Lecture 6

Mechanics of Bitcoin

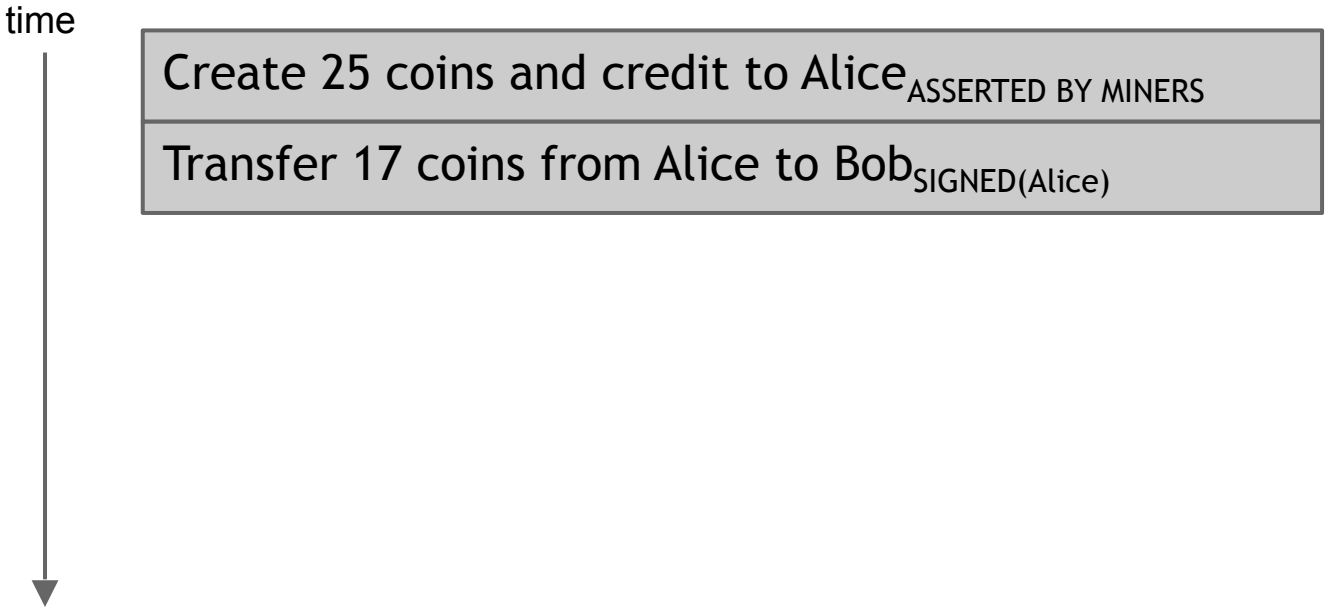
Bitcoin transactions

An account-based ledger (*not* Bitcoin)

Create 25 coins and credit to Alice$_{\text{ASSERTED BY MINERS}}$

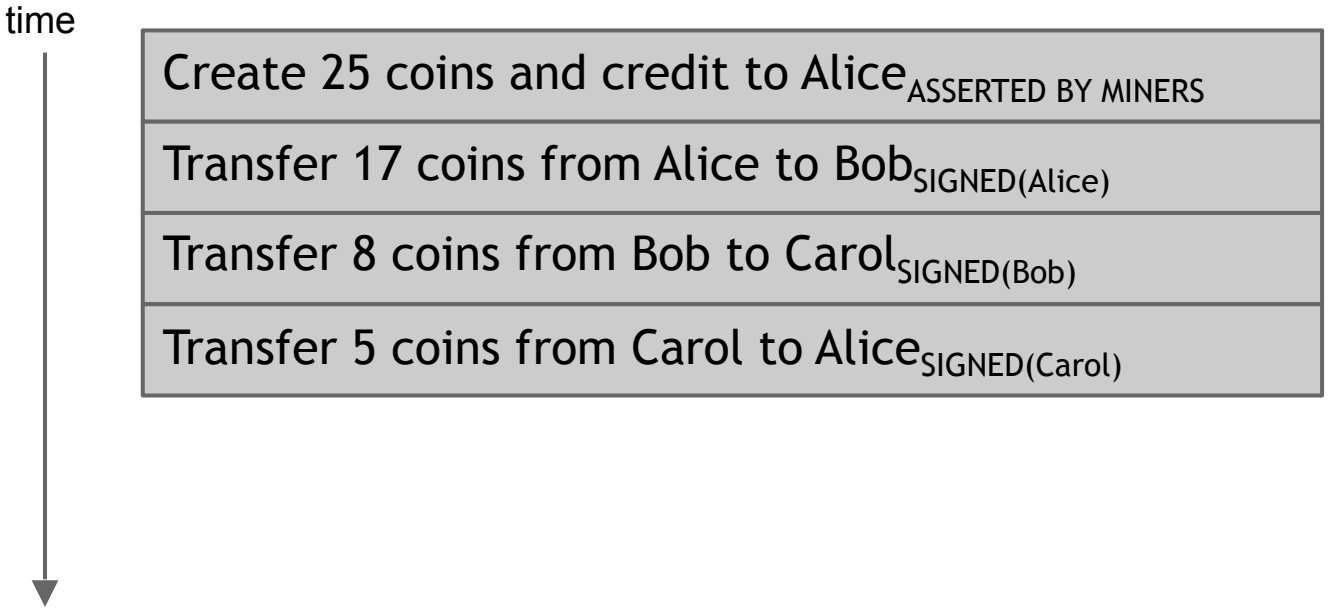
SIMPLIFICATION: only one transaction per block

An account-based ledger (*not* Bitcoin)



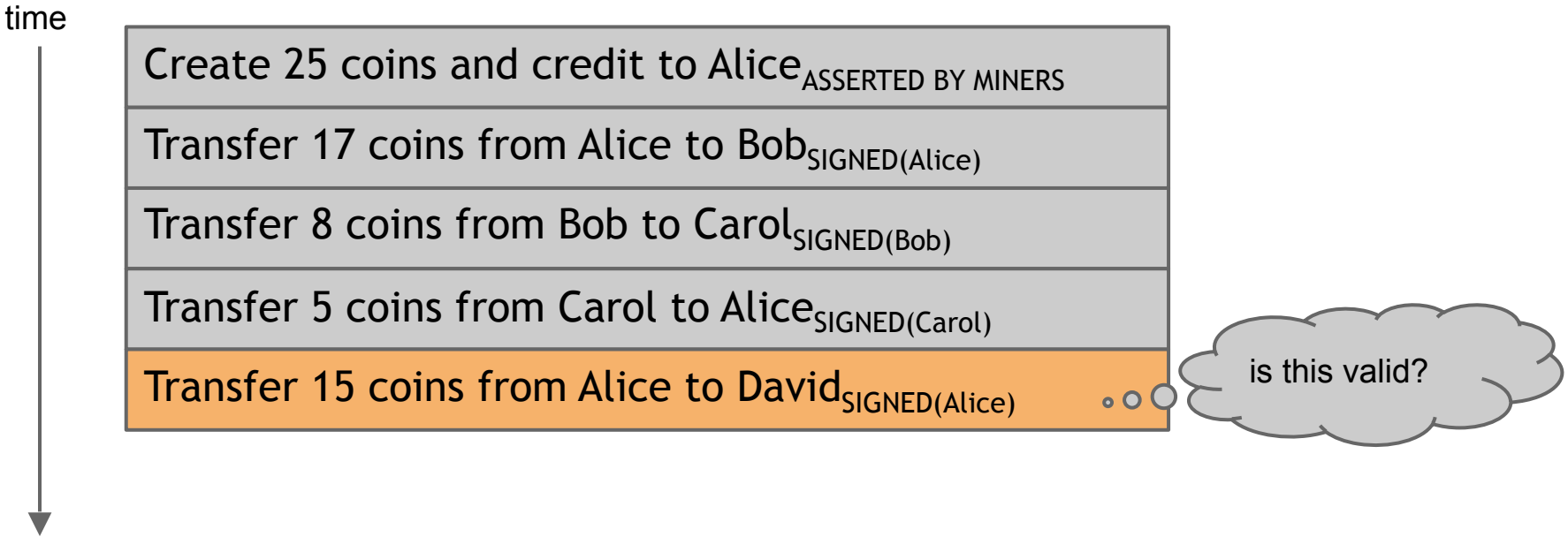
SIMPLIFICATION: only one transaction per block

An account-based ledger (*not* Bitcoin)



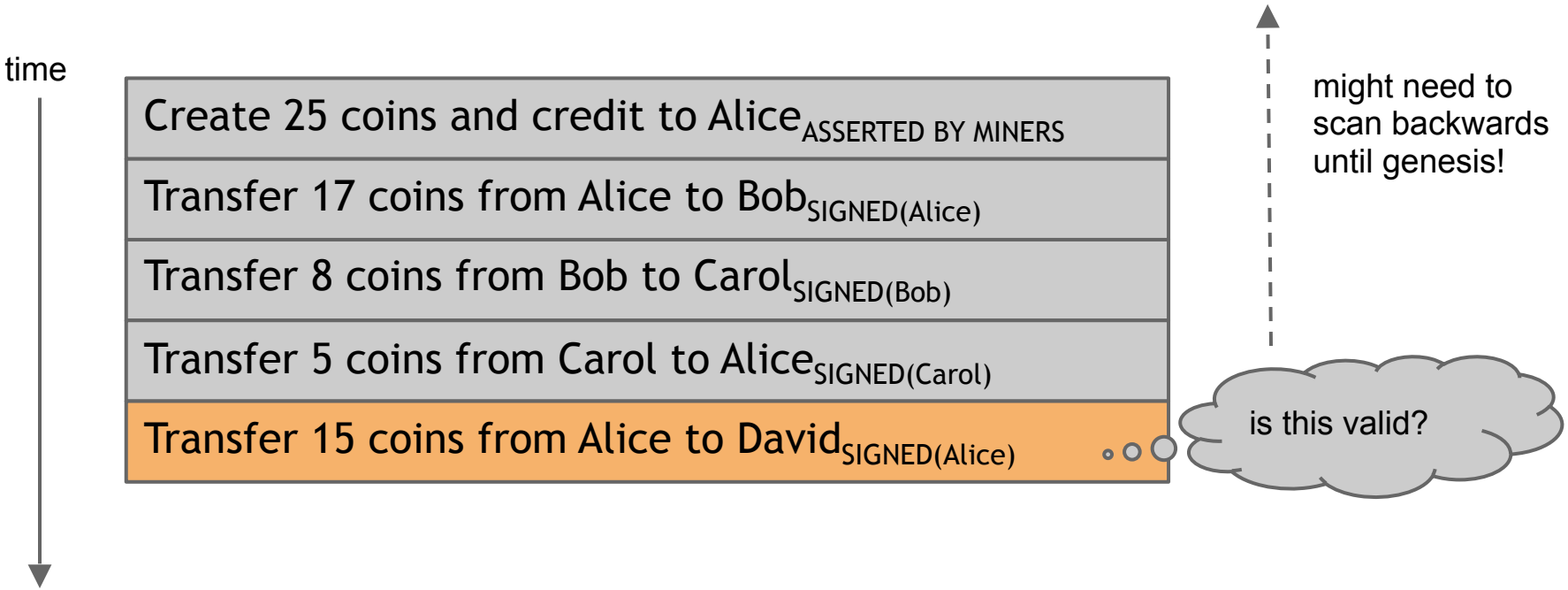
SIMPLIFICATION: only one transaction per block

An account-based ledger (*not* Bitcoin)



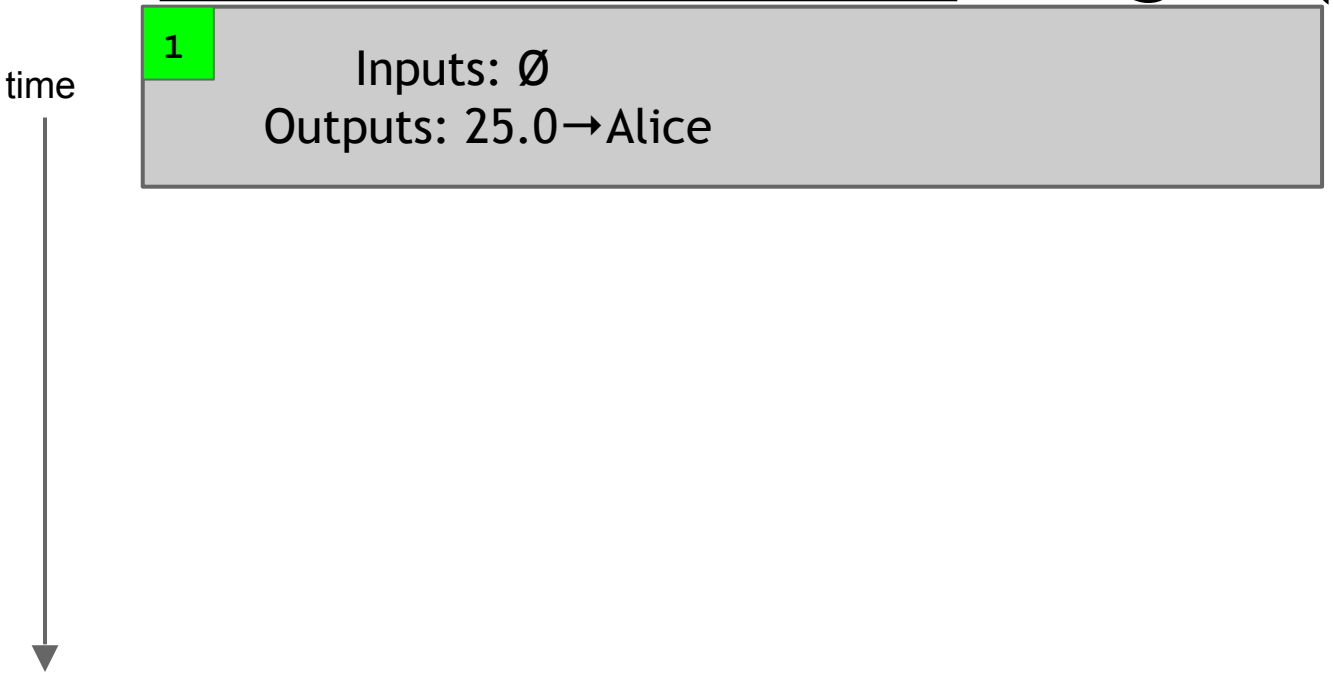
SIMPLIFICATION: only one transaction per block

An account-based ledger (*not* Bitcoin)



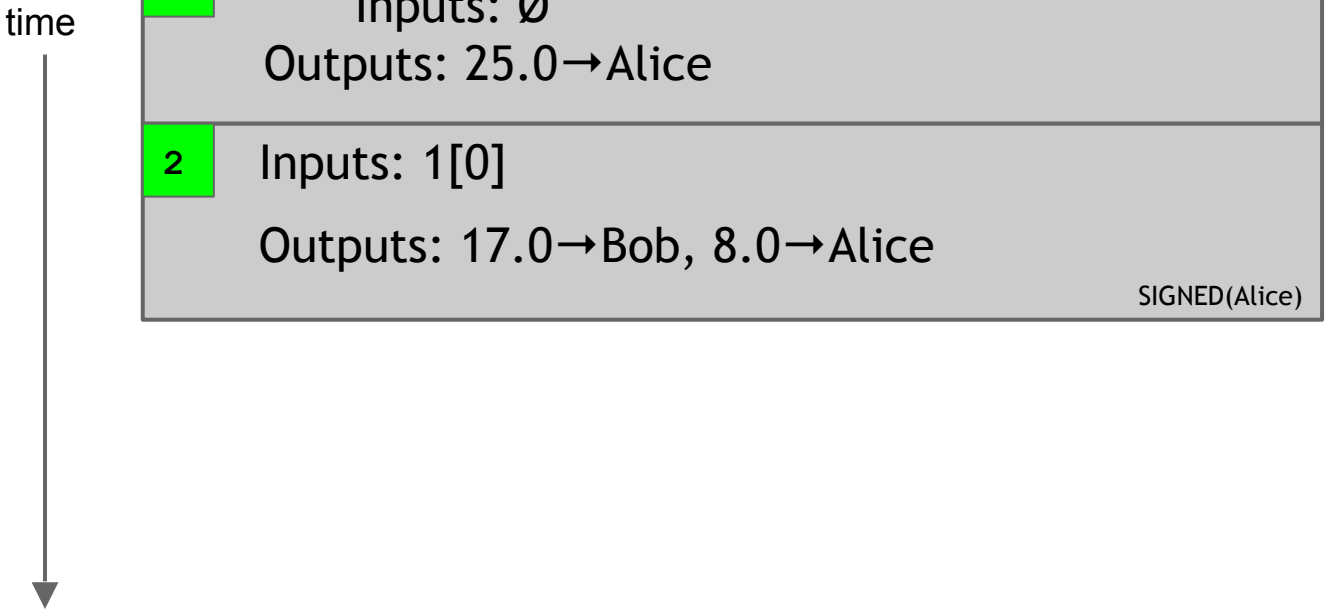
SIMPLIFICATION: only one transaction per block

A transaction-based ledger (Bitcoin)



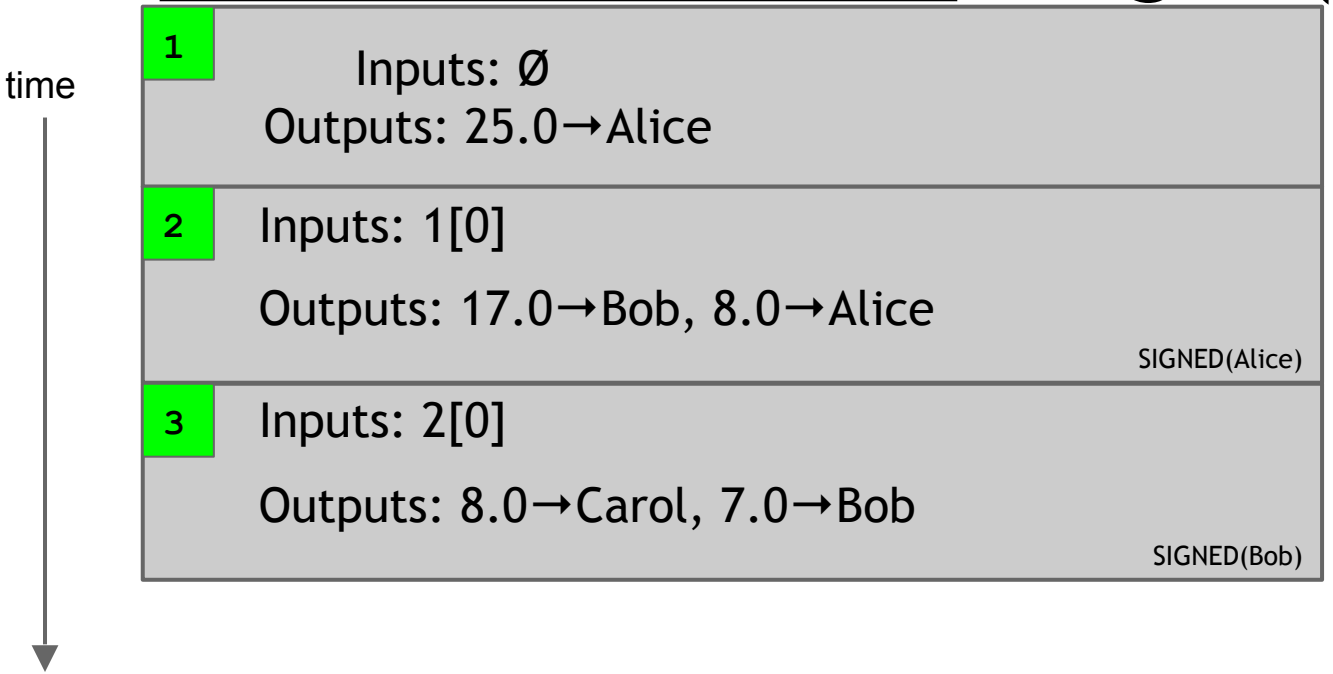
SIMPLIFICATION: only one transaction per block

A transaction-based ledger (Bitcoin)



