CSE446: Blockchain & Cryptocurrencies

Lecture – 8: Bitcoin-2

Agenda

- Bitcoin components
  - Node & Network
  - Bitcoin Blockchain
    - Transaction
    - Block
    - Blockchain

Bitcoin P2P network

- Bitcoin nodes communicate in a decentralised fashion, meaning that no single entity or node is superior, all nodes are equal
  - Ad-hoc protocol (runs on TCP port 8333)
  - Ad-hoc network with random topology
- New nodes can join at any time
- Forget non-responding nodes after 3 hr

Bitcoin Basics- Gallersdörfer, U., Holl, P., & Matthes, F. (2020). "Blockchain-based Systems Engineering". Lecture Slides. TU Munich.

Bitcoin P2P network

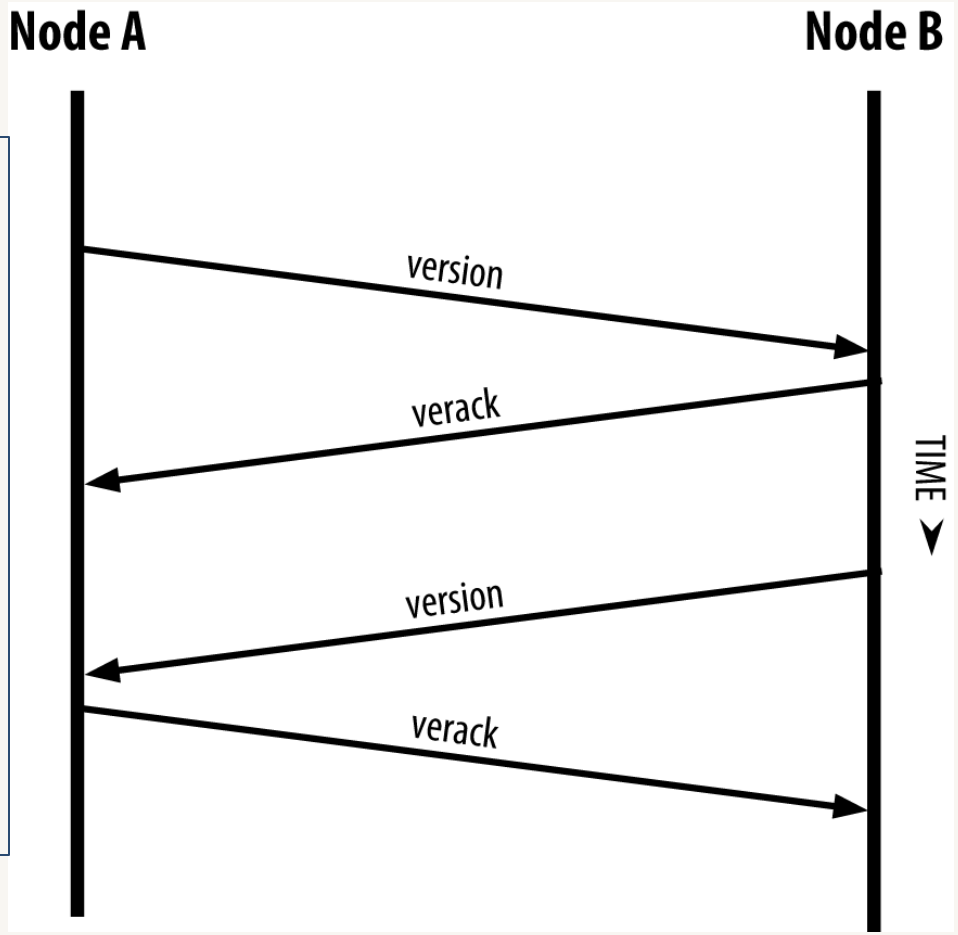
- To communicate, they need to have clear rules
  - How to find other nodes (bootstrapping)
  - How to send and receive transactions
  - How to send and receive blocks
  - How to sync the blockchain
- The basic network uses a peer-to-peer gossip protocol for
  - Node discovery, node status maintenance
  - Messages about new blocks or transactions

Node discovery/bootstrapping

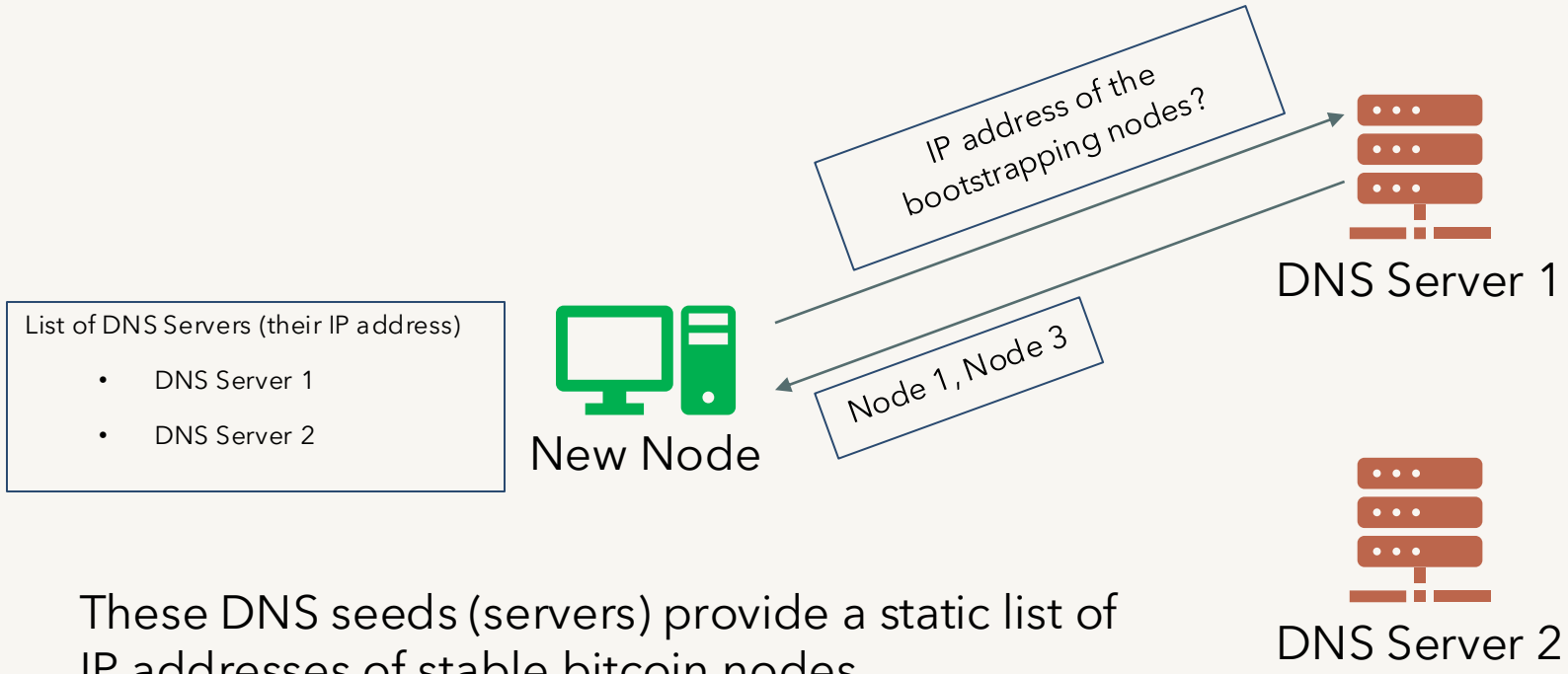
- Adding a new node into the network is called bootstrapping
- The new node needs to discover other nodes in the network to connect to the P2P network
- How does it know who to connect to?
  - Hard-coded DNS-services which offer IP-addresses of nodes
  - Hard-coded seed addresses (last resort)
  - Command-line provided addresses
  - Text-file provided addresses

