows the following algorithm to create a valid block. First, it collects any amount of transactions from the network and add them to a mempool, which is a set of validated but unconfirmed transactions [3]. Then, it writes $B = \delta||\bar{x}||ctr$, and then solves the Proof-of-Work Equation. In other words, it finds a value of ctr (which is also known as the nonce) such that $H(B) \leq T$. If the honest party is successful, it will broadcast the mined block to the network. Otherwise, if another block has been found and received by the honest party first, it will validate the new block, update it's UTXO by removing all the transactions that were spent in the newest block, and update the freshest block in its memory. As such, B would necessarily change, and the honest party will have to solve a new Proof-of-Work equation.

[3]The number of transactions it collects will be formalized later