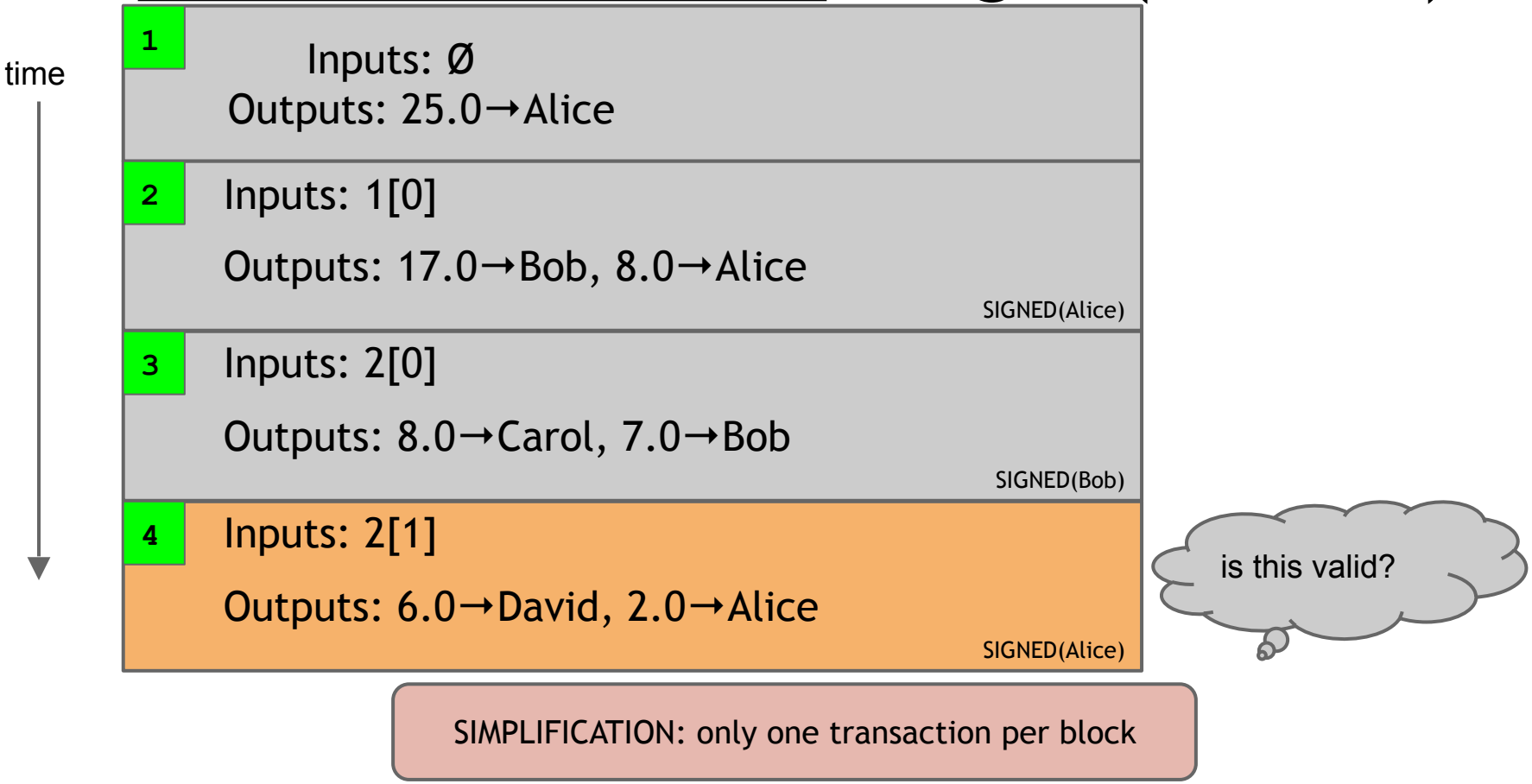
SIMPLIFICATION: only one transaction per block

A transaction-based ledger (Bitcoin)

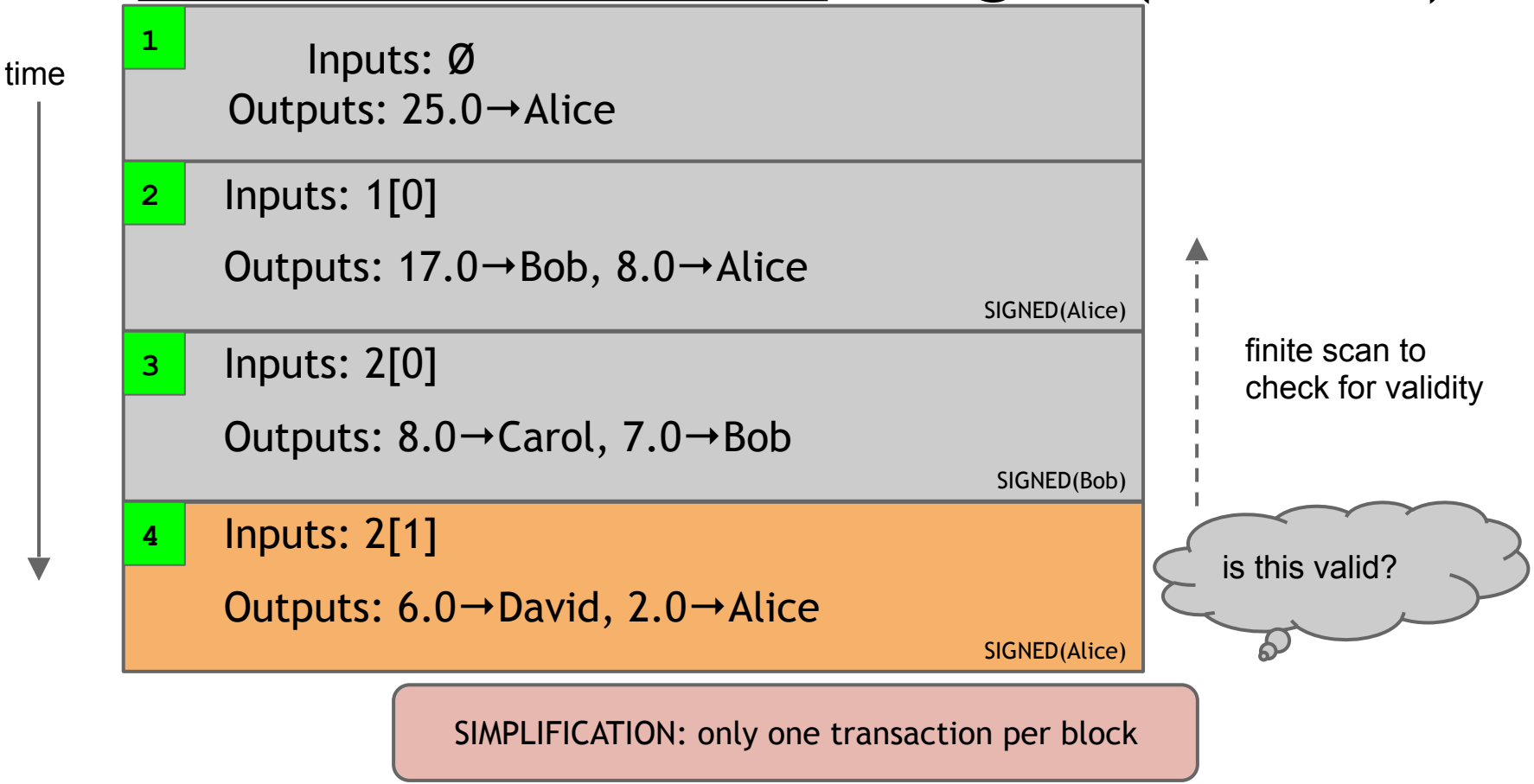


SIMPLIFICATION: only one transaction per block

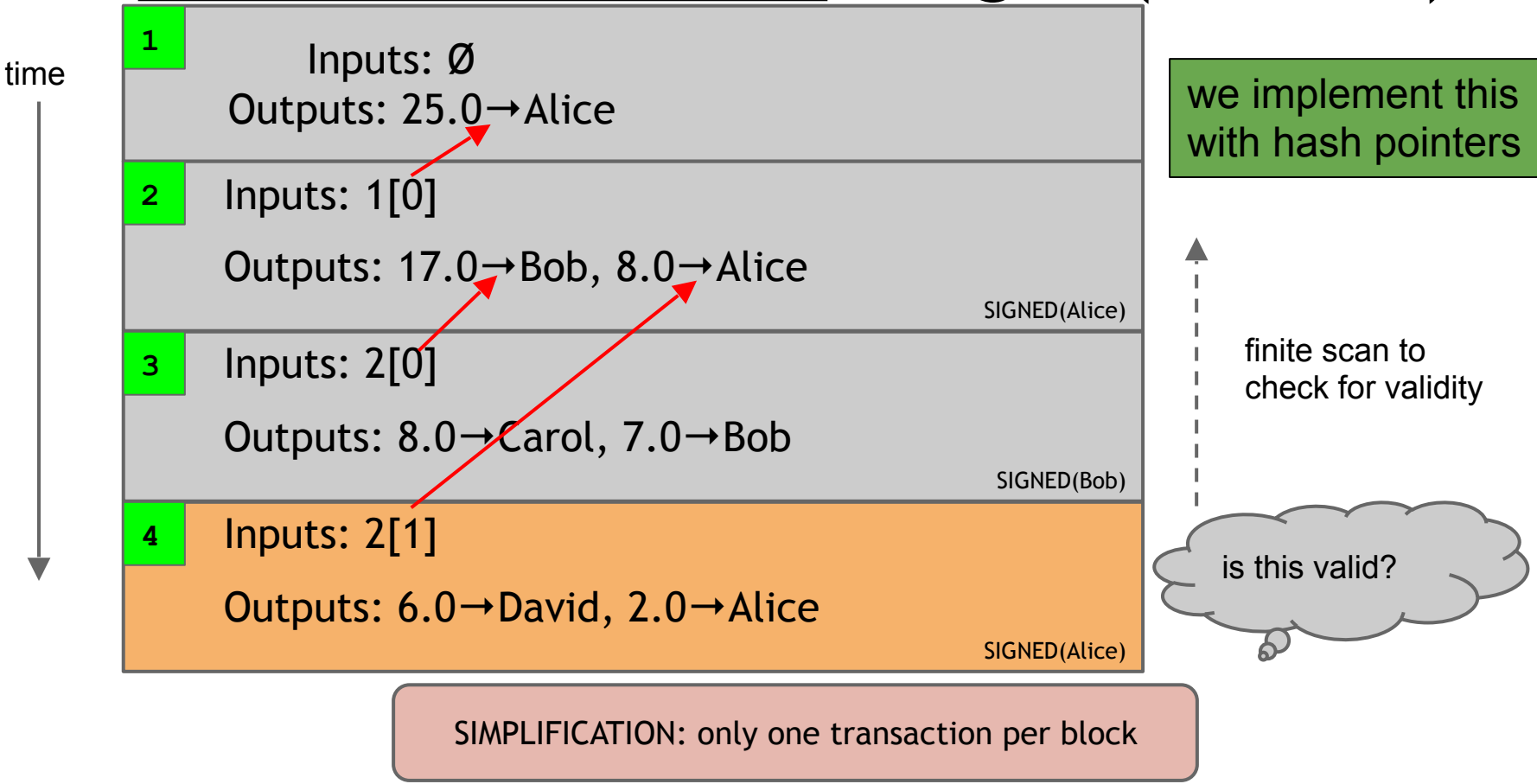
A transaction-based ledger (Bitcoin)



A transaction-based ledger (Bitcoin)



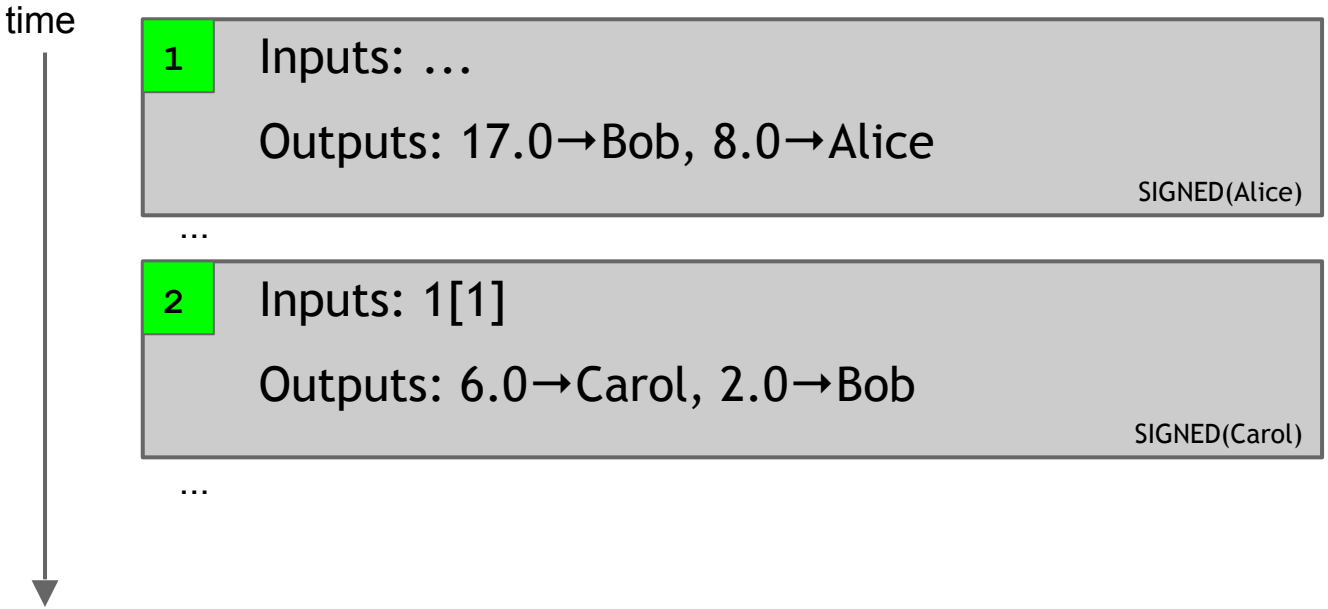
A transaction-based ledger (Bitcoin)



Referencing Transactions

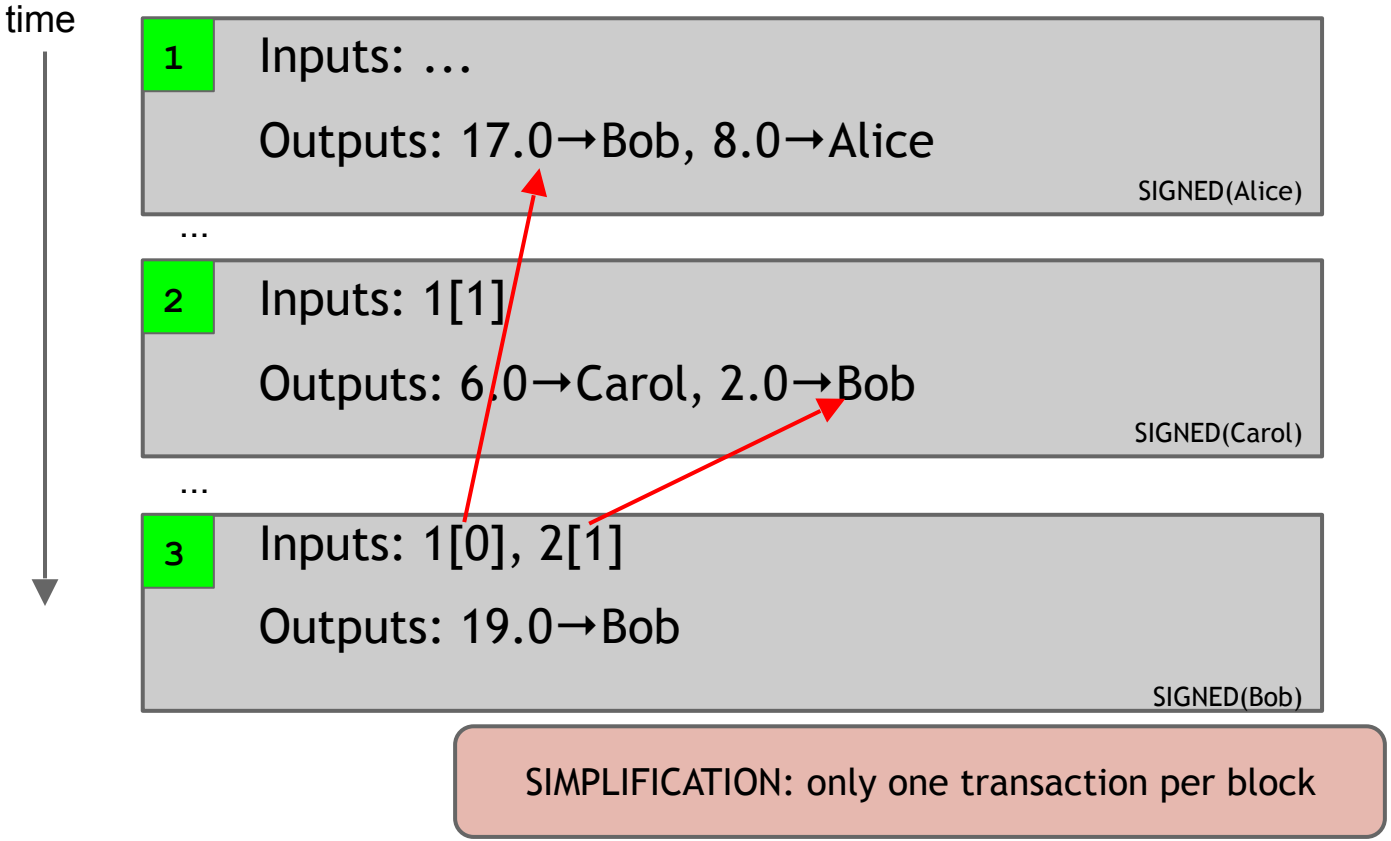
- Hash pointers for blocks
- Hash pointers for transactions
- Within a transaction, refer to a particular output via serial numbers