Node discovery/bootstrapping

- The first message to another Bitcoin peer is the version message
- Using version each node check if the other node is compatible
- If compatible, the other node sends the version acknowledgement (verack) message

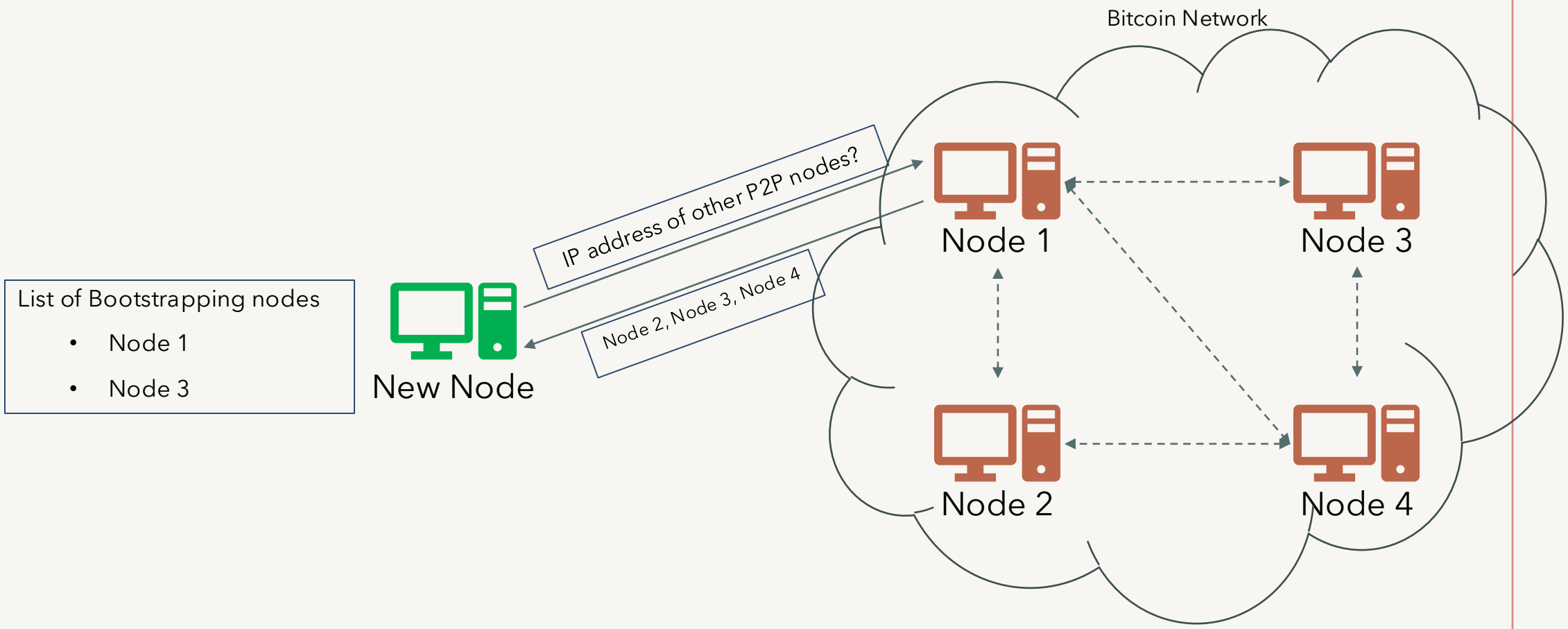


Node discovery/bootstrapping

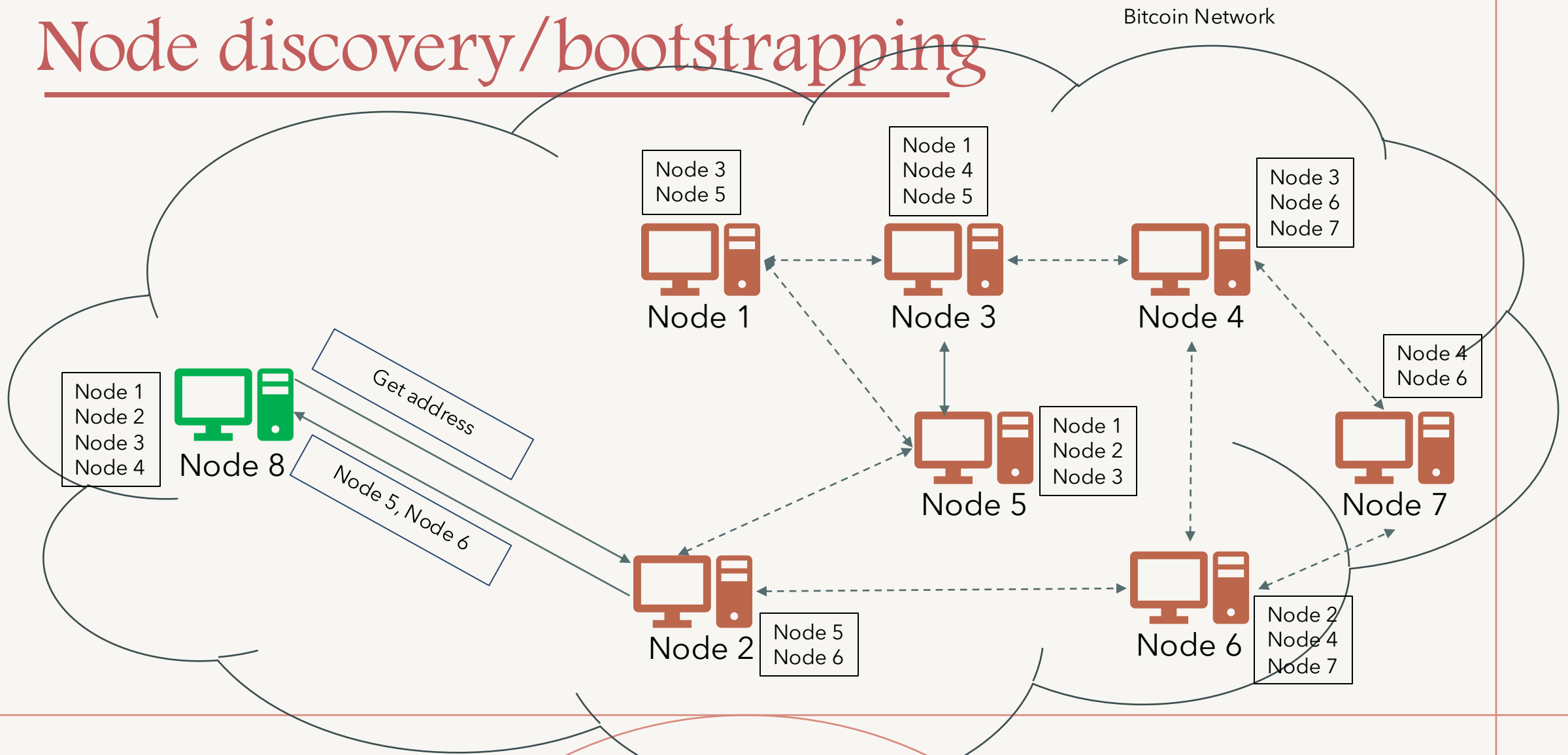


These DNS seeds (servers) provide a static list of IP addresses of stable bitcoin nodes

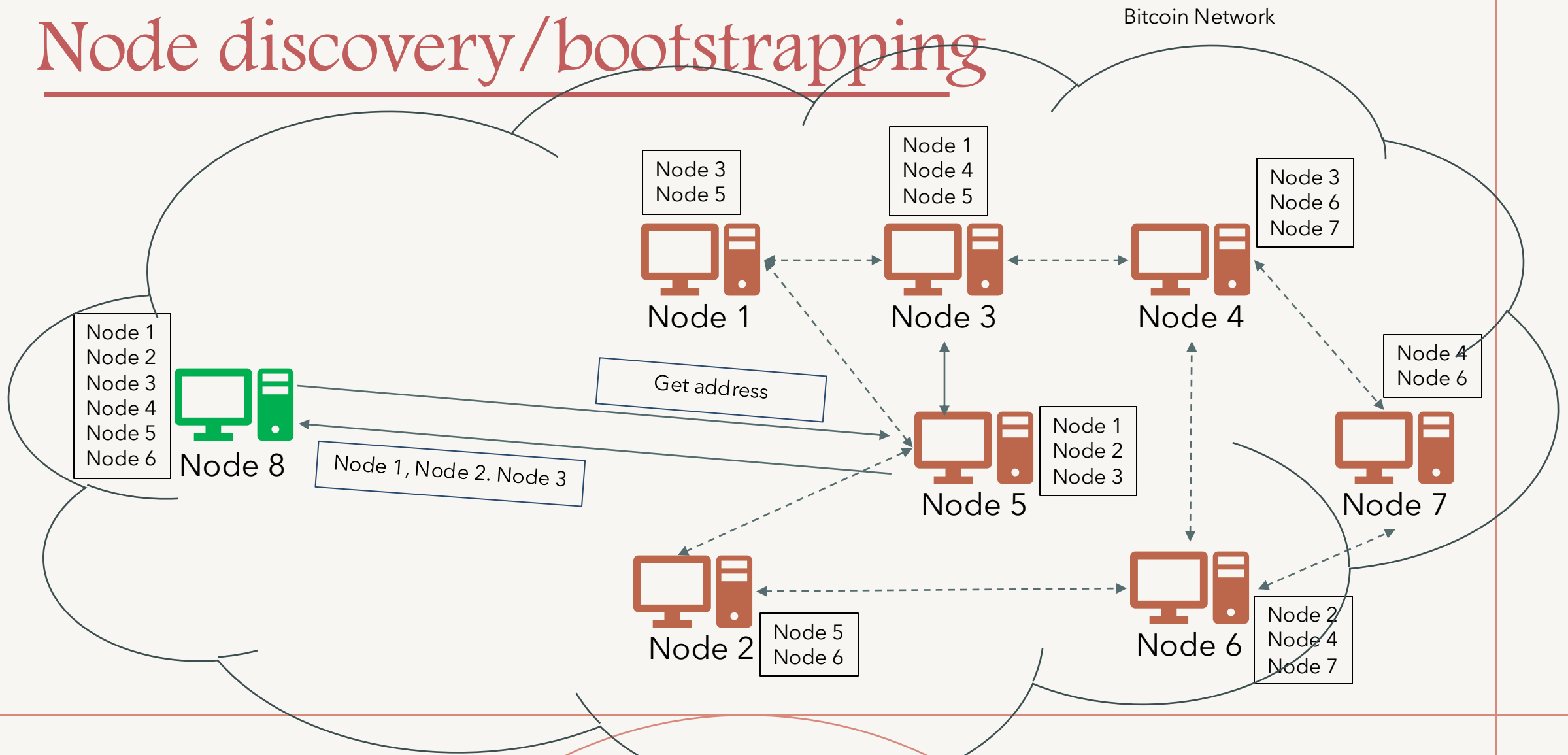
Node discovery/bootstrapping



Node discovery/bootstrapping



Node discovery/bootstrapping



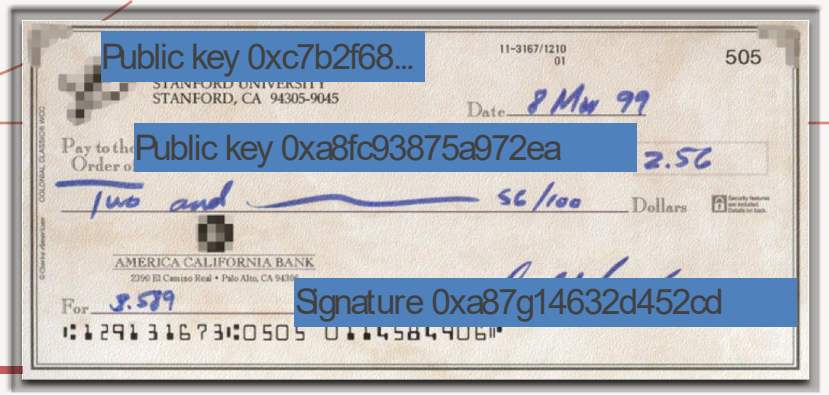
Bitcoin blockchain

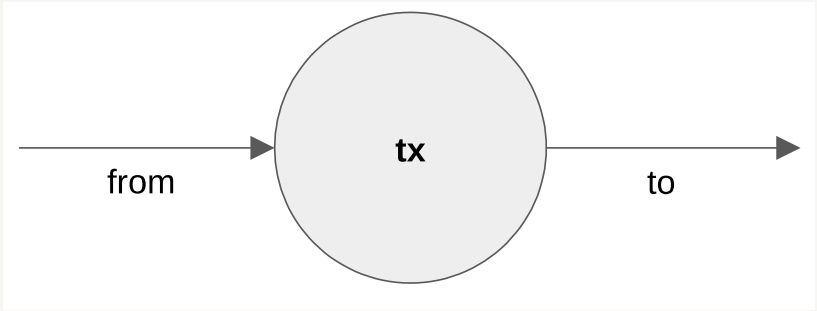
- There are three different things to understand
  - Transaction
  - Block
  - Blockchain