Merging value

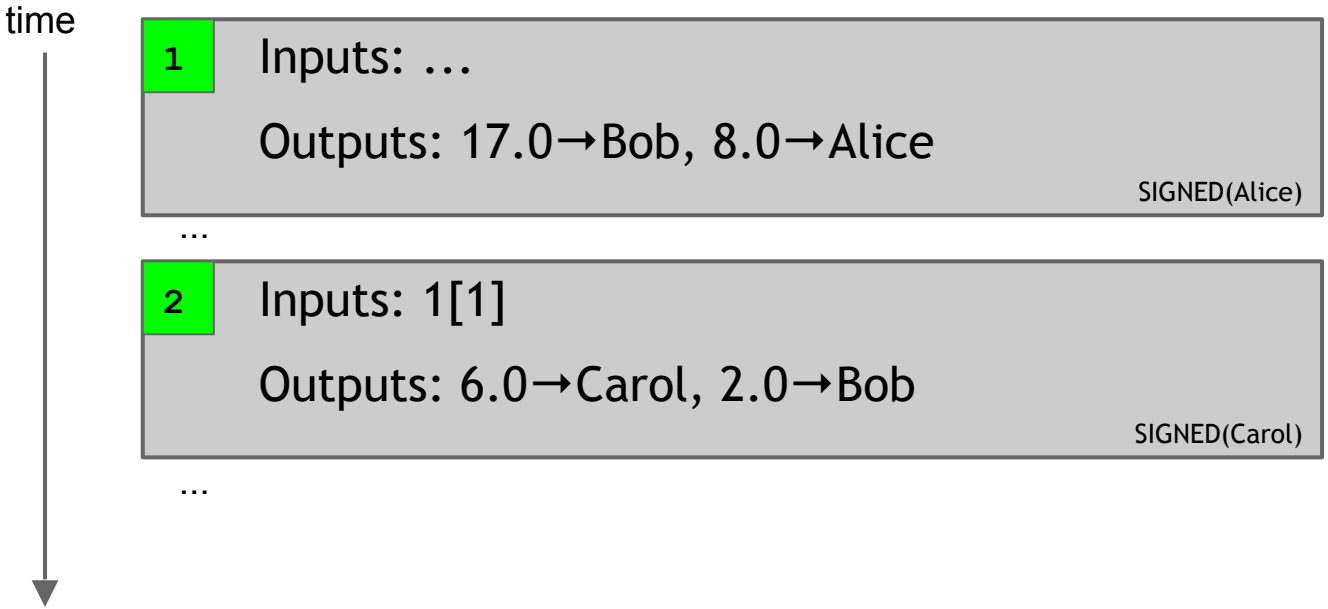


SIMPLIFICATION: only one transaction per block

Merging value

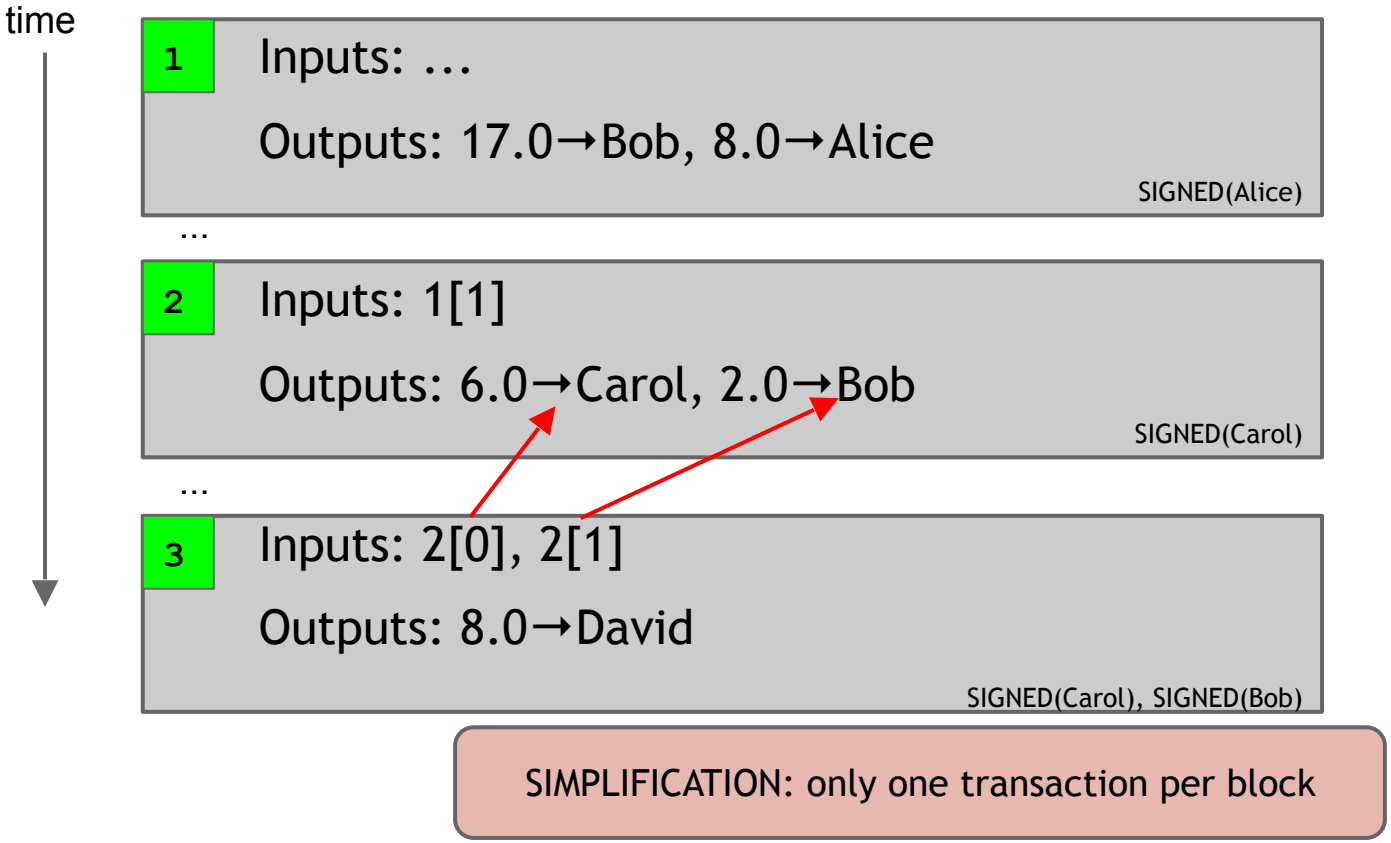


Joint payments

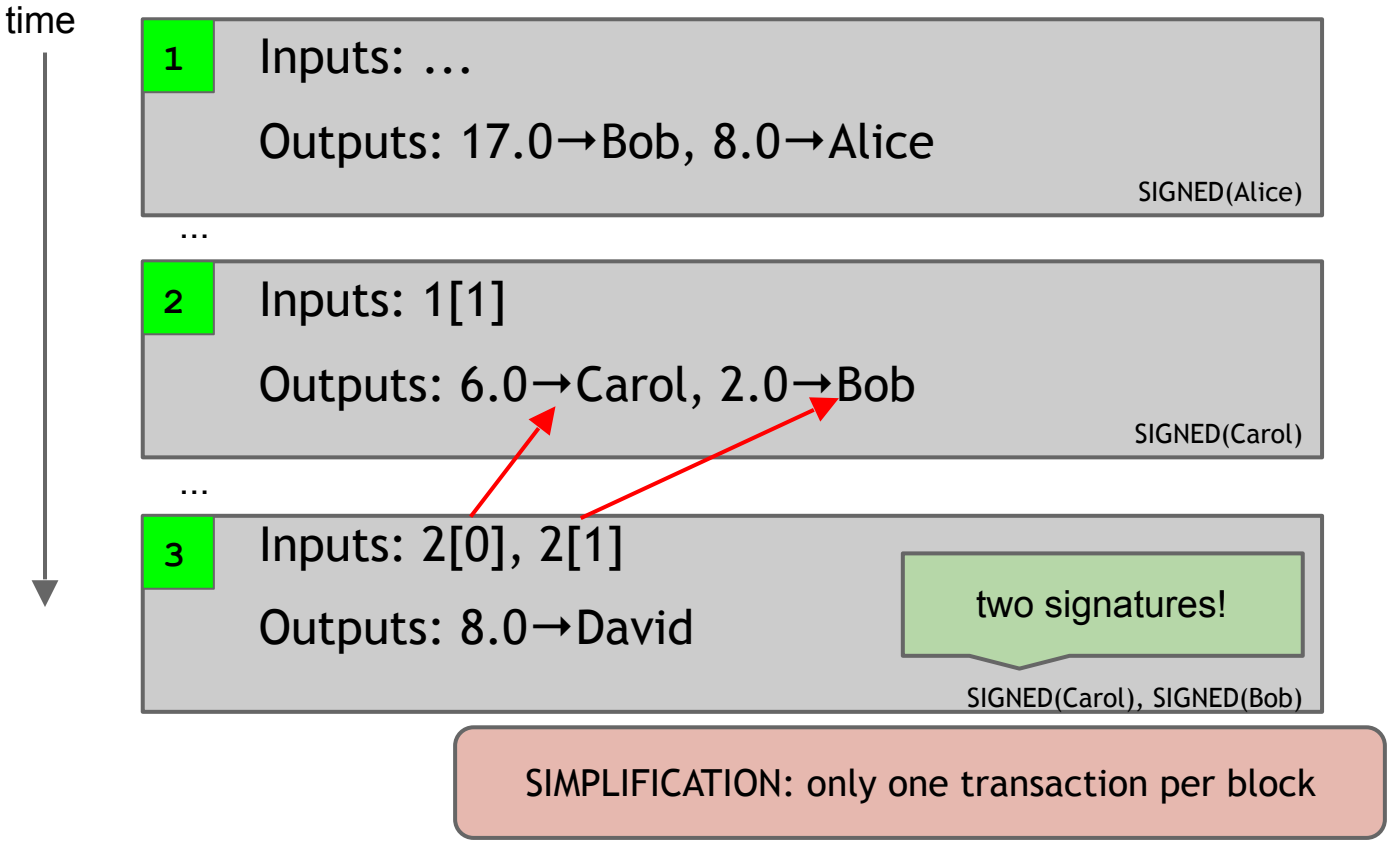


SIMPLIFICATION: only one transaction per block

Joint payments



Joint payments



The real deal: a Bitcoin transaction


```
{
    "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"
     }
    ]

}
```

The real deal: a Bitcoin transaction

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"
     }
    ]
}
```

The real deal: transaction metadata


```
{
    "hash":"5a42590...b8b6b",
      "ver":1,
      "vin_sz":2,
      "vout_sz":1,
      "lock_time":0,
      "size":404,

...
}
```

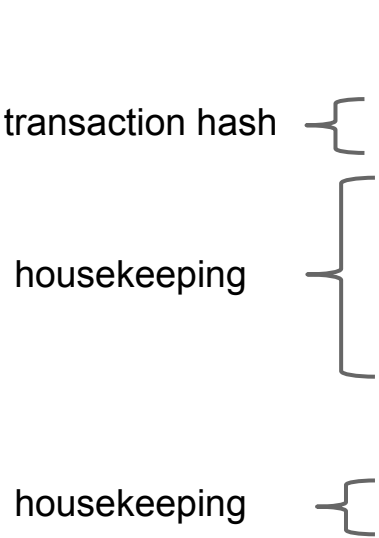
The real deal: transaction metadata

housekeeping

housekeeping

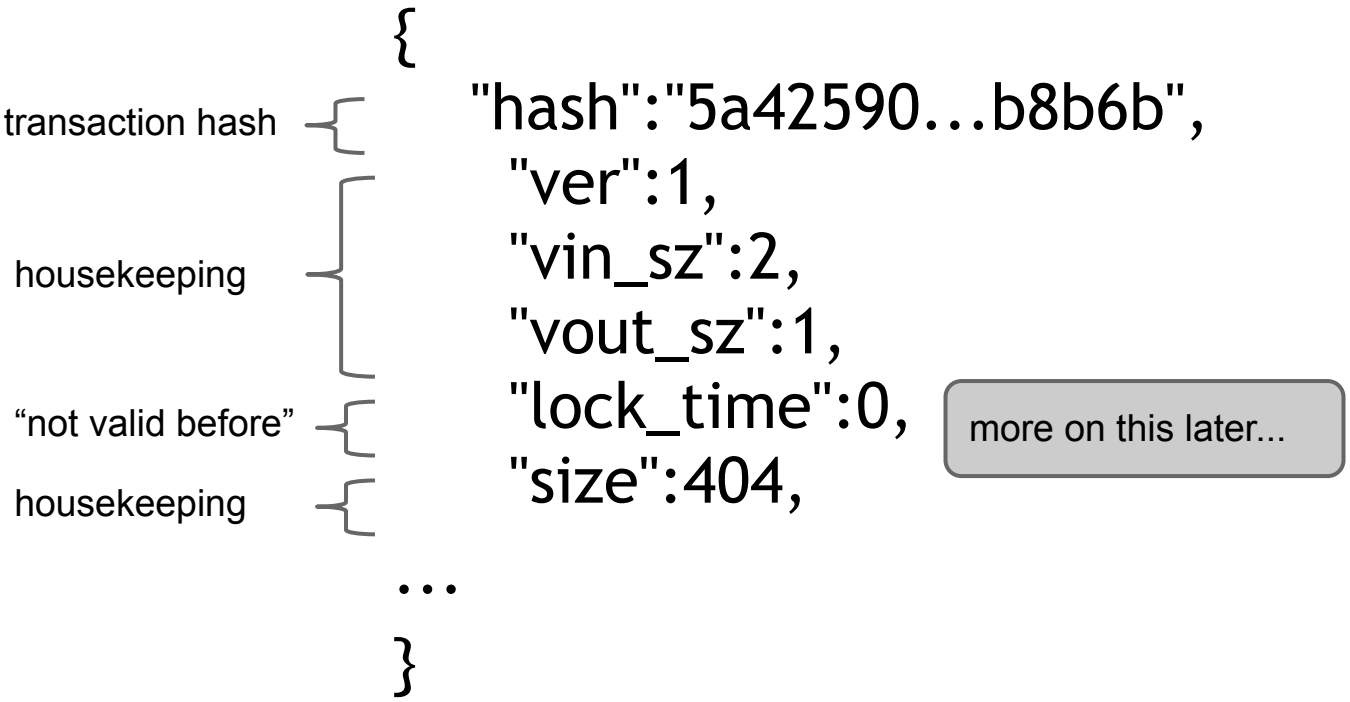

```
{
    "hash":"5a42590...b8b6b",
      "ver":1,
      "vin_sz":2,
      "vout_sz":1,
      "lock_time":0,
      "size":404,
...
}
```

The real deal: transaction metadata




```
{
    "hash":"5a42590...b8b6b",
      "ver":1,
      "vin_sz":2,
      "vout_sz":1,
      "lock_time":0,
      "size":404,
...
}
```

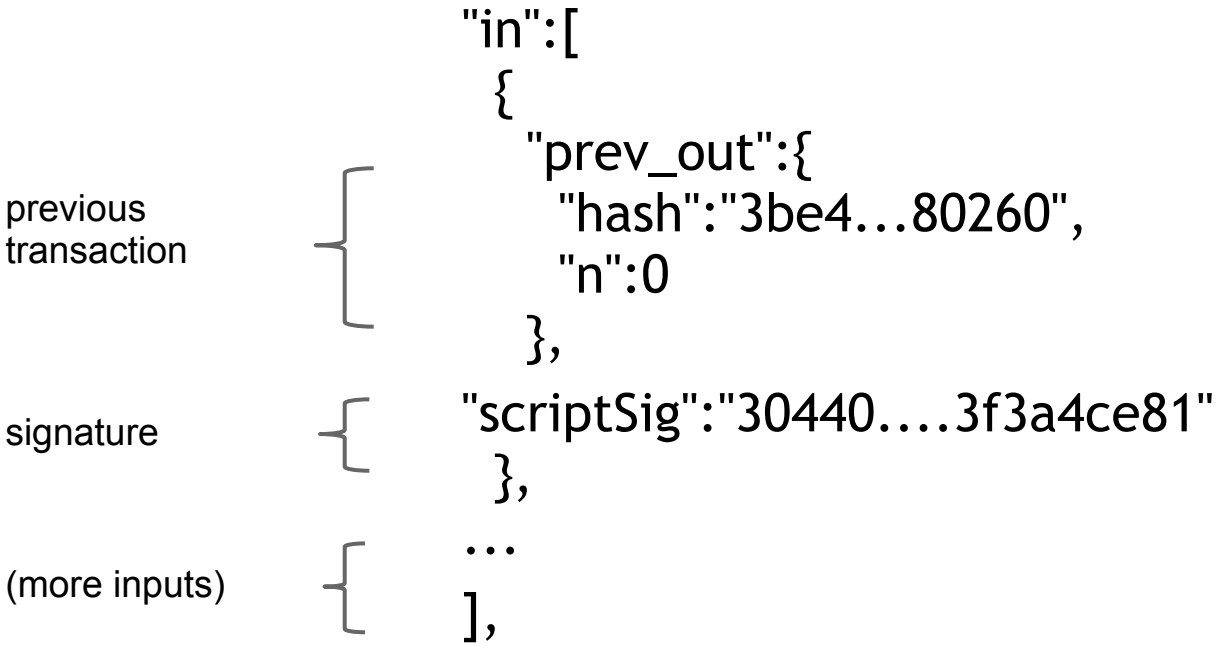
The real deal: transaction metadata



The real deal: transaction inputs

```
"in":[
  {
    "prev_out":{
      "hash":"3be4...80260",
      "n":0
    },
"scriptSig":"30440....3f3a4ce81"
  },
...
],
```

The real deal: transaction inputs



The real deal: transaction outputs


```
"out":[
      {
        "value":"10.12287097",
        "scriptPubKey":"OP_DUP OP_HASH160 69e...3d42e
OP_EQUALVERIFY OP_CHECKSIG"
      },
            ...
    ]
```

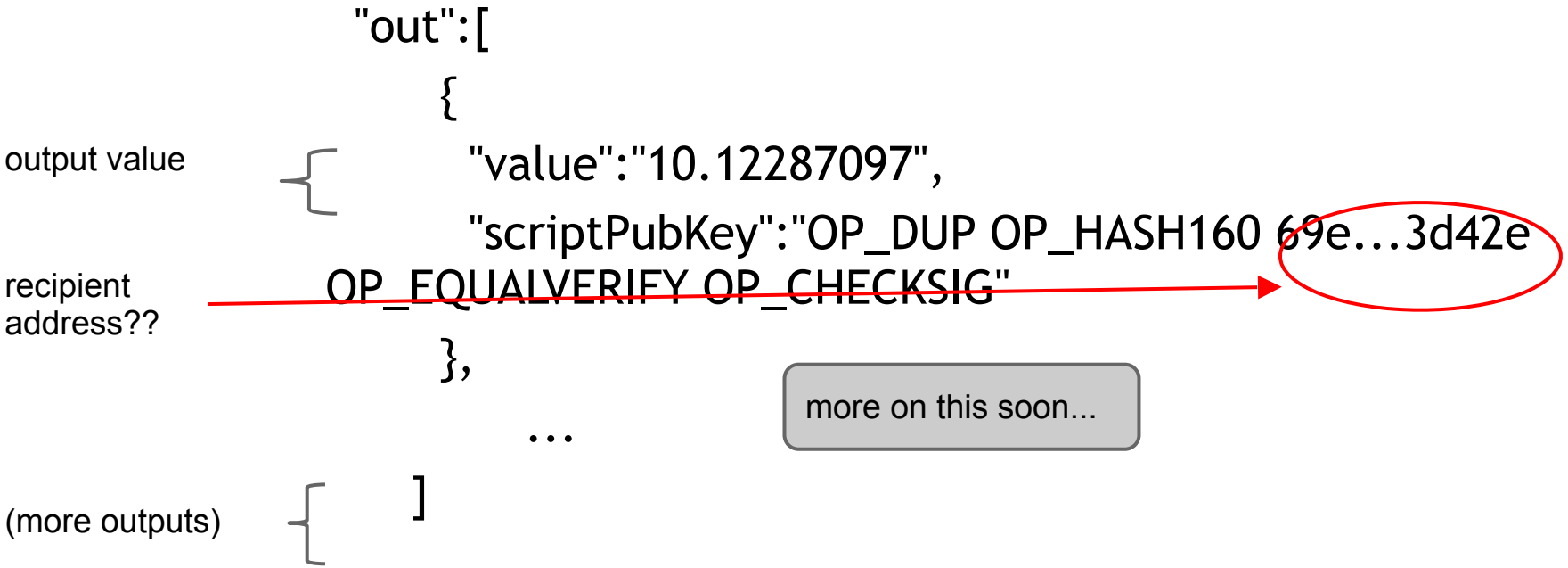
The real deal: transaction outputs

output value

(more outputs)