Transaction

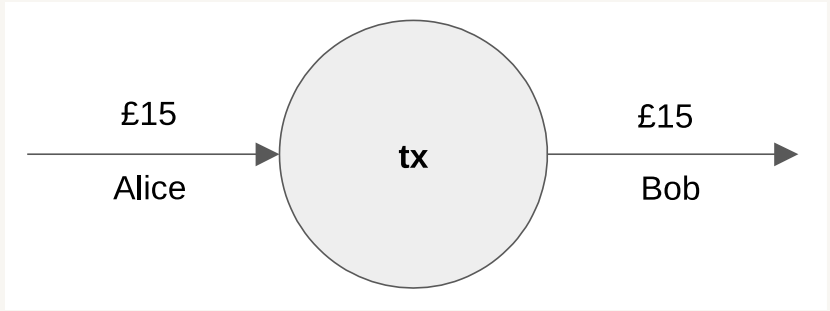
- Transactions are the most important part of the bitcoin system
- Everything else in bitcoin is designed to ensure that transactions can be created, propagated on the network, validated, and finally added to the global ledger of transactions (the blockchain)
- Transactions are data structures that encode the transfer of values between participants in the bitcoin system
  - A transaction transfers money from someone to somebody else
- Each transaction is a public entry in bitcoin's blockchain, the global double-entry bookkeeping ledger
  - an entry that affects at least two different accounts
- All transactions are public
- Everybody can see all transactions in a blockchain explorer