```
"out":[
    {
      "value":"10.12287097",
      "scriptPubKey":"OP_DUP OP_HASH160 69e...3d42e
OP_EQUALVERIFY OP_CHECKSIG"
    },
        ...
  ]
```

The real deal: transaction outputs



Bitcoin scripts

Output “addresses” are really *scripts*

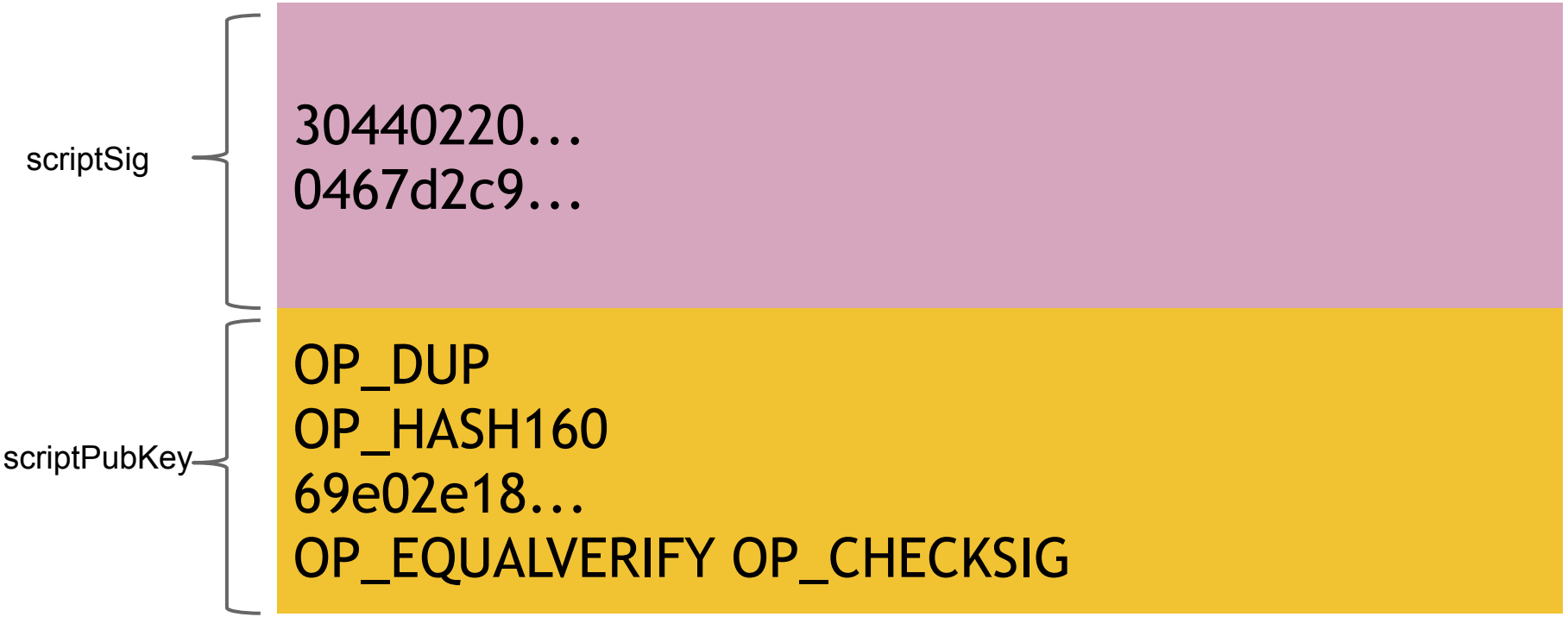
```
OP_DUP
OP_HASH160
69e02e18...
OP_EQUALVERIFY OP_CHECKSIG
```

Input “addresses” are *also* scripts

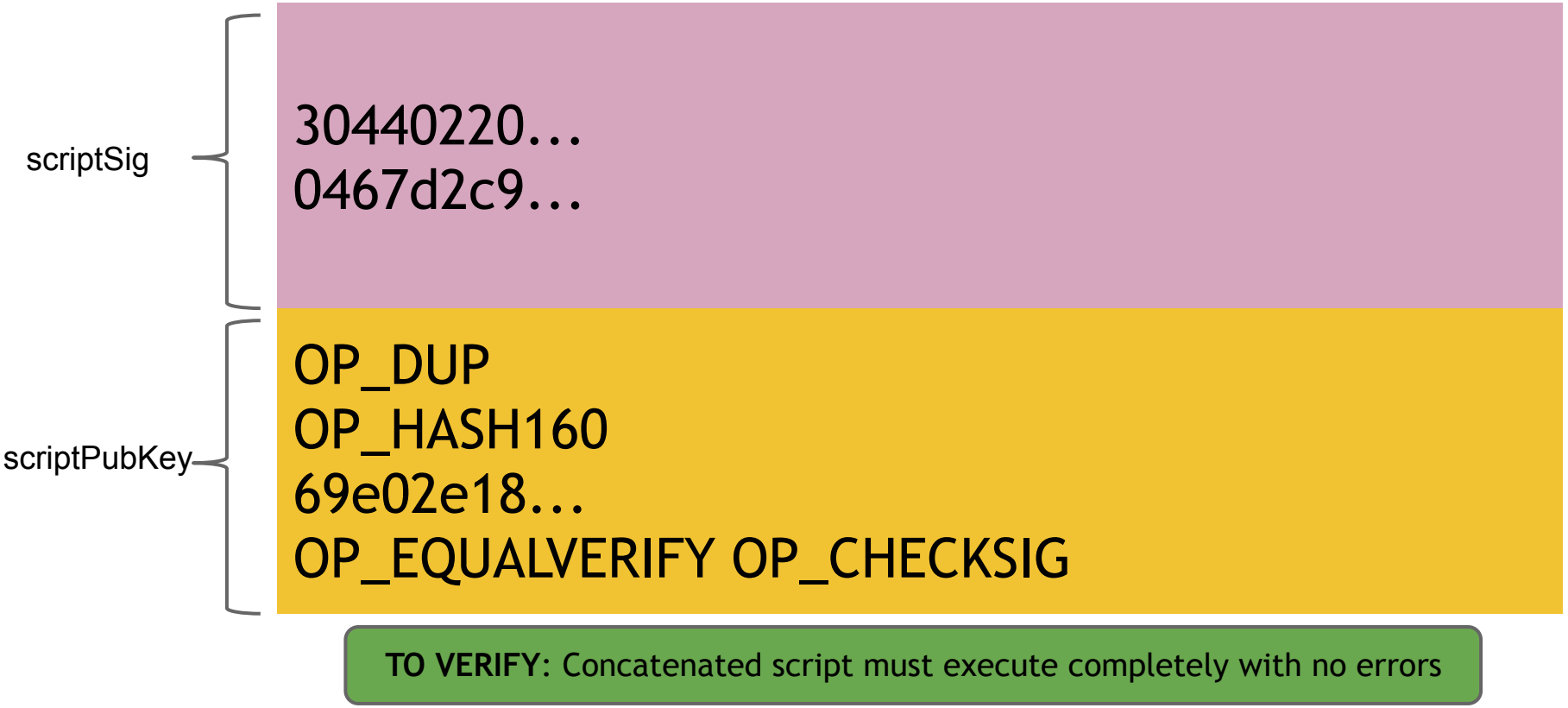
30440220...
0467d2c9...

OP_DUP
OP_HASH160
69e02e18...
OP_EQUALVERIFY OP_CHECKSIG

Input “addresses” are *also* scripts



Input “addresses” are *also* scripts



Bitcoin scripting language (“Script”)

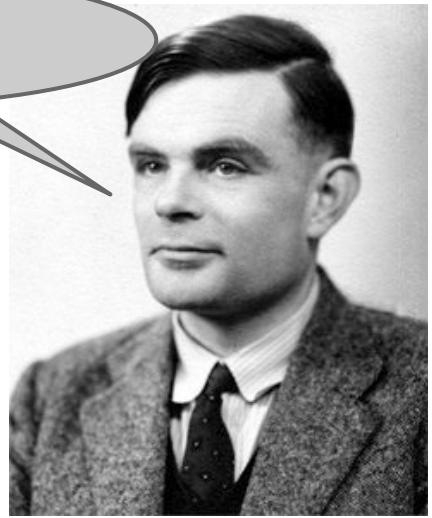
Design goals

- Built for Bitcoin
- Simple, compact
- Support for cryptography
- Stack-based
- Limits on time/memory
- No looping

Bitcoin scripting language (“Script”)

Design goals

- Built for Bitcoin
- Simple, compact
- Support for cryptography
- Stack-based
- Limits on time/memory
- No looping



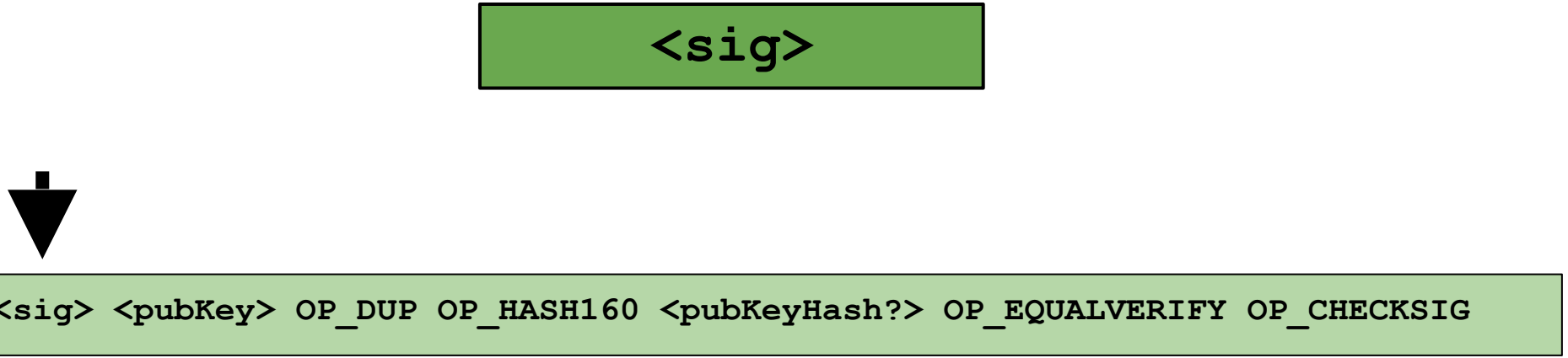
Bitcoin script execution example

```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

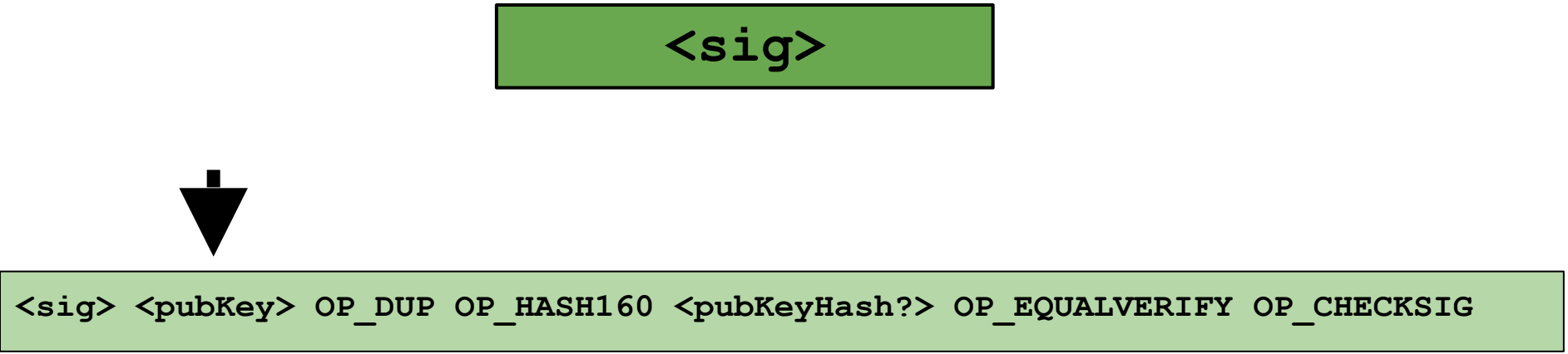
Bitcoin script execution example

```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

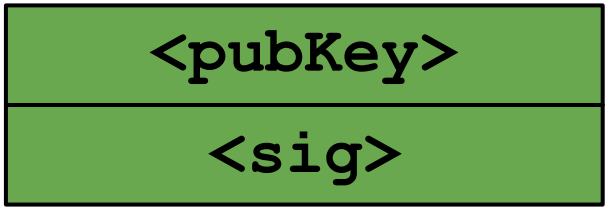
Bitcoin script execution example



Bitcoin script execution example

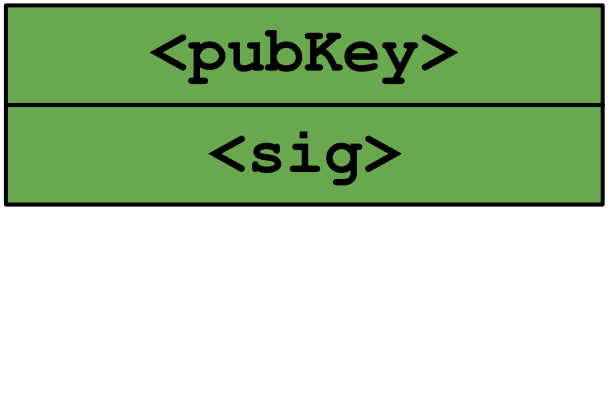


Bitcoin script execution example



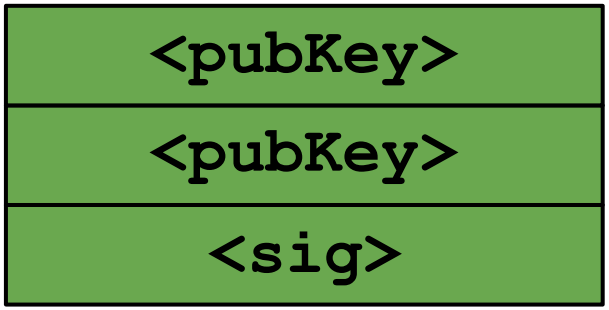
```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

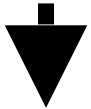
Bitcoin script execution example




```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

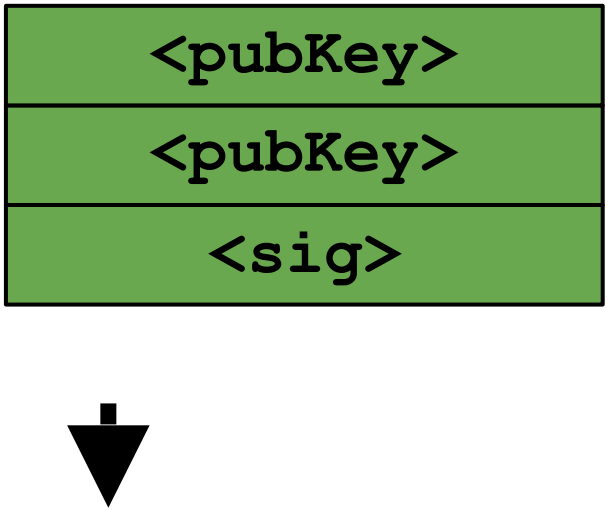
Bitcoin script execution example





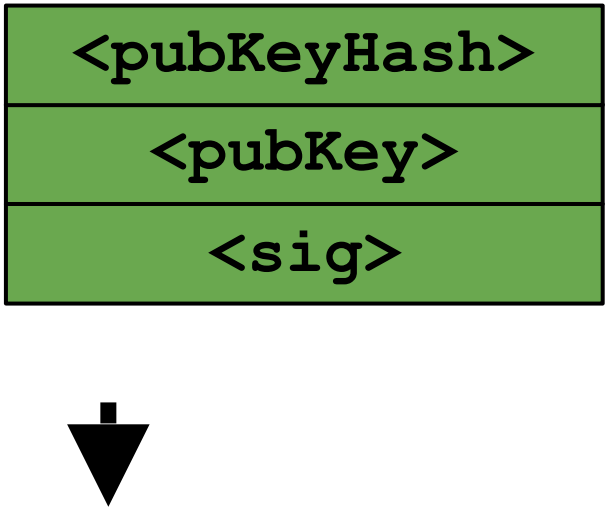
```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example



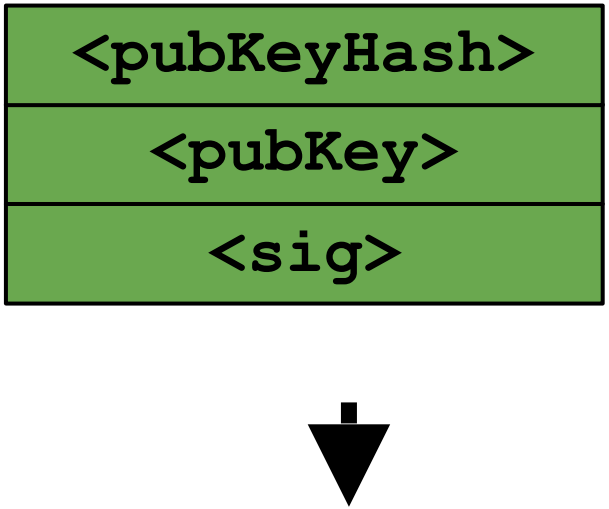

```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example



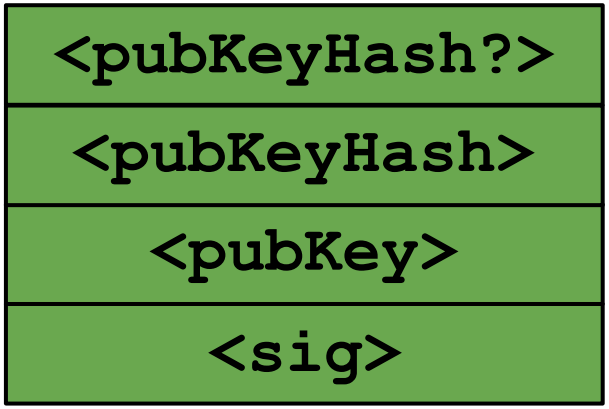

```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example



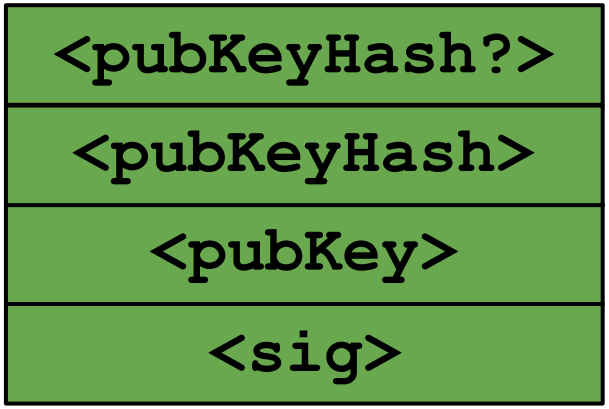

```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example



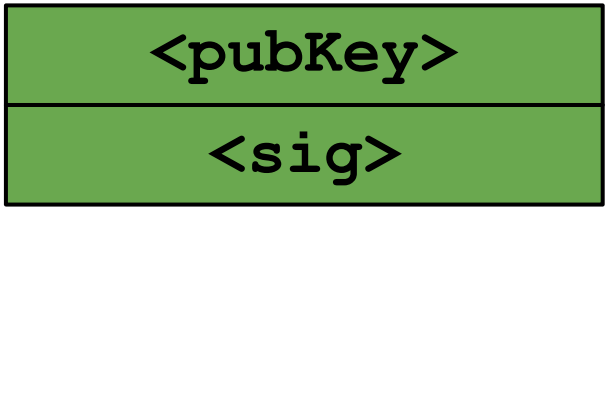

```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example



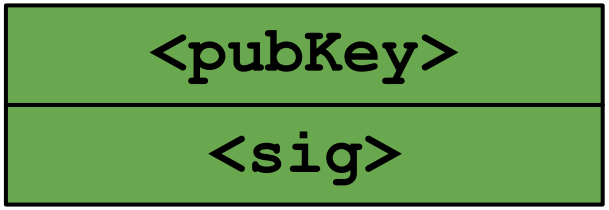

```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example




```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example



```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example

Bitcoin script execution example