Transaction

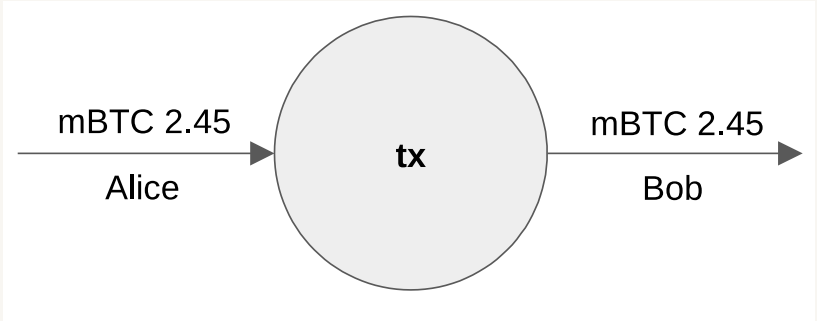




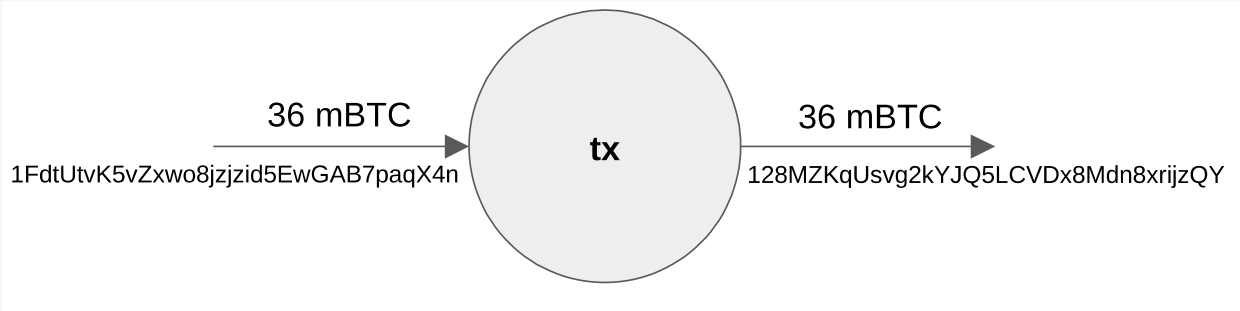
Abstract format of any transaction



A traditional transaction

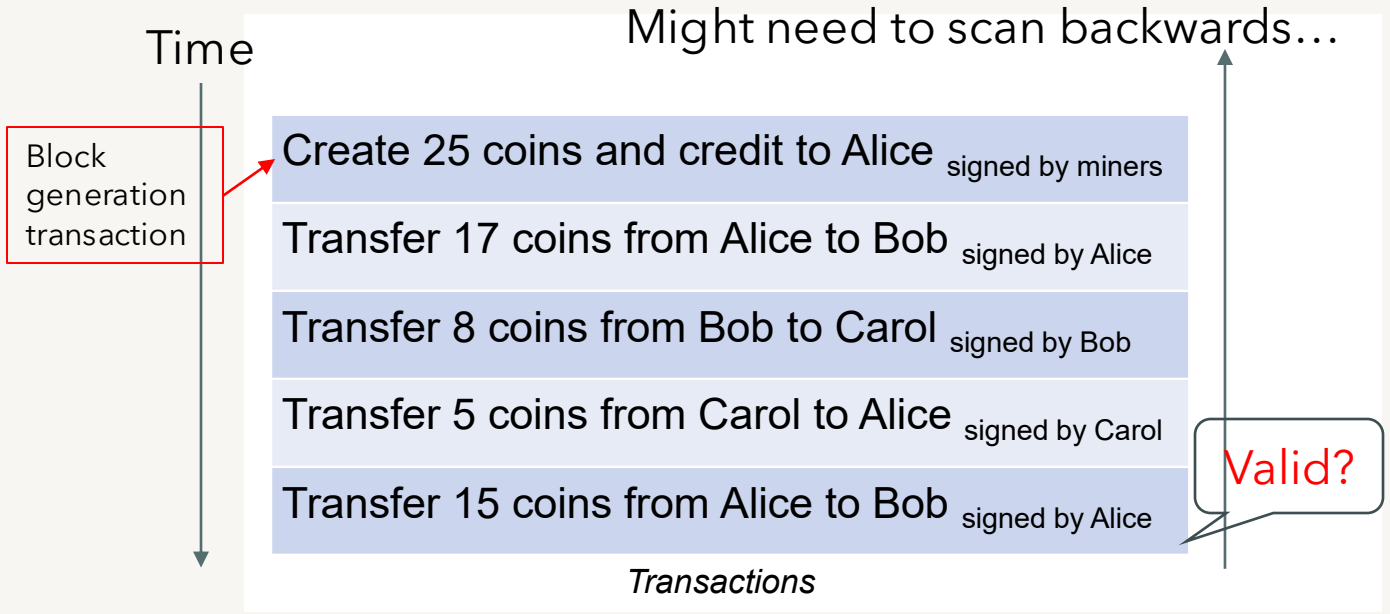


Abstract format of a bitcoin transaction



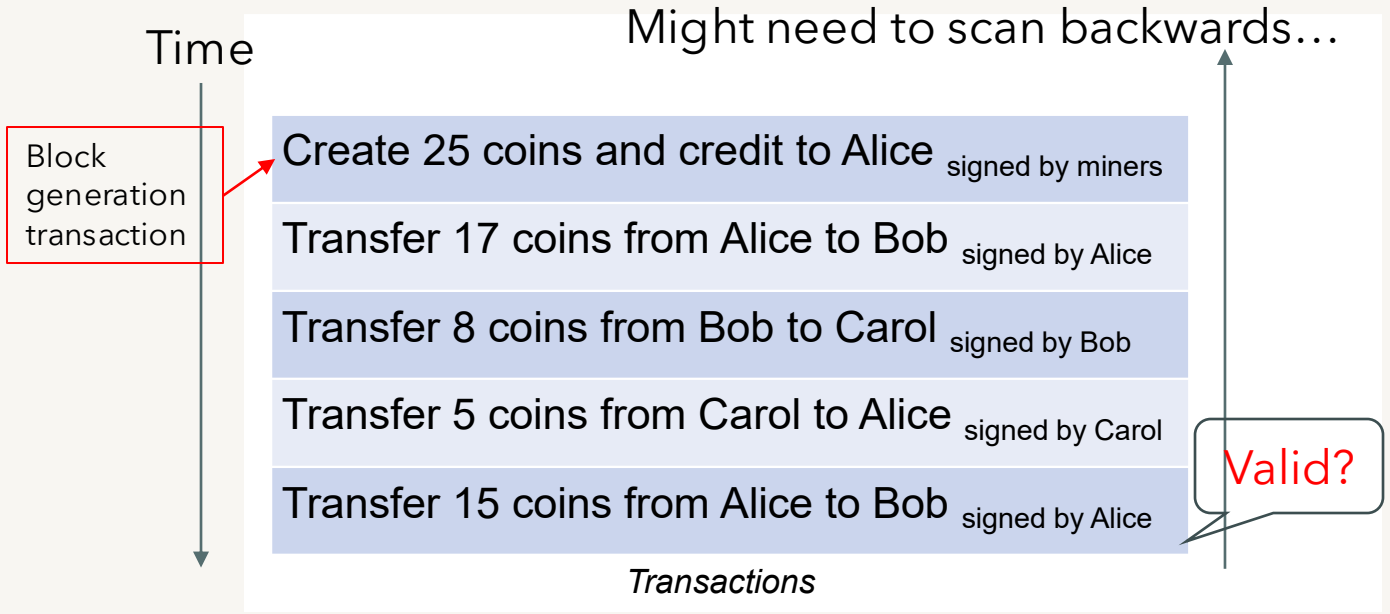
A bitcoin transaction

Account/transaction based Ledger?



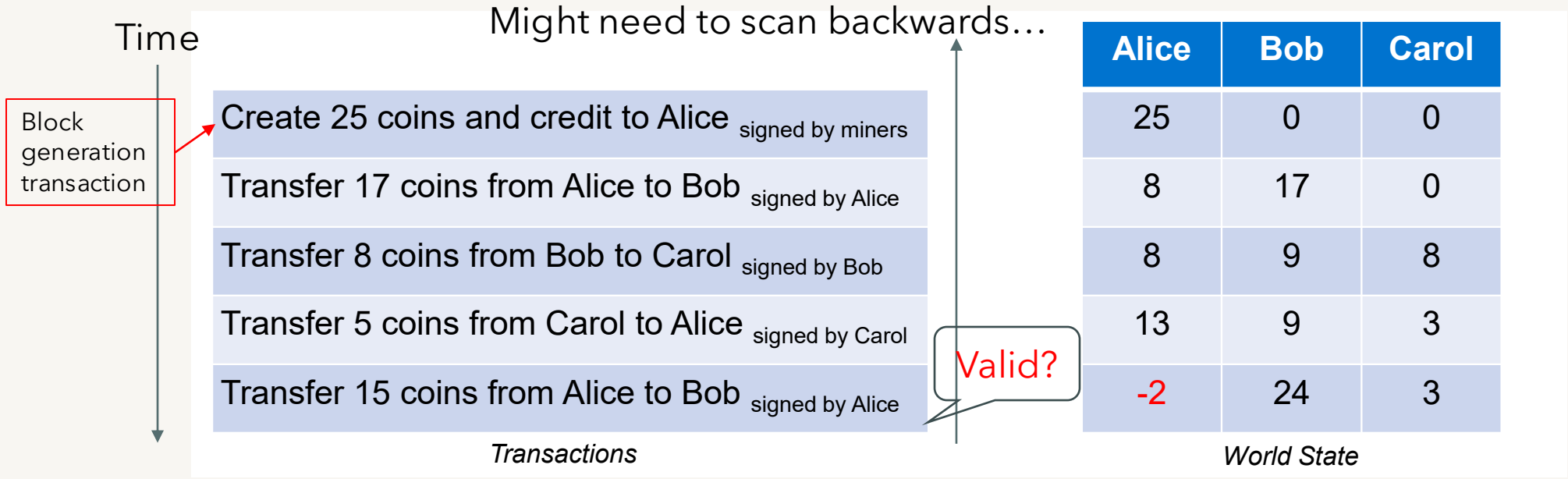
- *Intuitively*: At first, we consider Bitcoin to use an account-based ledger. However, an account-based approach takes a lot of effort to track the balances of every account
- In an account-based ledger, transactions can transfer arbitrary amounts of coins between accounts
- Transactions lead to a "world-state" of accounts and account balances

Account/transaction based Ledger?



- To validate a certain transaction, you might need to track very old transactions
- Since you do not know which old transaction, you need track each of the previous transactions one by one until you find the desired ones

Account/transaction based Ledger?

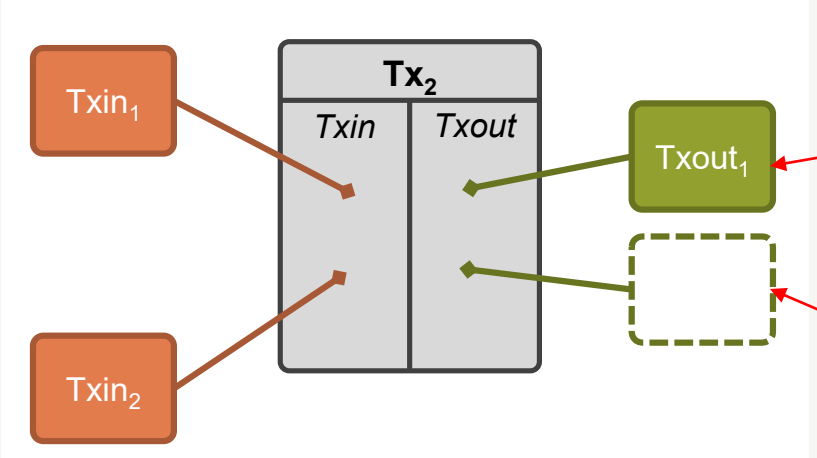


- One option is to maintain the world state in a separate database
  - Remember, in the world state you store the account balance
- That would require to maintain two separate databases: world state and transactions

Account/transaction based Ledger?