```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script execution example

true

```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script instructions

256 opcodes total (15 disabled, 75 reserved)

- Arithmetic
- If/then
- Logic/data handling
- Crypto!
  - Hashes
  - Signature verification
  - Multi-signature verification

OP_CHECKMULTISIG

- Built-in support for joint signatures
- Specify *n* public keys
- Specify *t*
- Verification requires *t* signatures

OP_CHECKMULTISIG

- Built-in support for joint signatures
- Specify *n* public keys
- Specify *t*
- Verification requires *t* signatures

BUG ALERT: Extra data value popped from the stack and ignored

Bitcoin scripts in practice (as of 2014)

- Most nodes whitelist known scripts
- 99.9% are simple signature checks
- ~0.01% are MULTISIG
- ~0.01% are Pay-to-Script-Hash
- Remainder are errors, proof-of-burn

Bitcoin scripts in practice (as of 2014)

- Most nodes whitelist known scripts
- 99.9% are simple signature checks
- ~0.01% are MULTISIG
- ~0.01% are Pay-to-Script-Hash
- Remainder are errors, proof-of-burn

More on this soon

Proof-of-burn

OP_RETURN
<arbitrary data>

Proof-of-burn

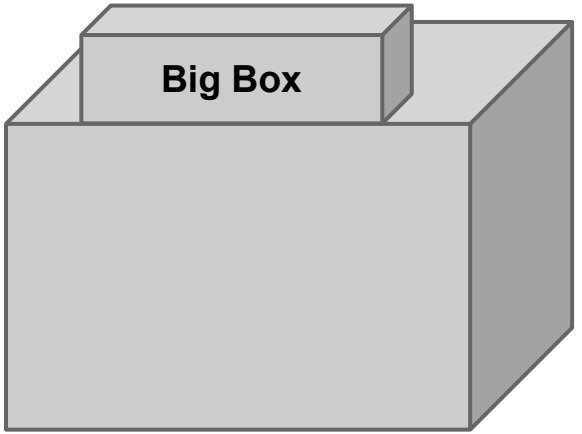
nothing’s going to redeem that ☹️

OP_RETURN
<arbitrary data>

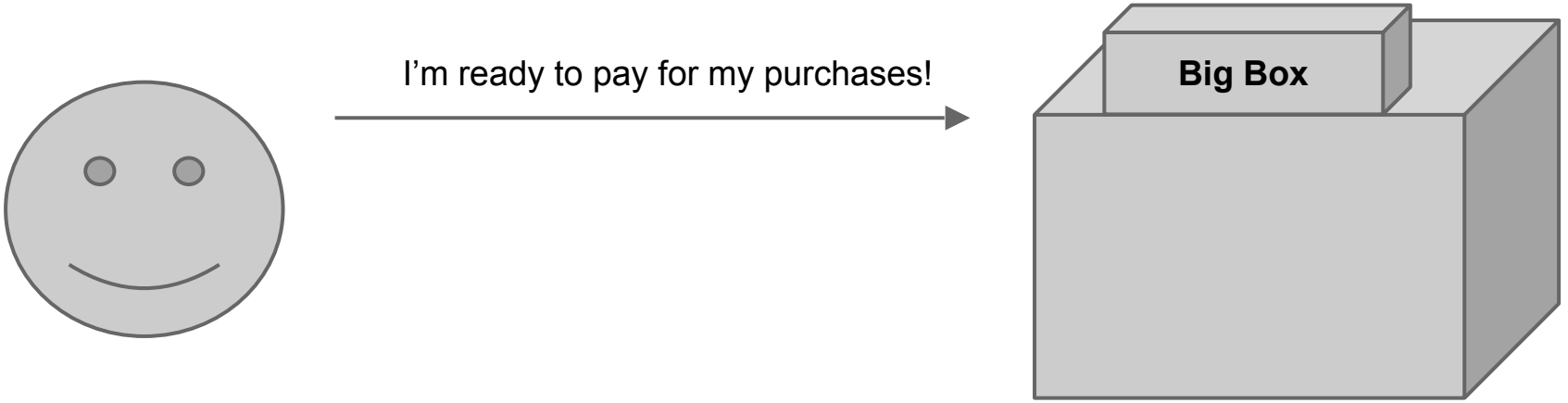
Proof-of-burn: Applications

- Can be used to publish arbitrary data on the blockchain (e.g., timestamping a document)
- Altcoins: Require people to burn bitcoin in order to get new altcoins

Should senders specify scripts?

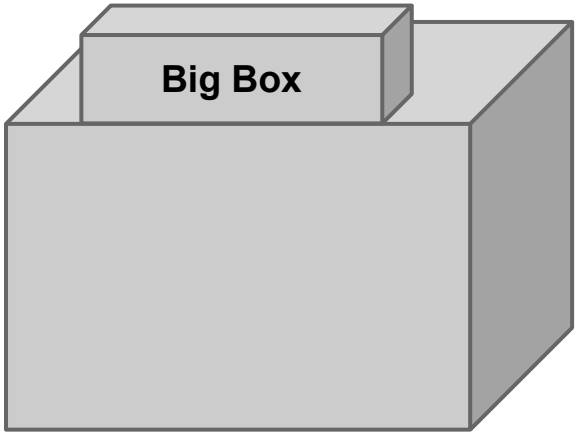


Should senders specify scripts?



Should senders specify scripts?

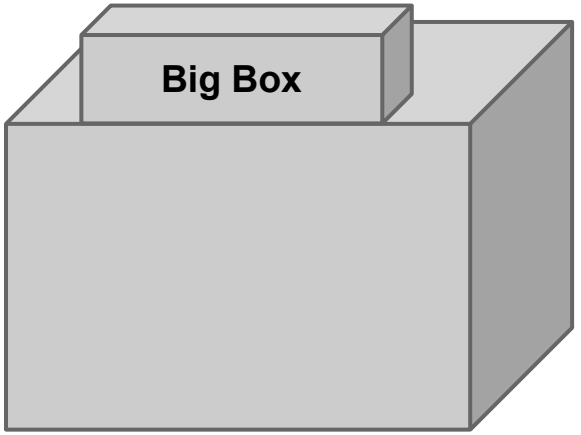
I’m ready to pay for my purchases!



Cool! Well we’re using MULTISIG now, so include a script requiring 2 of our 3 account managers to approve. Don’t get any of those details wrong. Thanks for shopping at Big Box!

Should senders specify scripts?

I’m ready to pay for my purchases!



Cool! Well we’re using MULTISIG now, so include a script requiring 2 of our 3 account managers to approve. Don’t get any of those details wrong. Thanks for shopping at Big Box!

Idea: use the hash of redemption script

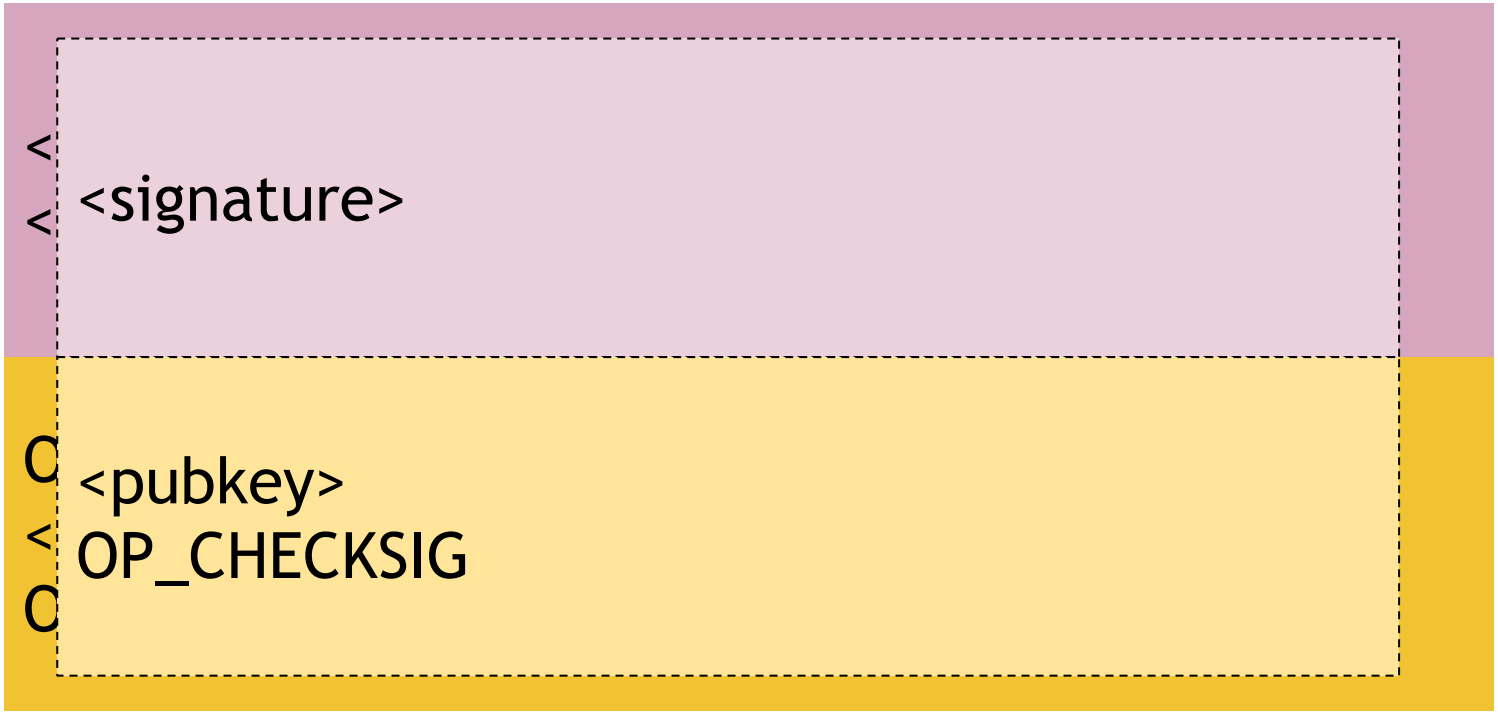
```
OP_HASH160
<hash of redemption script>
OP_EQUAL
```

Idea: use the hash of redemption script

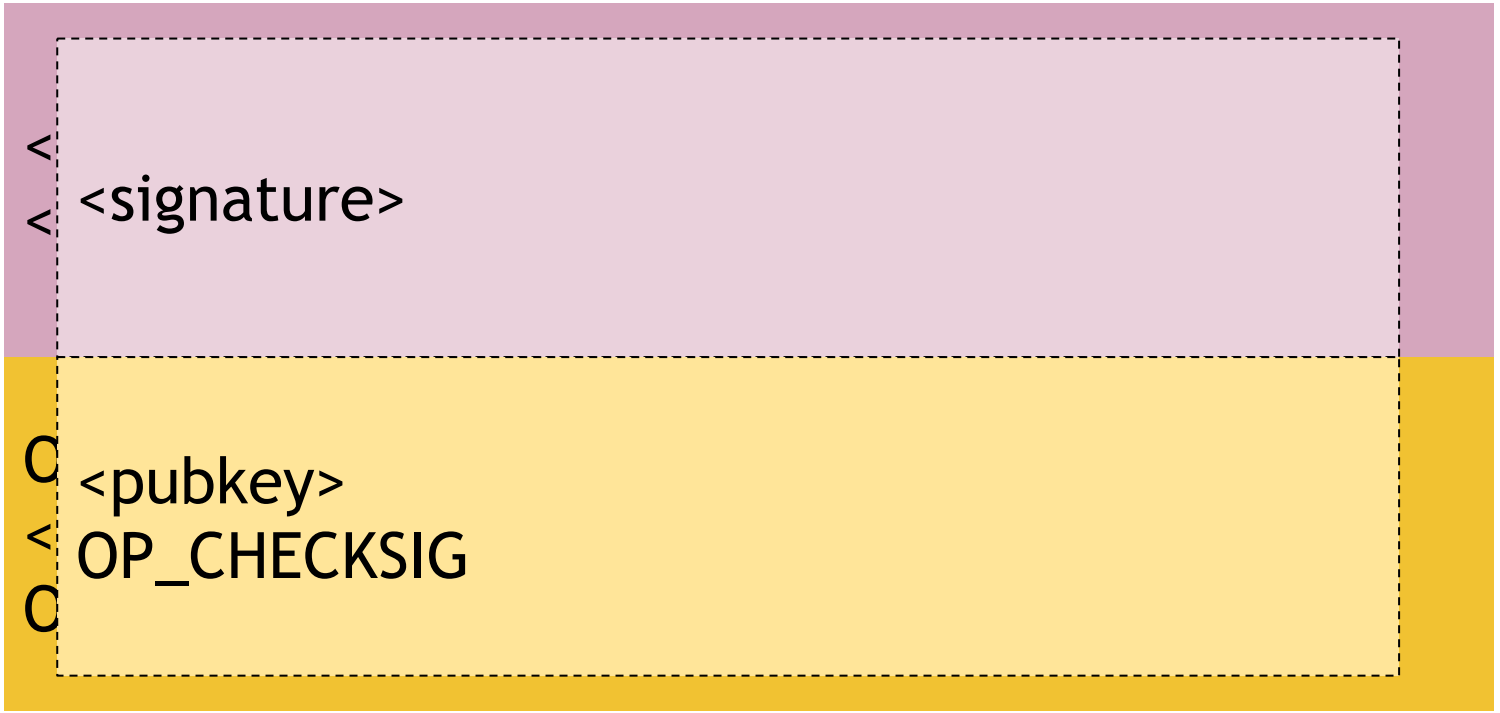
<signature>
<<pubkey> OP_CHECKSIG>

OP_HASH160
<hash of redemption script>
OP_EQUAL

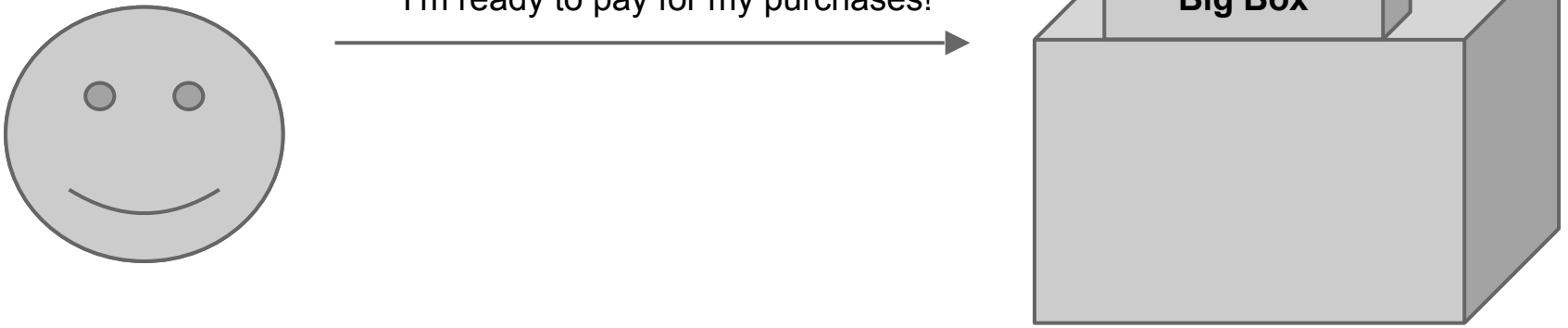
Idea: use the hash of redemption script



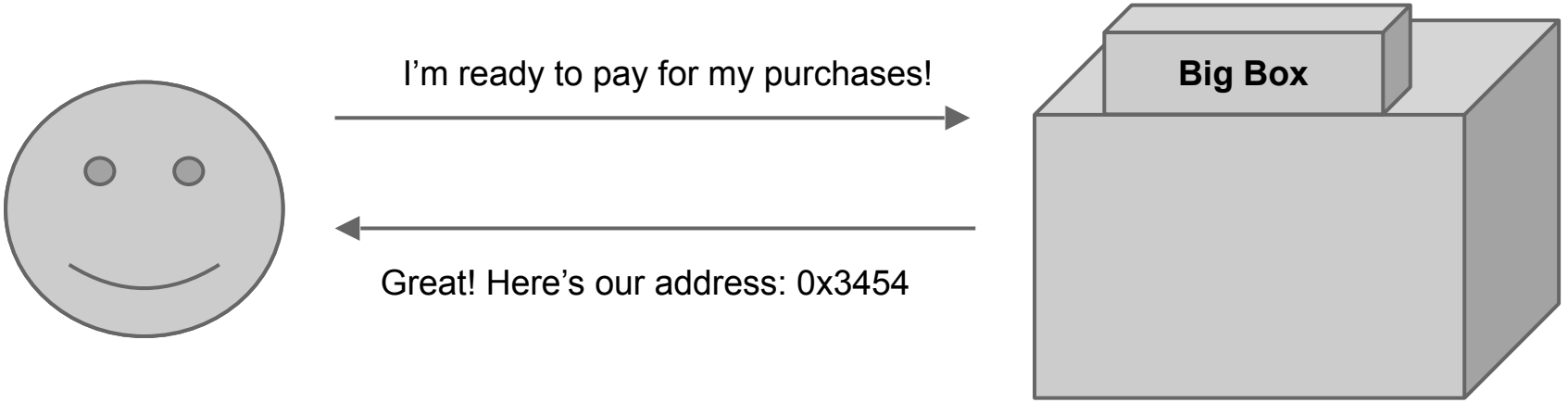
Idea: use the hash of redemption script



Pay to script hash



Pay to script hash



Applications of Bitcoin scripts

Example 1: “Fair” transactions

- Problem: Alice wants to buy a product from an online vendor Bob
- Alice doesn’t want to pay until after Bob ships
- Bob doesn’t want to ship until after Alice pays

Example 1: Fair transactions via Escrow

Alice

Bob

Example 1: Fair transactions via Escrow