- Bitcoin’s solution: a transaction-based ledger
- By using a transaction-based ledger, Bitcoin enables wallet owners to define conditional transactions using Bitcoin Script

Transaction Based Ledger



Output not consumed, not used as inputs in other transactions These are known as **UTXO (Unspent Transaction Output)**

Output consumed, used as inputs in other transactions. These are known as **STXO (Spent Transaction Output**)

- Transactions (Tx) have a number of inputs and a number of outputs
  - Inputs (Txin): Former outputs, that are being consumed
  - Outputs (Txout): New outputs transferring the value
- In transactions (coinbase transaction) where new coins are created, no Txin is used (no coins are consumed)
- Each transaction has a unique identifier (TxID). Each output has a unique identifier within a transaction
- We refer to them (in this example) as *#TX[#txout]*, e.g., 1[1], which is the second Txout of the second transaction

Transaction Based Ledger

This is known as a coinbase transaction

Create 25 coins and credit to Alice	signed by miners
Transfer 17 coins from Alice to Bob	signed by Alice
Transfer 8 coins from Bob to Carol	signed by Bob
Transfer 5 coins from Carol to Alice	signed by Carol
Transfer 15 coins from Alice to Bob	signed by Alice

Transactions

0	Txin: ∅ Txout: 25.0 -> Alice	signed by the miner/system
1	Txin: 0[0] Txout: 17.0 → Bob, 8.0 → Alice	signed by Alice
2	Txin: 1[0] Txout: Txout: 8.0 → Carol, 9.0 → Bob	signed by Bob
3	Txin: 2[0] Txout: 5.0 → Alice, 3.0 → Carol	signed by Carol
4	Txin: 1[1], 3[0] Txout: 15.0 → Bob, ? → Alice	signed by Alice

Transaction Based Ledger

Create 25 coins and credit to Alice	signed by miners
Transfer 17 coins from Alice to Bob	signed by Alice
Transfer 8 coins from Bob to Carol	signed by Bob
Transfer 5 coins from Carol to Alice	signed by Carol
Transfer 15 coins from Alice to Bob	signed by Alice

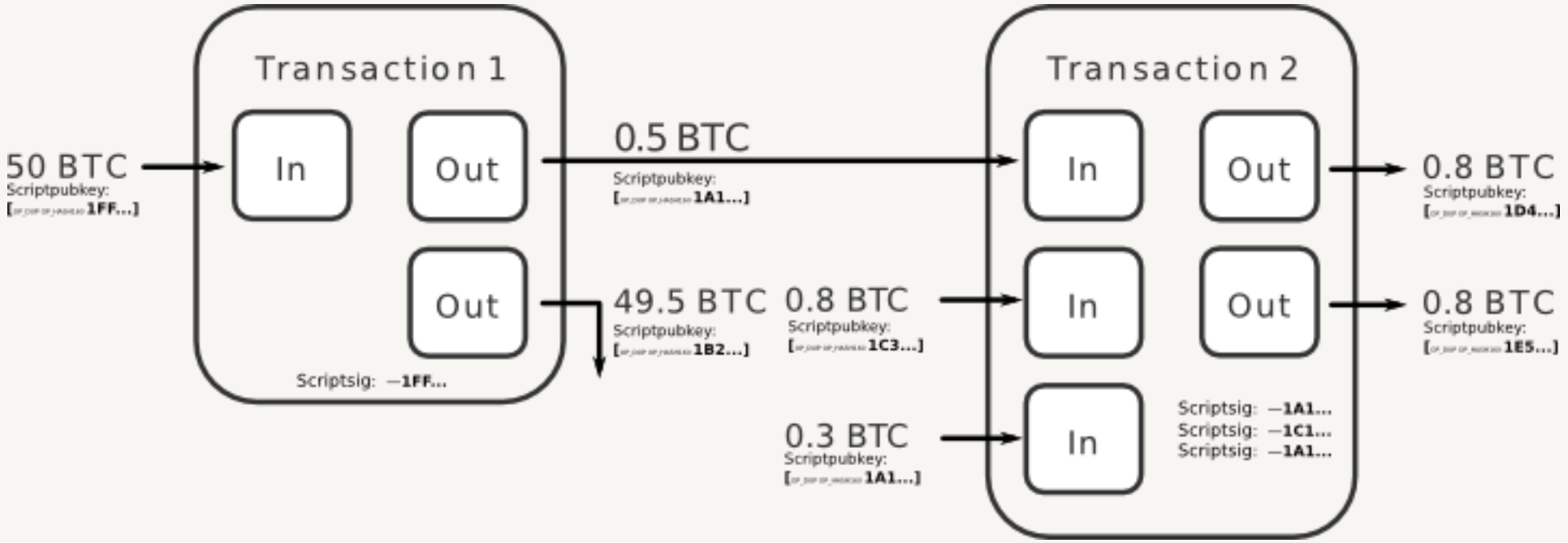
Transactions

Joined payment

0	Txin: ∅ Txout: 25.0 -> Alice signed by the miner/system
1	Txin: 0[0] Txout: 17.0 → Bob, 8.0 → Alice signed by Alice
2	Txin: 1[0] Txout: Txout: 8.0 → Carol, 9.0 → Bob signed by Bob
3	Txin: 2[0] Txout: 5.0 → Alice, 3.0 → Carol signed by Carol
4	Txin: 1[1], 2[0] Txout: 15.0 → Bob, ? → Alice signed by Alice

Bitcoin Basics- Gallersdörfer, U., Holl, P., & Matthes, F. (2020). "Blockchain-based Systems Engineering". Lecture Slides. TU Munich.