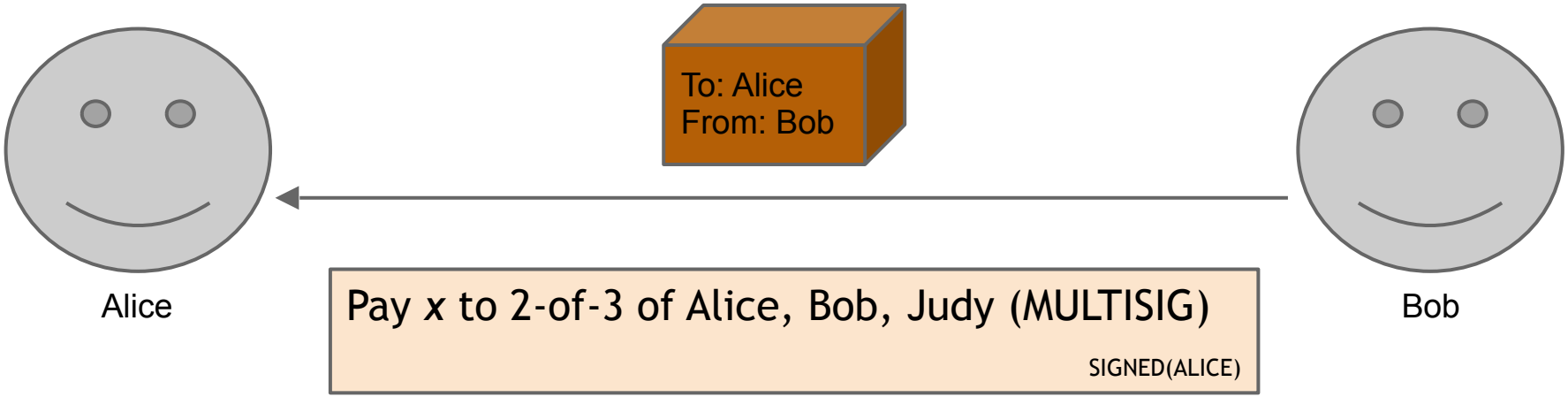
Alice

Pay *x* to 2-of-3 of Alice, Bob, Judy (MULTISIG)

SIGNED(ALICE)

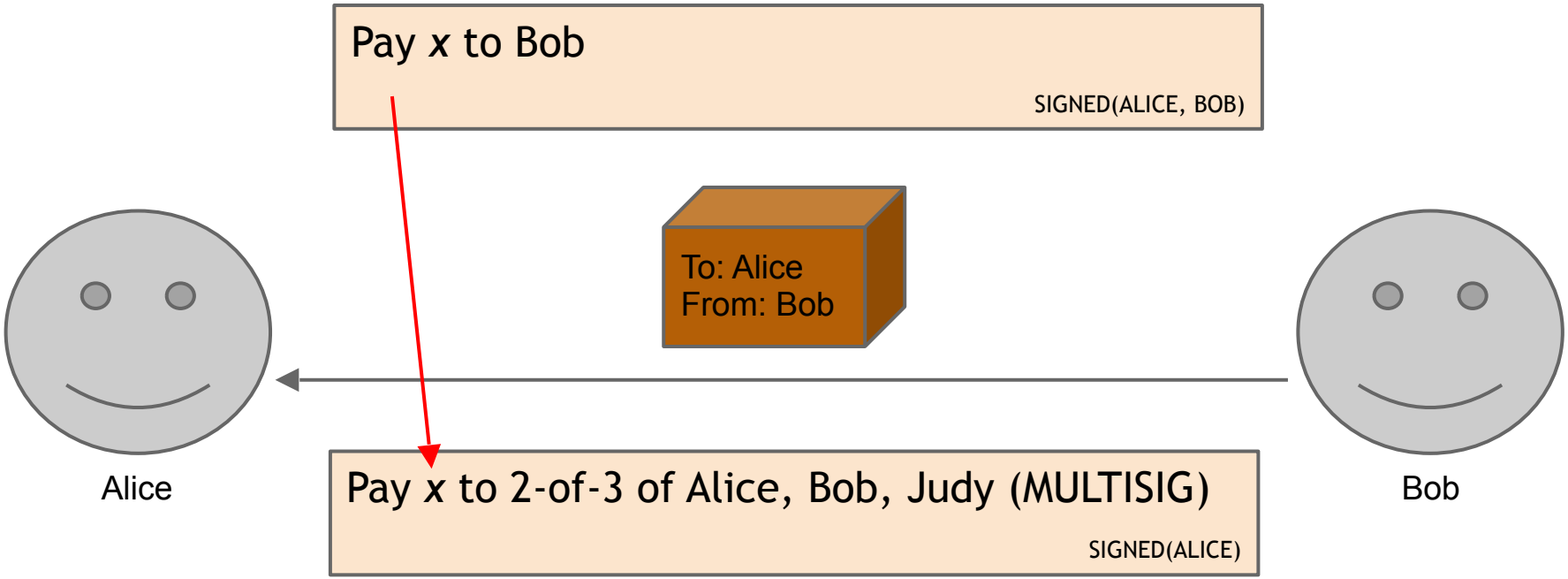
Bob

Example 1: Fair transactions via Escrow

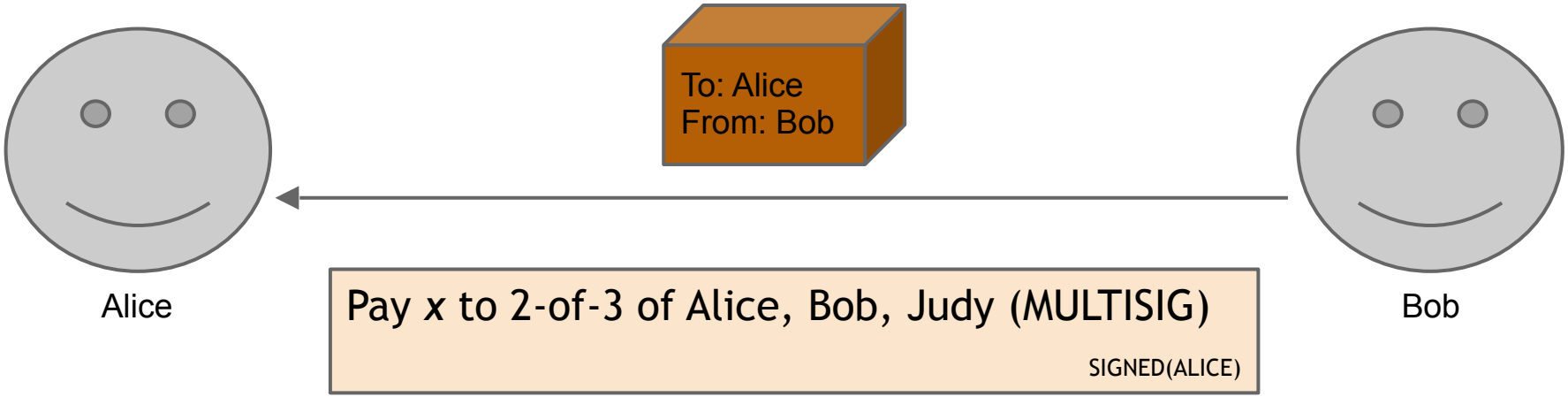


Example 1: Fair transactions via Escrow

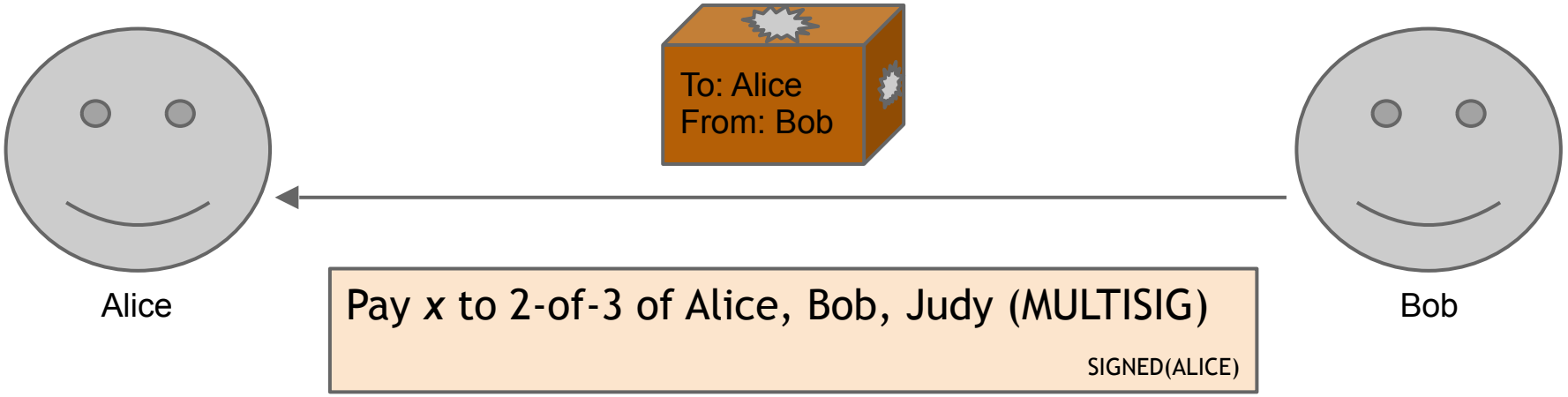
(normal case)



Example 1: Fair transactions via Escrow

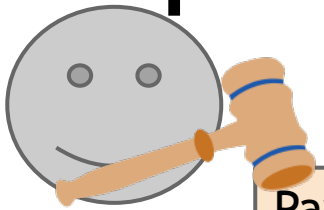


Example 1: Fair transactions via Escrow



Example 1: Fair transactions via Escrow

(disputed case)



Judy

Pay *x* to Alice

SIGNED(ALICE, JUDY)

Alice

Bob

Pay *x* to 2-of-3 of Alice, Bob, Judy (MULTISIG)

SIGNED(ALICE)

Example 2: Micro-payments

- Pay-as-you-go WIFI: Alice wants to pay WIFI provider (Bob) for each minute of WIFI service. But she doesn't want to incur a transaction fee for every minute
- Similarly, pay-as-you-go online subscriptions
- Ad-free websites

Example 2: Micro-payments with Bitcoin

Example 2: Micro-payments with Bitcoin

- Main Idea: Instead of doing several transactions, do a single transaction for total payment (and thus incur only a single transaction fee)
- *How to implement it?*

Example 2: Micro-payments with Bitcoin

Alice

Bob

Example 2: Micro-payments with Bitcoin

Alice

Input: *y*; Pay 100 to Bob/Alice (MULTISIG)

SIGNED(ALICE)

Bob

Example 2: Micro-payments with Bitcoin

Alice

Input: *x*; Pay 01 to Bob, 99 to Alice

SIGNED(ALICE)_____

Input: *y*; Pay 100 to Bob/Alice (MULTISIG)

SIGNED(ALICE)

Bob

Example 2: Micro-payments with Bitcoin

Alice

Input: *x*; Pay 02 to Bob, 98 to Alice
SIGNED(ALICE)_____

Input: *x*; Pay 01 to Bob, 99 to Alice
SIGNED(ALICE)_____

Input: *y*; Pay 100 to Bob/Alice (MULTISIG)
SIGNED(ALICE)

Bob

Example 2: Micro-payments with Bitcoin

Alice

Input: *x*; Pay 04 to Bob, 96 to Alice
SIGNED(ALICE)_____

Input: *x*; Pay 03 to Bob, 97 to Alice
SIGNED(ALICE)_____

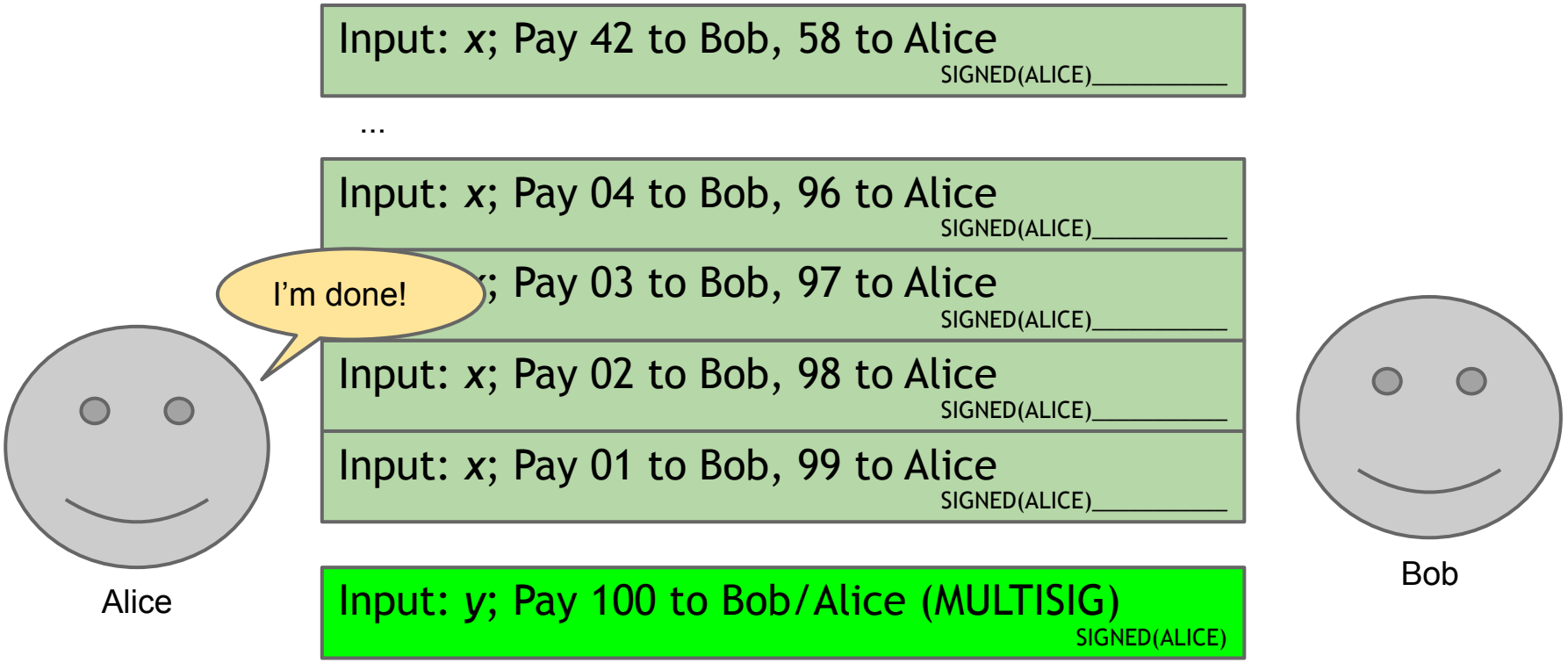
Input: *x*; Pay 02 to Bob, 98 to Alice
SIGNED(ALICE)_____

Input: *x*; Pay 01 to Bob, 99 to Alice
SIGNED(ALICE)_____

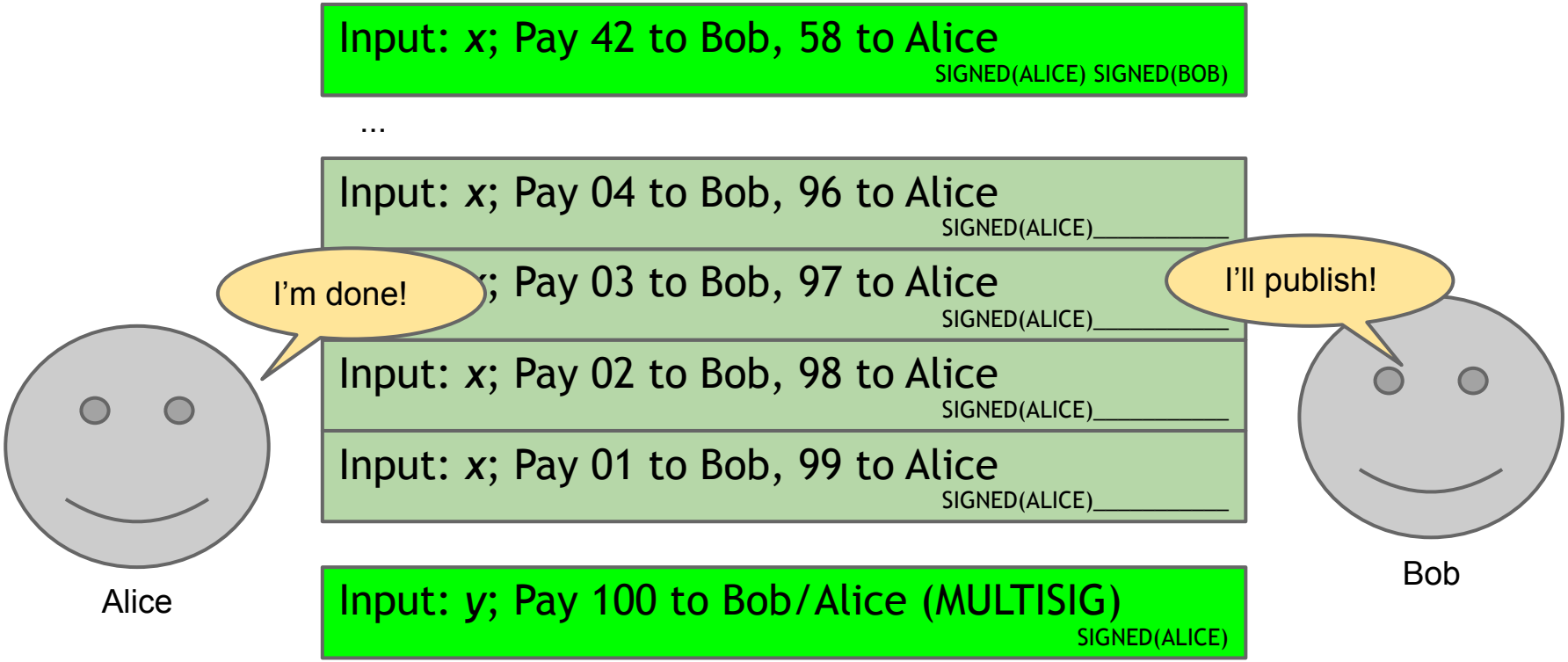
Input: *y*; Pay 100 to Bob/Alice (MULTISIG)
SIGNED(ALICE)

Bob

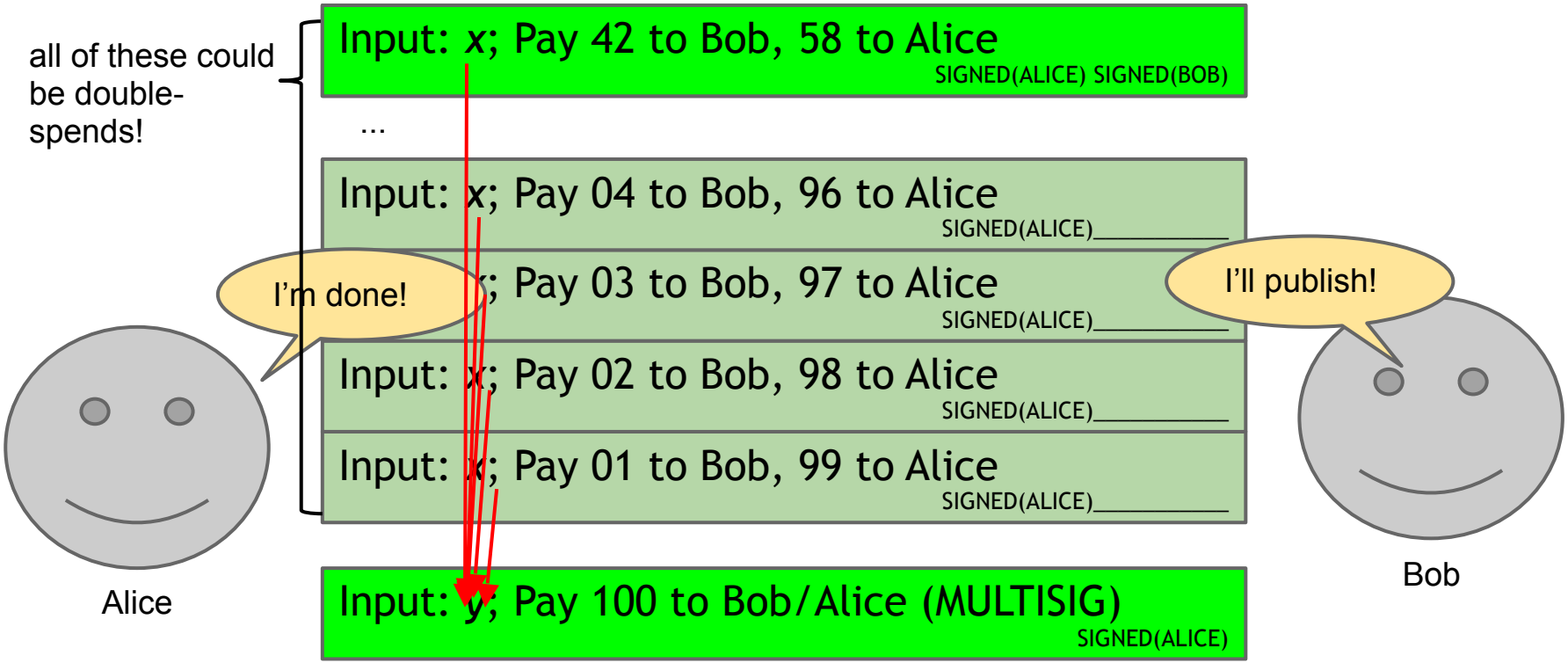
Example 2: Micro-payments with Bitcoin



Example 2: Micro-payments with Bitcoin



Example 2: Micro-payments with Bitcoin



Example 2: Micro-payments with Bitcoin

Input: *x*; Pay 42 to Bob, 58 to Alice

SIGNED(ALICE)_____

Alice

Bob

Input: *y*; Pay 100 to Bob/Alice (MULTISIG)

SIGNED(ALICE)

Example 2: Micro-payments with Bitcoin

What if Bob never signs??

Input: *x*; Pay 42 to Bob, 58 to Alice

SIGNED(ALICE)_____

Alice

Input: *y*; Pay 100 to Bob/Alice (MULTISIG)

SIGNED(ALICE)

Bob

Example 2: Micro-payments with Bitcoin

What if Bob never signs??

Input: *x*; Pay 42 to Bob, 58 to Alice
SIGNED(ALICE)_____

Alice demands a timed refund transaction before starting

Input: *x*; Pay 100 to Alice, LOCK until time *t*
SIGNED(ALICE) SIGNED(BOB)

Alice

Bob

Input: *y*; Pay 100 to Bob/Alice (MULTISIG)
SIGNED(ALICE)

lock_time