Connecting different transactions

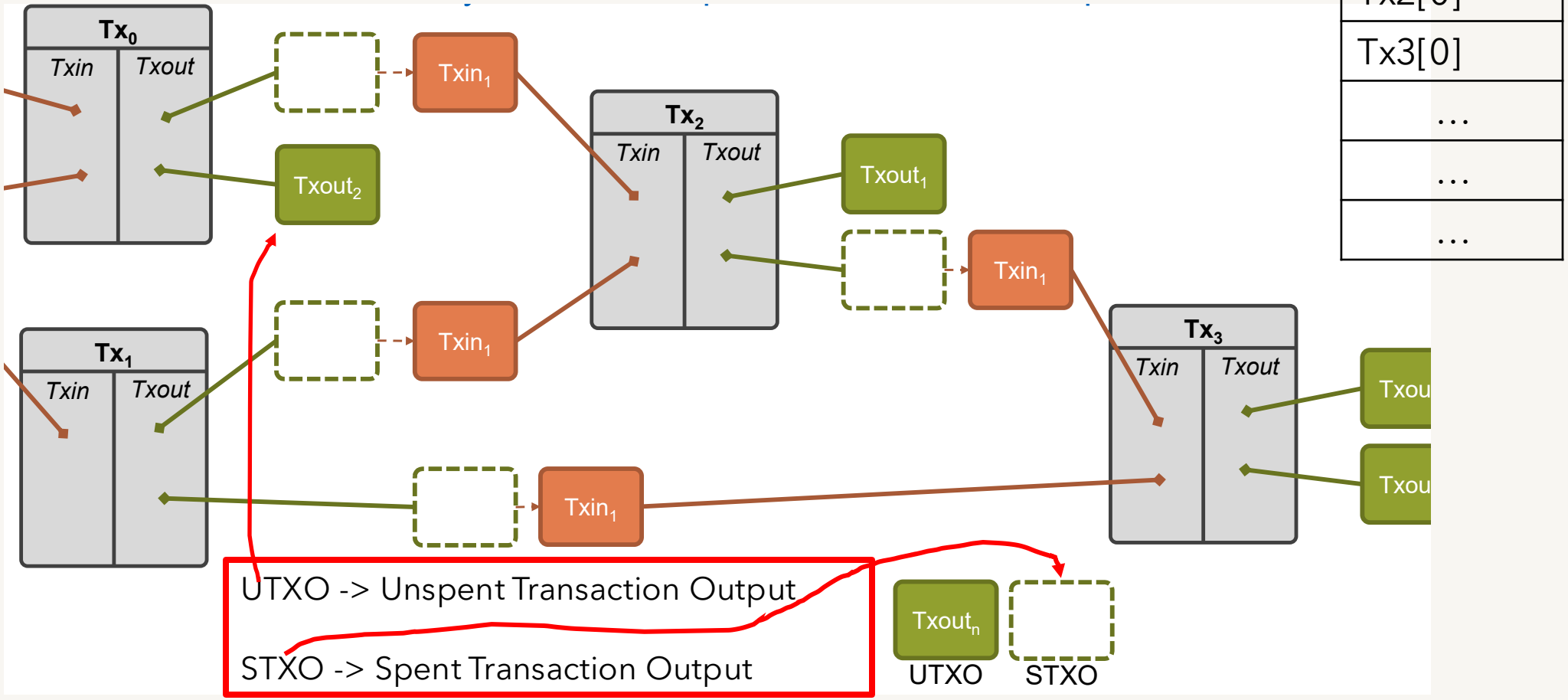


Source: http://s3.amazonaws.com/main-newsbtc-images/2015/11/27120846/Coinjoin.png

Connecting different transactions

All UTXOs are maintained in an UTXO Table

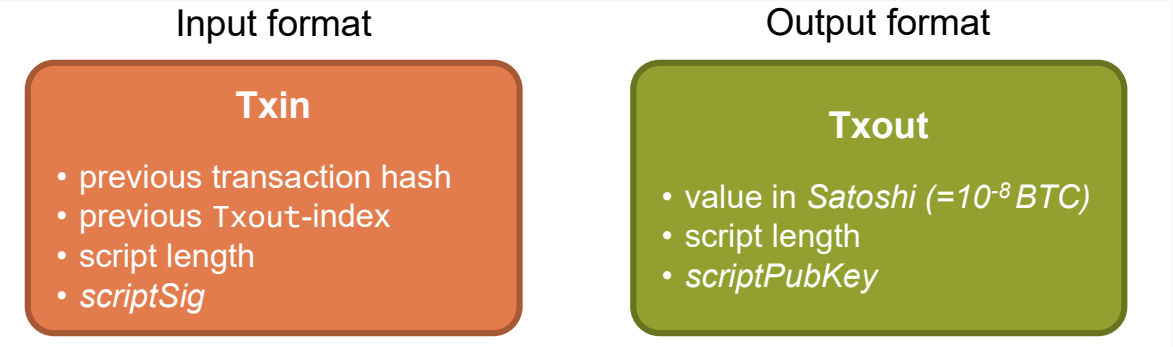
Tx0[1]
Tx2[0]
Tx3[0]
…
…
…

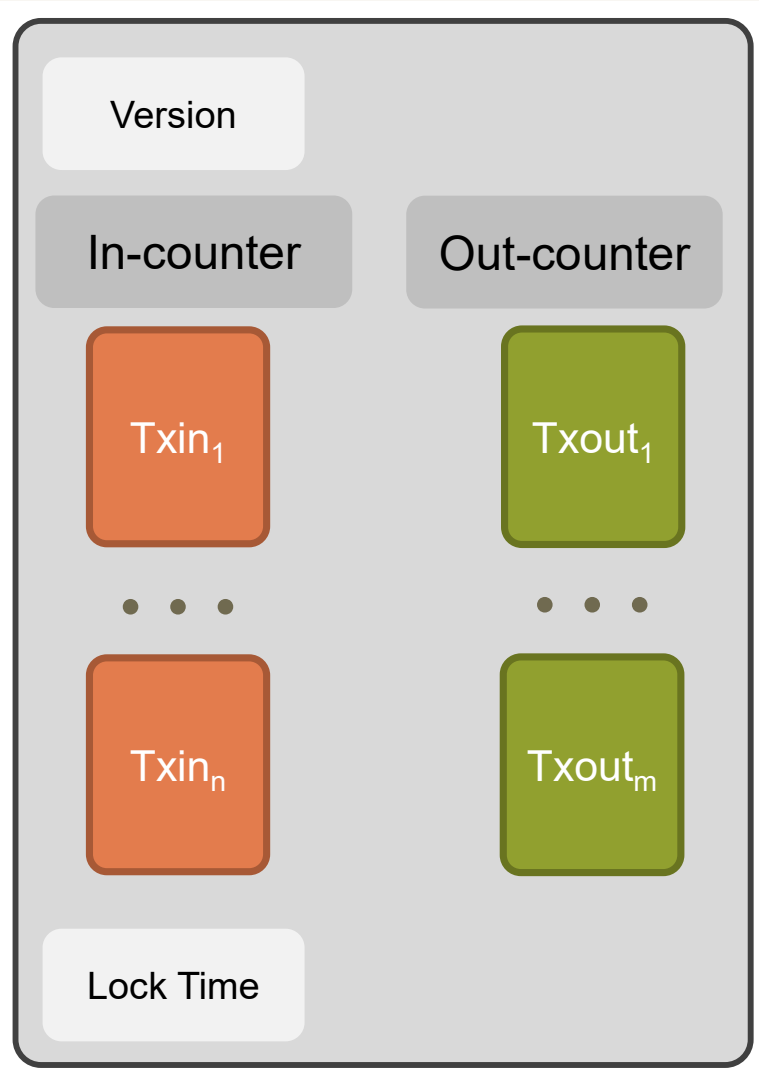


Bitcoin Basics- Gallersdörfer, U., Holl, P., & Matthes, F. (2020). "Blockchain-based Systems Engineering". Lecture Slides. TU Munich.