```
{
    "hash":"5a42590...b8b6b",
      "ver":1,
      "vin_sz":2,
      "vout_sz":1,
      "lock_time":315415,
      "size":404,

...
}
```

lock_time


```
{
    "hash":"5a42590...b8b6b",
      "ver":1,
      "vin_sz":2,
      "vout_sz":1,
      "lock_time":315415,
      "size":404,
...
}
```


Block index or real-world timestamp before which this transaction can’t be published

Micro-payments from Cryptocurrencies

More recent constructions, that achieve better properties

- Pass, shelat [CCS'16]
- Chiesa, Green, Liu, Miao, Miers, Mishra [EUROCRYPT'17]

More advanced scripts

- Fair multiplayer lotteries and fair multiparty computation [Andrychowichz-Dziembowski-Malinowski-Mazurek, S&P'14; Bentov-Kumaresan, CRYPTO'14]
- Hash pre-image challenges

More advanced scripts

- Fair multiplayer lotteries and fair multiparty computation [Andrychowichz-Dziembowski-Malinowski-Mazurek, S&P'14; Bentov-Kumaresan, CRYPTO'14]
- Hash pre-image challenges

"Smart contracts"

More advanced scripts

- Fair multiplayer lotteries and fair multiparty computation [Andrychowichz-Dziembowski-Malinowski-Mazurek, S&P'14; Bentov-Kumaresan, CRYPTO'14]
- Hash pre-image challenges

"Smart contracts"

Later: More powerful smart contracts with Ethereum (Turing-complete scripting language)

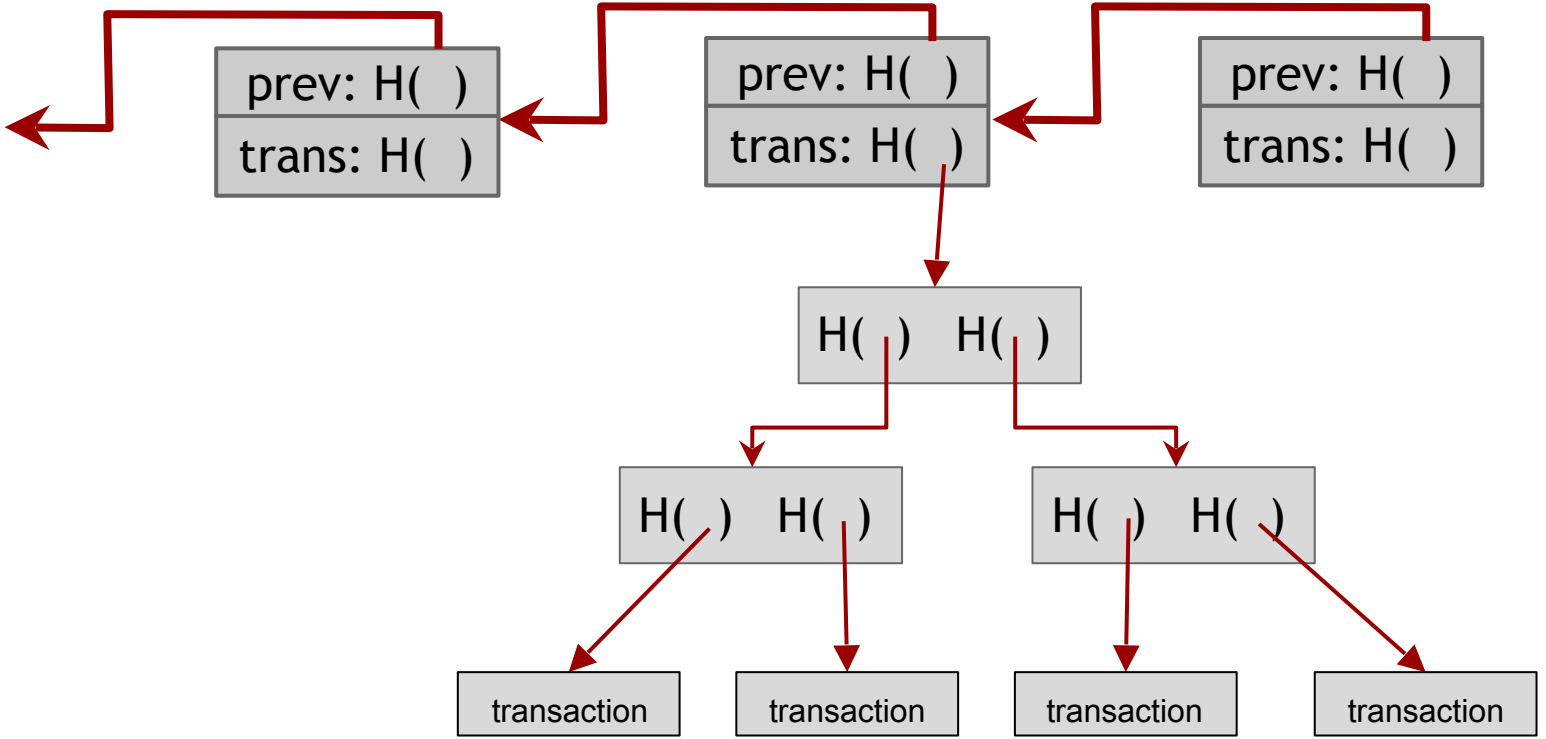
Bitcoin blocks

Bitcoin blocks

Why bundle transactions together?

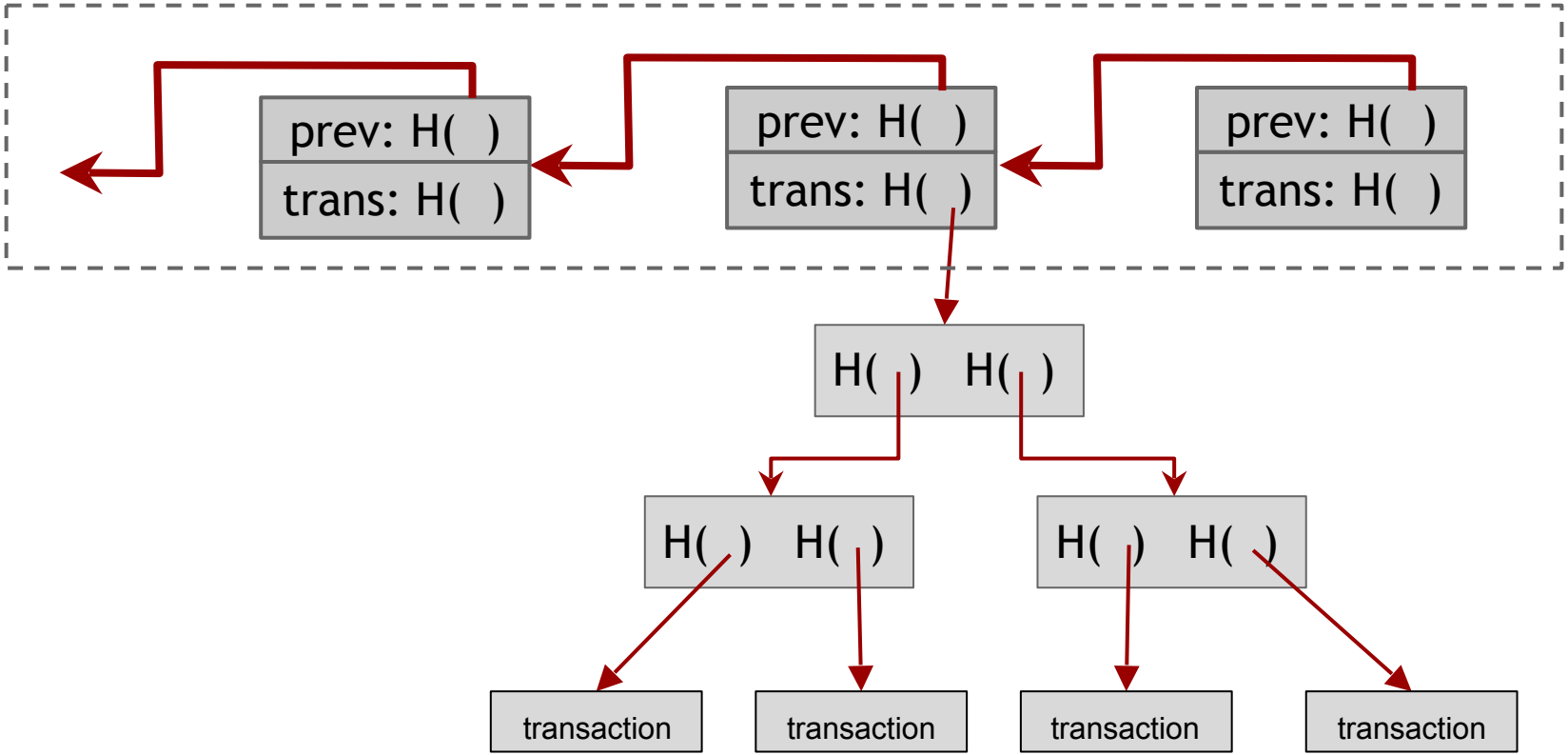
- Single unit of work for miners
- Limit length of hash-chain of blocks
    - Faster to verify history

Bitcoin block structure



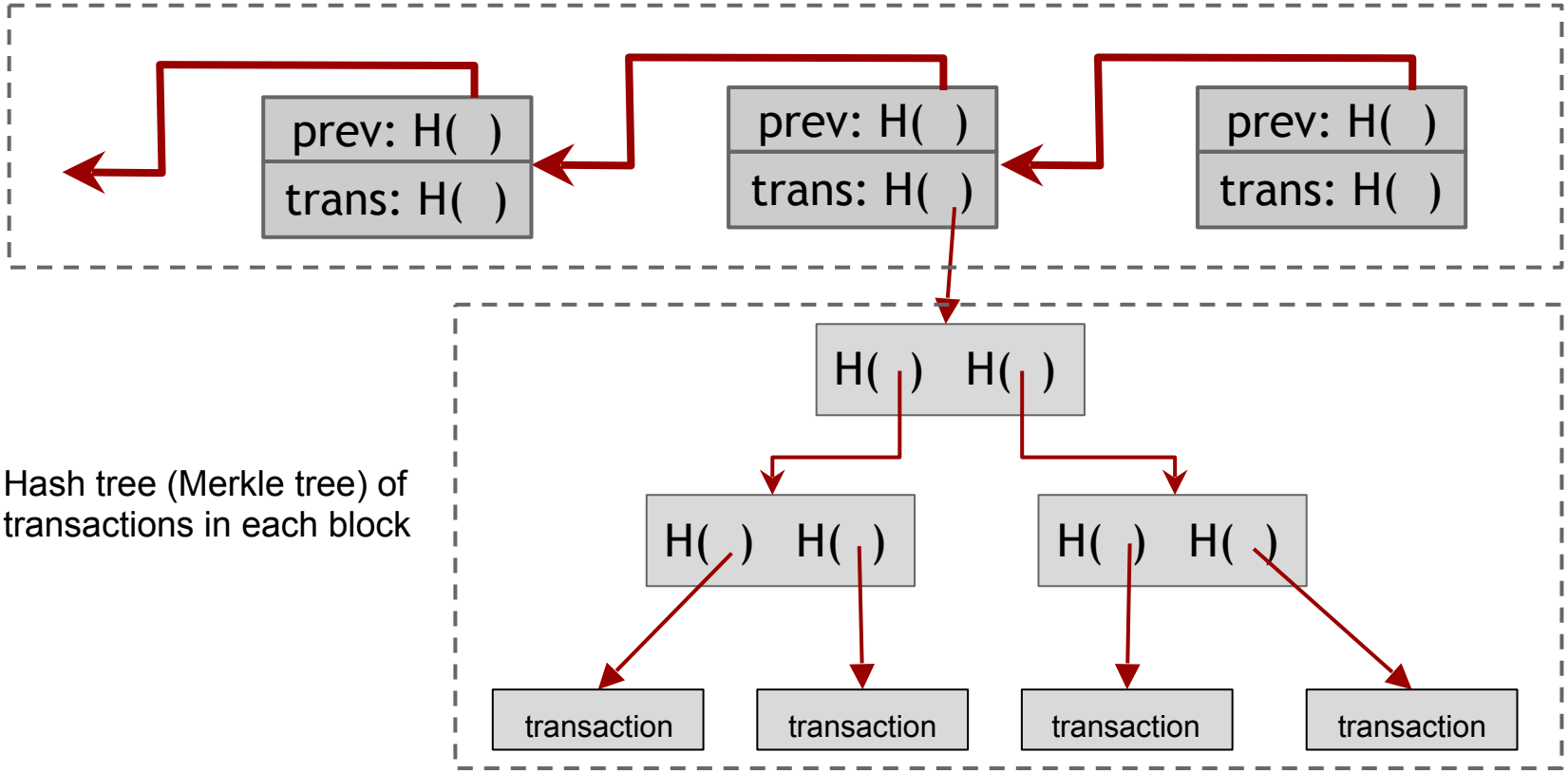
Bitcoin block structure

Hash chain of blocks



Bitcoin block structure

Hash chain of blocks



Hash tree (Merkle tree) of transactions in each block

The real deal: a Bitcoin block

block header

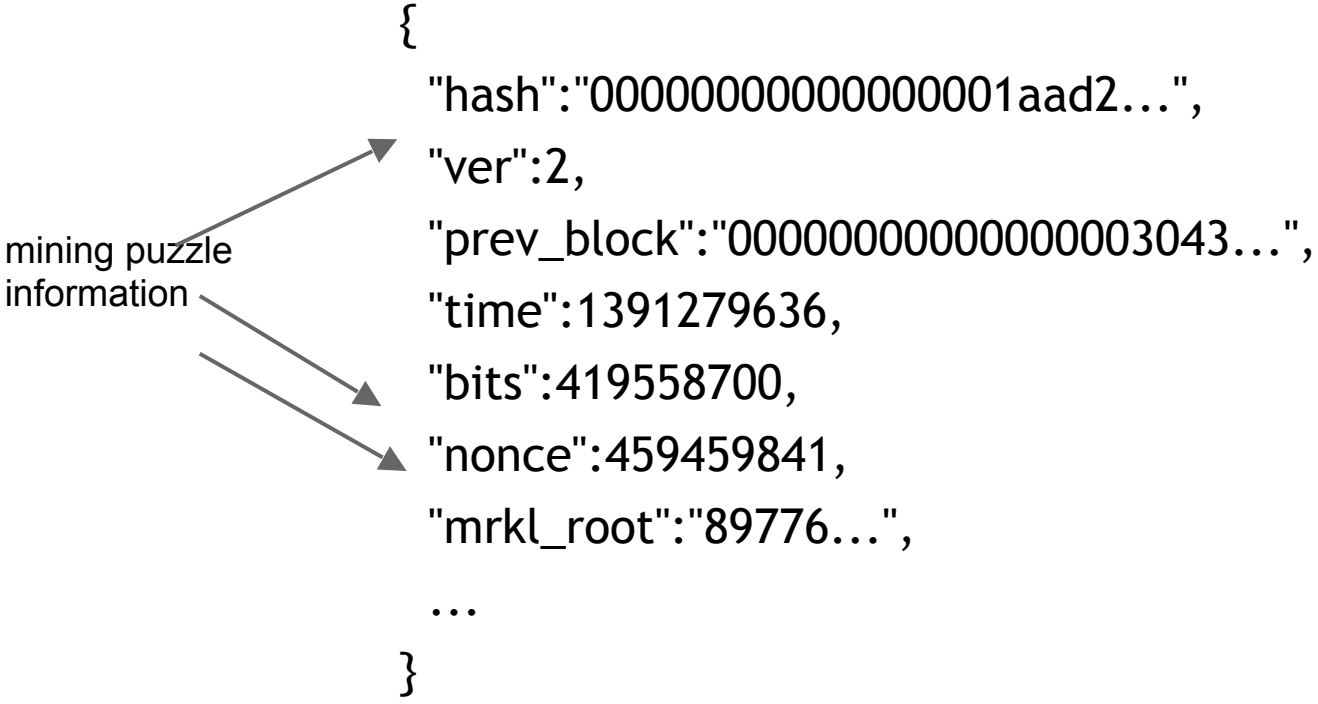
transaction data


```
{
  "hash":"00000000000000001aad2...",
  "ver":2,
  "prev_block":"00000000000000003043...",
  "time":1391279636,
  "bits":419558700,
  "nonce":459459841,
  "mrkl_root":"89776...",
  "n_tx":354,
  "size":181520,
  "tx":[
    ...
  ],
  "mrkl_tree":[
    "6bd5eb25...",
    ...
    "89776cdb..."
  ]
}
```

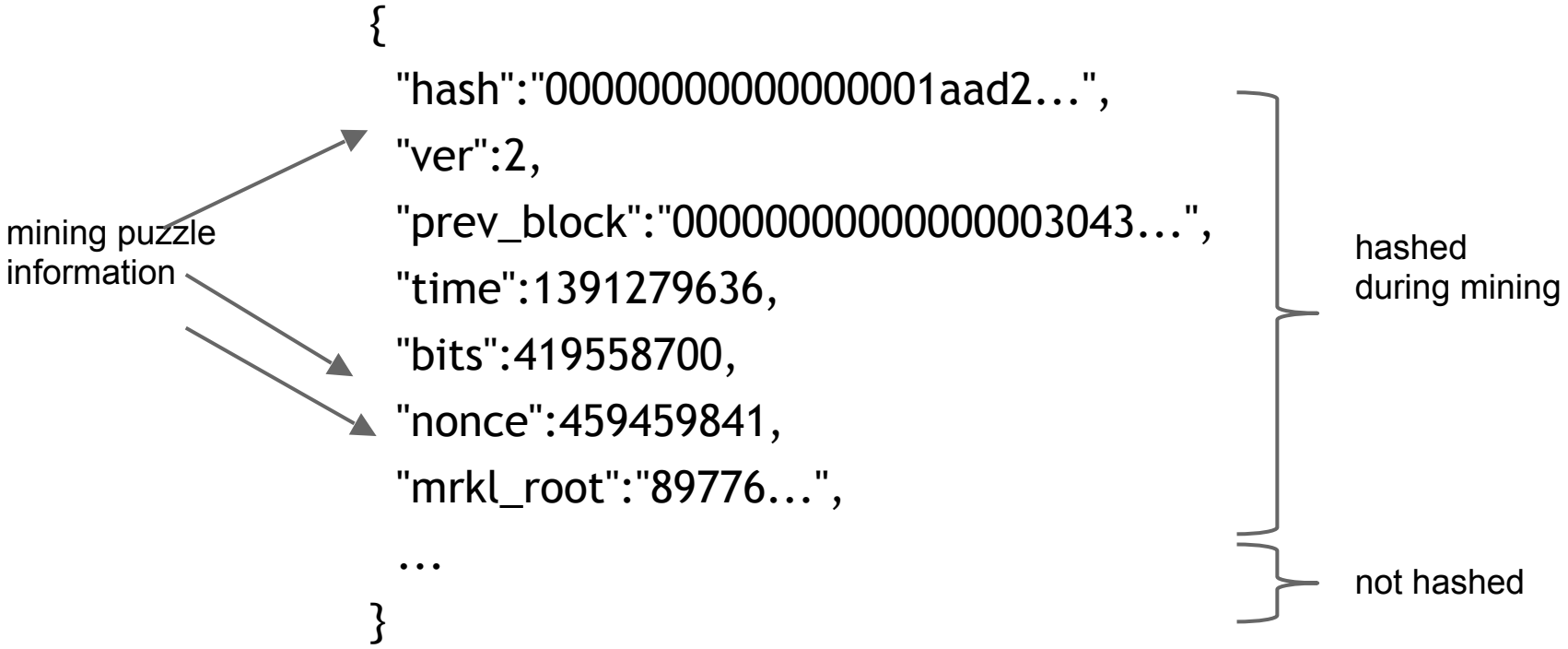
The real deal: a Bitcoin block header


```
{
  "hash":"0000000000000001aad2...",
  "ver":2,
  "prev_block":"0000000000000003043...",
  "time":1391279636,
  "bits":419558700,
  "nonce":459459841,
  "mrkl_root":"89776...",
  ...
}
```

The real deal: a Bitcoin block header



The real deal: a Bitcoin block header



The real deal: coinbase transaction