Transaction

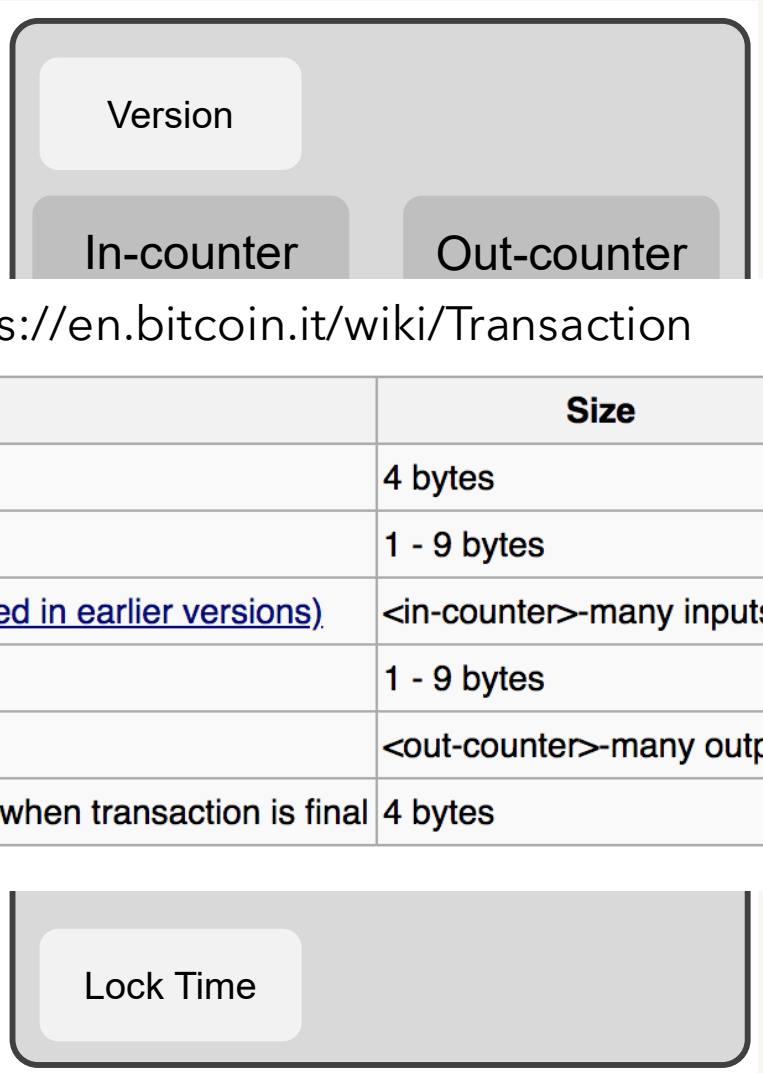
- Each transaction has a list of inputs and outputs
- All inputs reference an existing unspent output or a coinbase transaction
- Inputs and outputs contain scripts (scriptSig, scriptPubKey) for verification and other metadata
- lock_time: is the time at which a particular transaction can be added to the blockchain, 0 means now





Transaction

- All inputs reference an existing unspent output or a



General format of a Bitcoin transaction (inside a block)

https://en.bitcoin.it/wiki/Transaction

Field	Description	Size
Version no	currently 1	4 bytes
In-counter	positive integer VI = VarInt	1 - 9 bytes
list of inputs	the first input of the first transaction is also called "coinbase" (its content was ignored in earlier versions)	<in-counter>-many inputs
Out-counter	positive integer VI = VarInt	1 - 9 bytes
list of outputs	the outputs of the first transaction spend the mined bitcoins for the block	<out-counter>-many outputs
lock_time	if non-zero and sequence numbers are < 0xFFFFFFFF: block height or timestamp when transaction is final	4 bytes

- previous transaction hash
- previous `Txout`-index
- script length
- *scriptSig*

- value in *Satoshi* (=10^{-8} *BTC*)
- script length
- *scriptPubKey*

Transaction

It contains the size of the transaction, the number of inputs and outputs, the version and a lock-time. The hash is the transaction ID (TxID) discussed later

An array of all inputs. Each input contains the previous transaction hash (TxID) and the index of Txout. Also a signature *script (scriptSig)* is provided.

An array of all outputs. One output has two fields: the amount of the transferred coins and the scriptPubKey.

metadata

input(s)

output(s)


```
{
 "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
 "ver":1,
 "vin_sz":2,
 "vout_sz":1,
 "lock_time":0,
 "size":404,
 "in":[
  {
   "prev_out":{
    "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
    "n":0
   },
    "scriptSig":"30440..."
  },
  {
   "prev_out":{
    "hash":"7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",
    "n":0
   },
   "scriptSig":"3f3a4ce81...."
  }
 ],
 "out":[
  {
   "value":"10.12287097",
   "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"
  }
 ]
}
```

Transaction output

- An output contains instructions for sending bitcoins
  - Value is the number of Satoshi (1 BTC = 100,000,000 Satoshi) that this output will be worth when claimed
- There can be more than one output, and they share the combined value of the inputs
- If the input is worth 50 BTC but you only want to send 25 BTC,
  - Bitcoin will create two outputs worth 25 BTC: one to the receiver, and one back to you (known as "*change*", though you send it to yourself)
- Any input bitcoins not redeemed in an output is considered a *transaction fee*; whoever generates the block will get it


```
"vout": [
  {
    "value": 0.01500000,
    "scriptPubKey": "OP_DUP OP_HASH160 ab68025513c3dbd2f7b92a94e0581f5d50f654e7 OP_EQUALVERIFY
    OP_CHECKSIG"
  },
  {
    "value": 0.08450000,
    "scriptPubKey": "OP_DUP OP_HASH160 7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG",
  }
]
```


https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch06.asciidoc

Transaction output

- Almost all outputs are spendable chunks, known as UTXO (unspent transaction outputs)
  - The collection of all UTXO is known as the UTXO set or the UTXO table
- The UTXO set grows as new UTXO is created and shrinks when UTXO is consumed
- Every transaction represents a change in the UTXO set
- When we say that a user’s wallet has "received" bitcoin
  - what we mean is that the wallet has detected an UTXO that can be spent with one of the keys controlled by that wallet

Transaction output

- The concept of a balance is created by the wallet application
  - The wallet calculates the user's balance by scanning the blockchain and aggregating the value of any UTXO the wallet can spend with the keys it controls
- Most wallets maintain a database or use a database service to store a quick reference set of all the UTXO they can spend with the keys they control
- Transaction outputs has another component:
  - A cryptographic puzzle that determines the conditions required to spend the output
  - The cryptographic puzzle is also known as a *locking script*, a *witness script*, or a scriptPubKey


THANK YOU
QUESTIONS?