```
"in":[
  {
    "prev_out":{
      "hash":"000000.....0000000",
      "n":4294967295
    },
  "coinbase":"..."
  },
 "out":[
{
  "value":"12.53371419",
  "scriptPubKey":"OPDUP OPHASH160 ... ”
  }
```

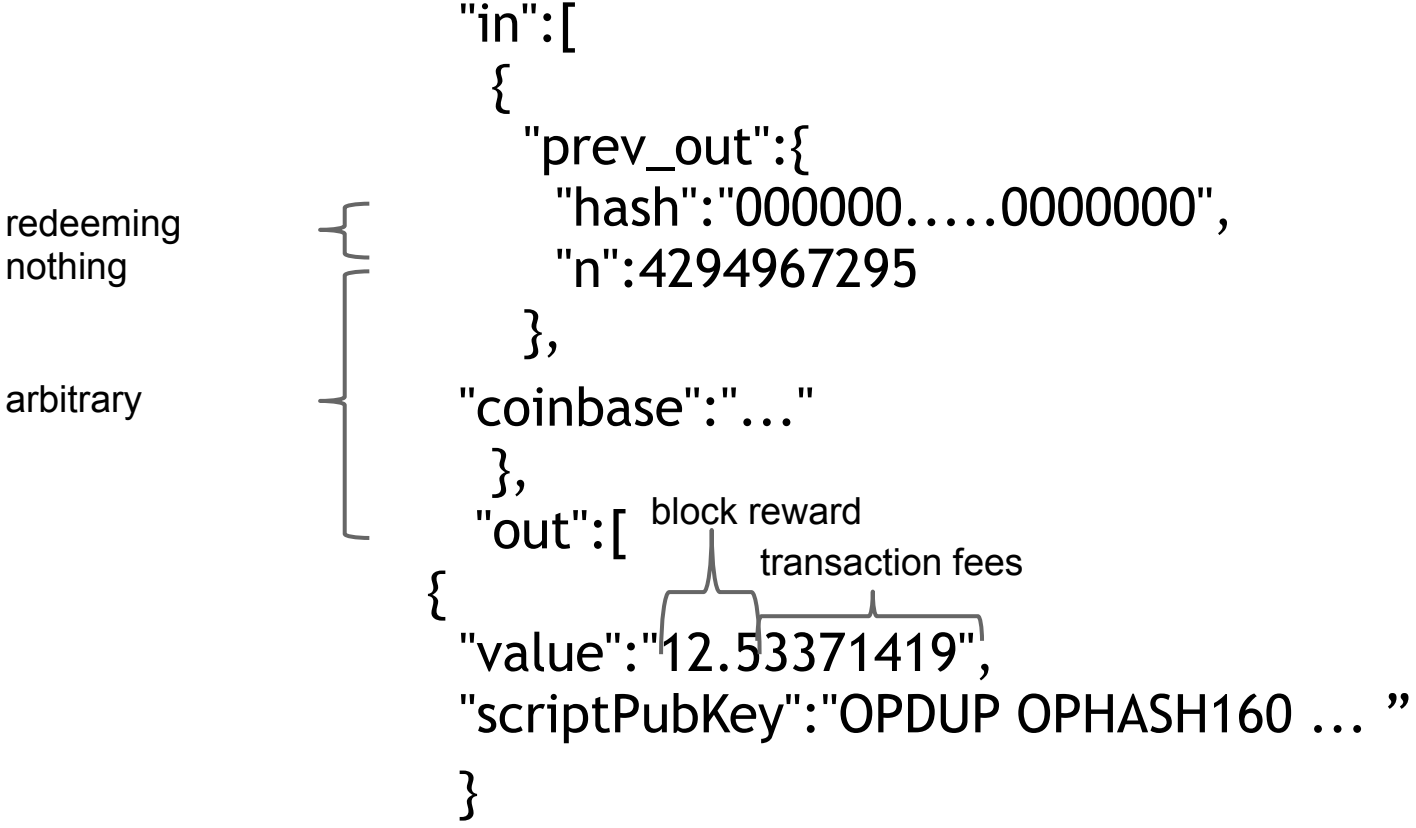
The real deal: coinbase transaction

redeeming nothing

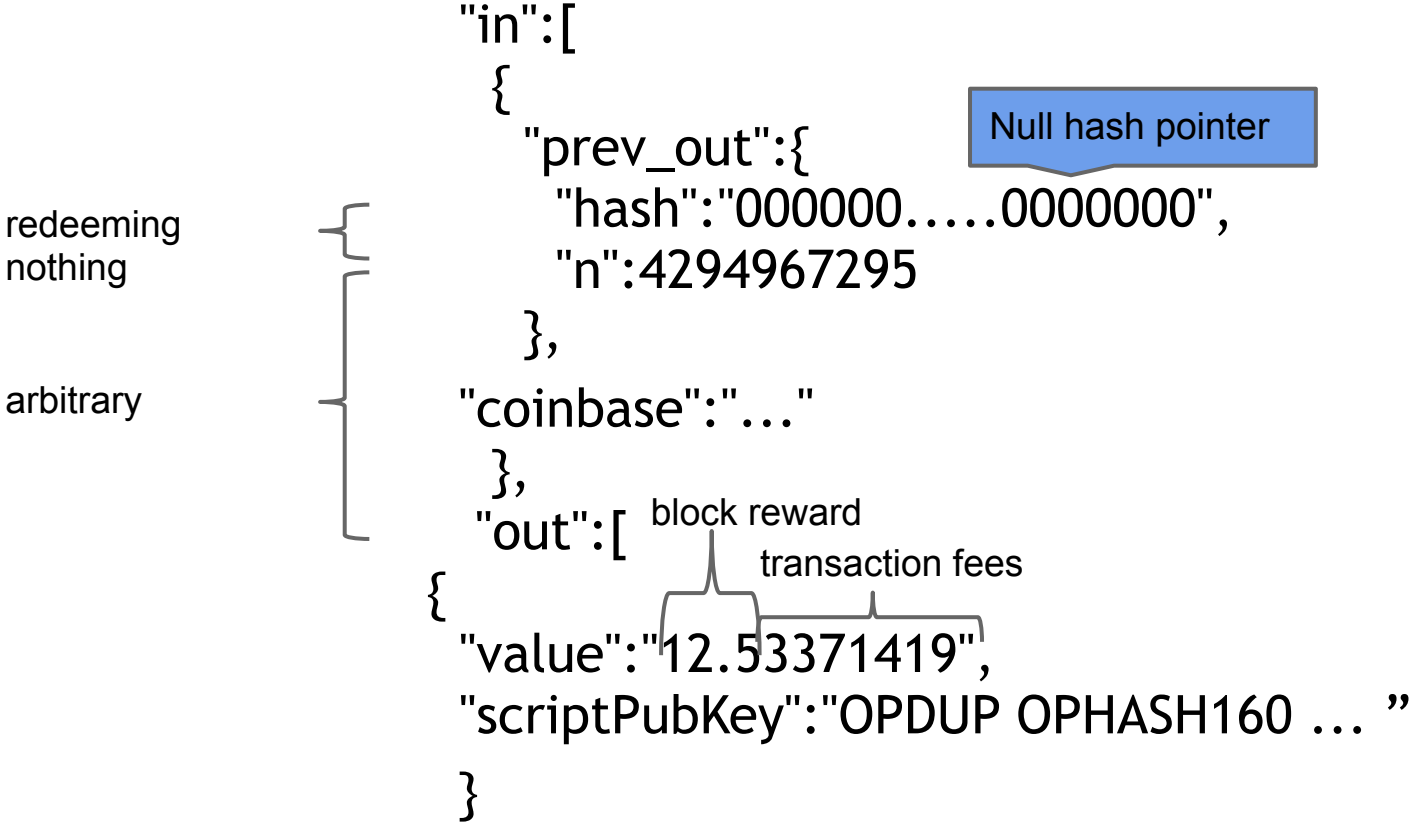
arbitrary


```
"in":[
 {
  "prev_out":{
   "hash":"000000.....0000000",
   "n":4294967295
  },
 "coinbase":"..."
 },
 "out":[
{
 "value":"12.53371419",
 "scriptPubKey":"OPDUP OPHASH160 ... ”
 }
```

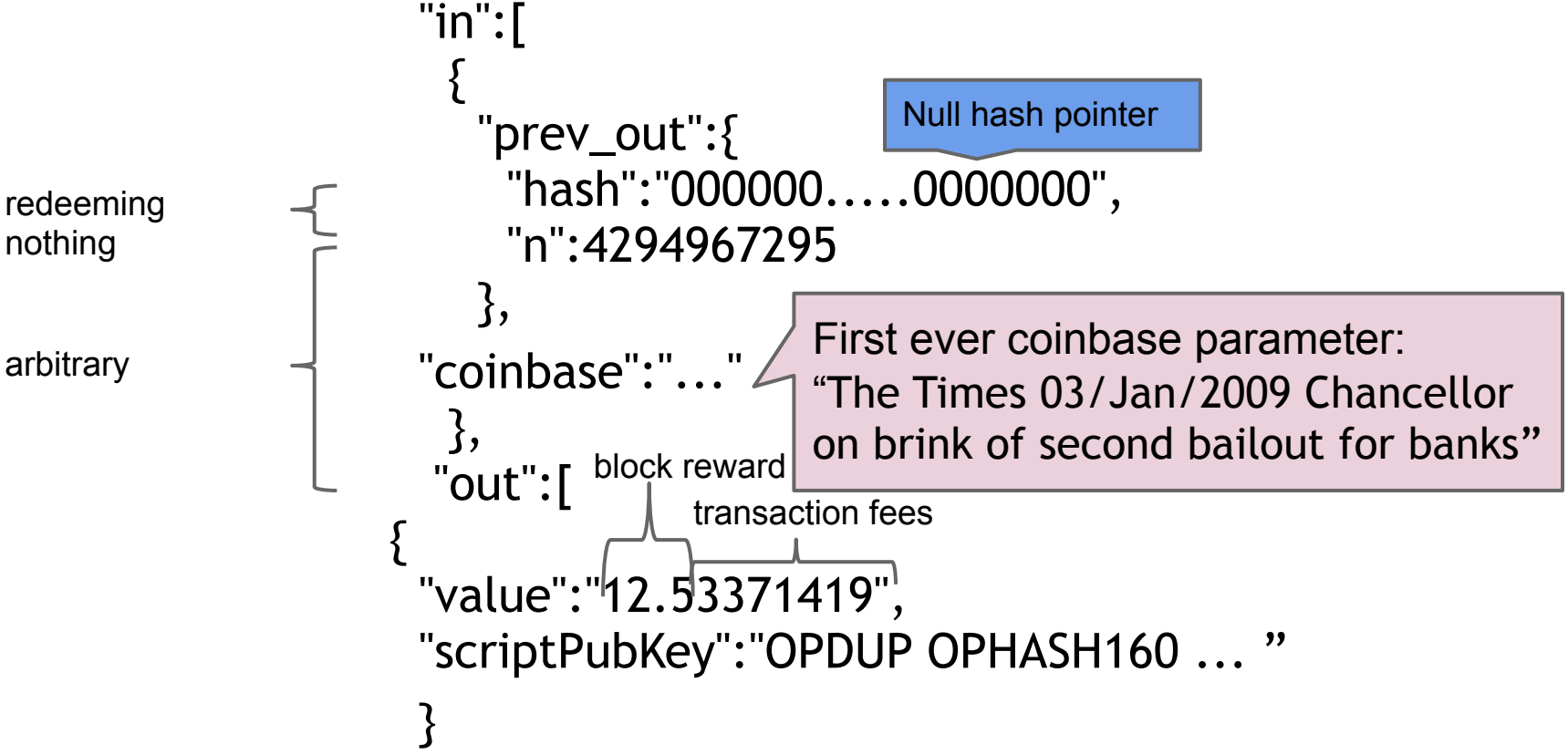
The real deal: coinbase transaction



The real deal: coinbase transaction



The real deal: coinbase transaction



See for yourself!

Transaction View information about a bitcoin transaction

151b750d1f13e76d84e82b34b12688811b23a8e3119a1cba4b4810f9b0ef408d

1KryFUt9tXHvaoCYTNPbqpWPJKQ717YmL5 → 1KvrdrQ3oGqMAiDTMEYCcdDSnVaGNW2YZh 1.0194 BTC
1KryFUt9tXHvaoCYTNPbqpWPJKQ717YmL5 3.458 BTC

9 Confirmations | 4.4774 BTC

Summary	
Size	257 (bytes)
Received Time	2014-08-05 01:55:25
Included In Blocks	314018 (2014-08-05 02:00:40 +5 minutes)
Confirmations	9 Confirmations
Relayed by IP	Blockchain.info
Visualize	View Tree Chart

Inputs and Outputs	
Total Input	4.4775 BTC
Total Output	4.4774 BTC
Fees	0.0001 BTC
Estimated BTC Transacted	1.0194 BTC
Scripts	Show scripts & coinbase

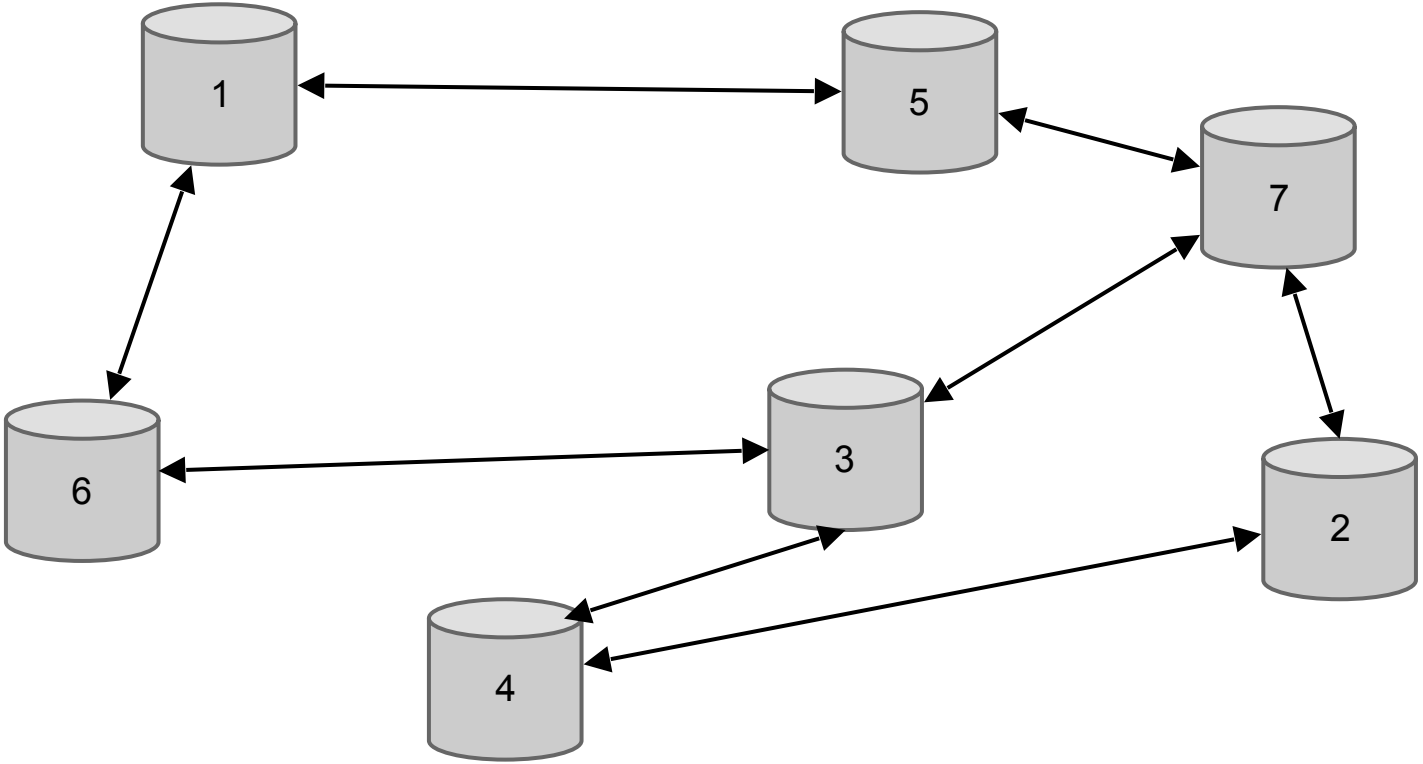
blockchain.info (and many other sites)

The Bitcoin network

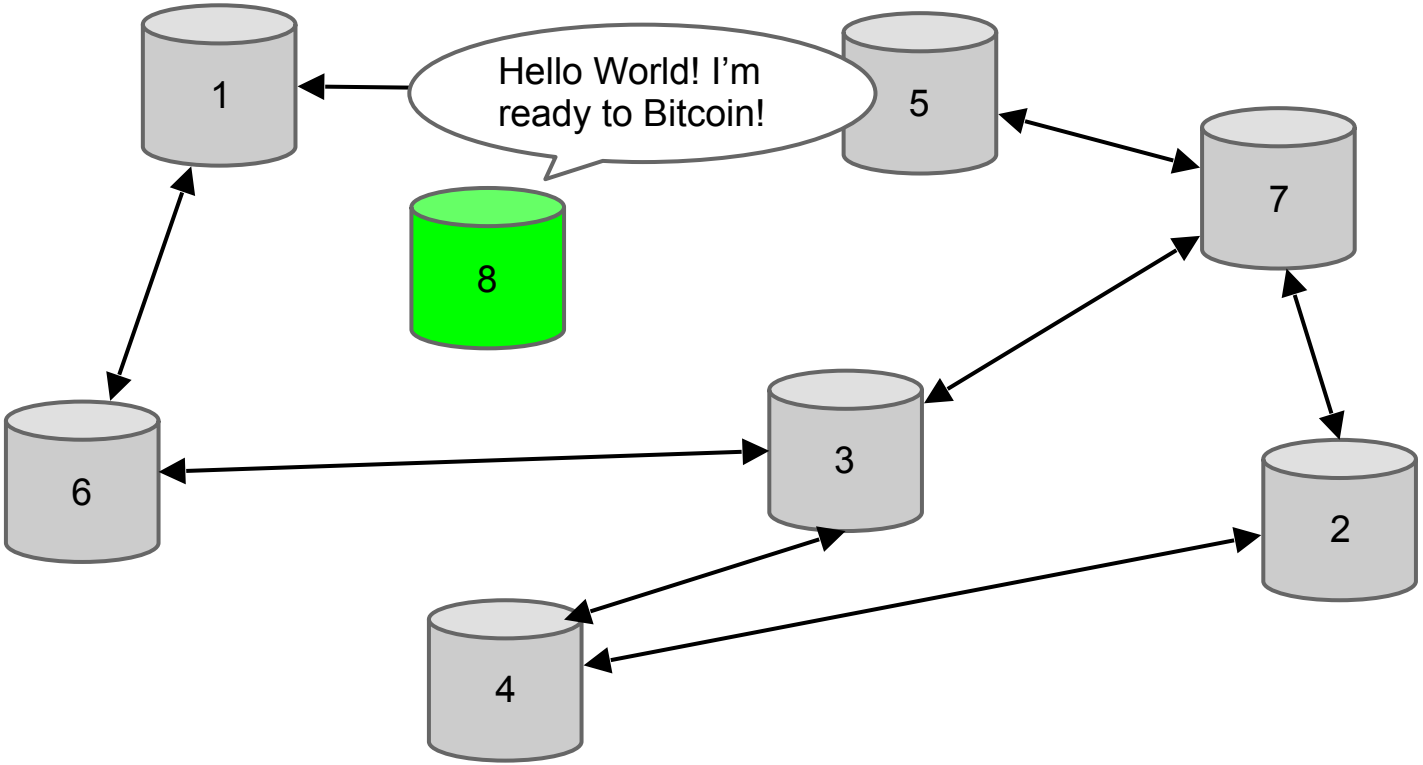
Bitcoin P2P network

- Ad-hoc protocol (runs on TCP port 8333)
- Ad-hoc network with random topology
- All nodes are equal
- New nodes can join at any time
- Forget non-responding nodes after 3 hr

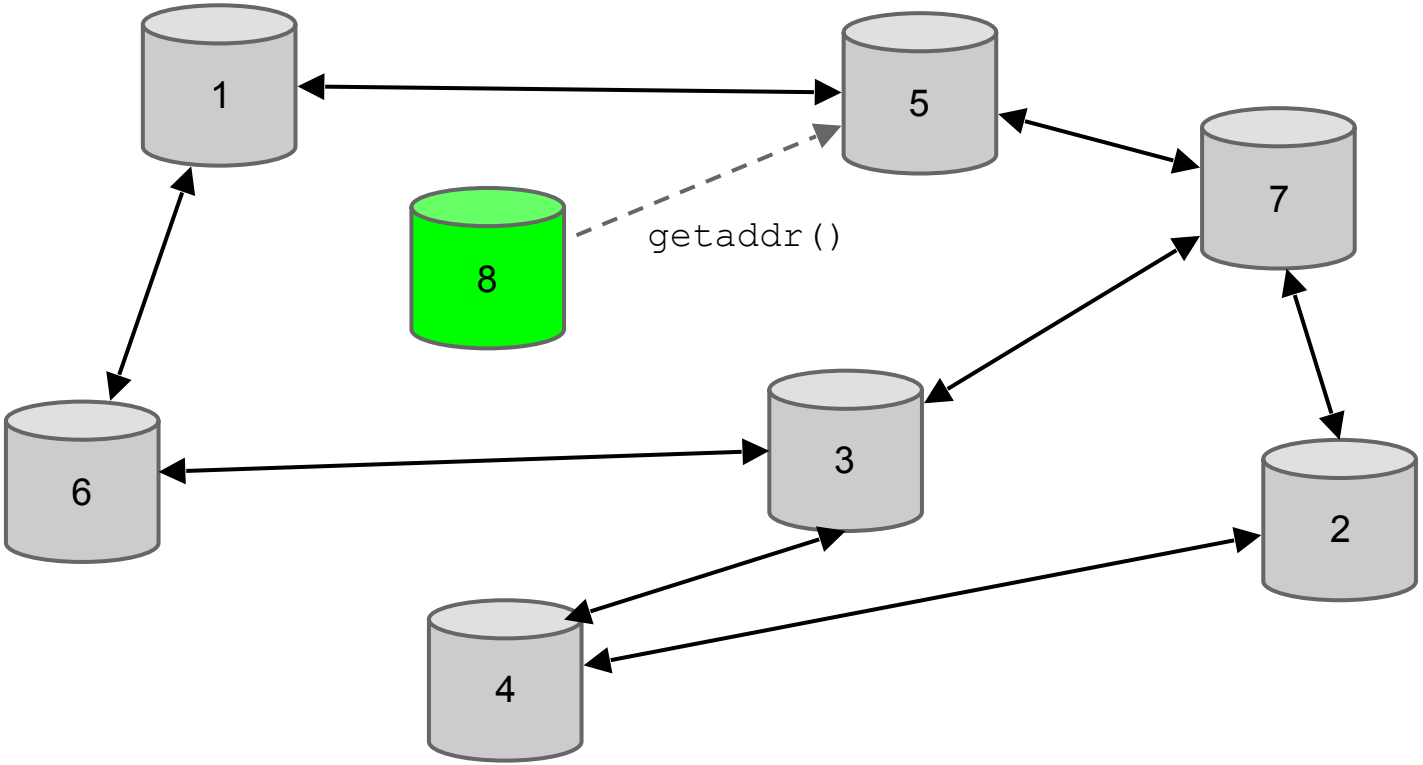
Joining the Bitcoin P2P network



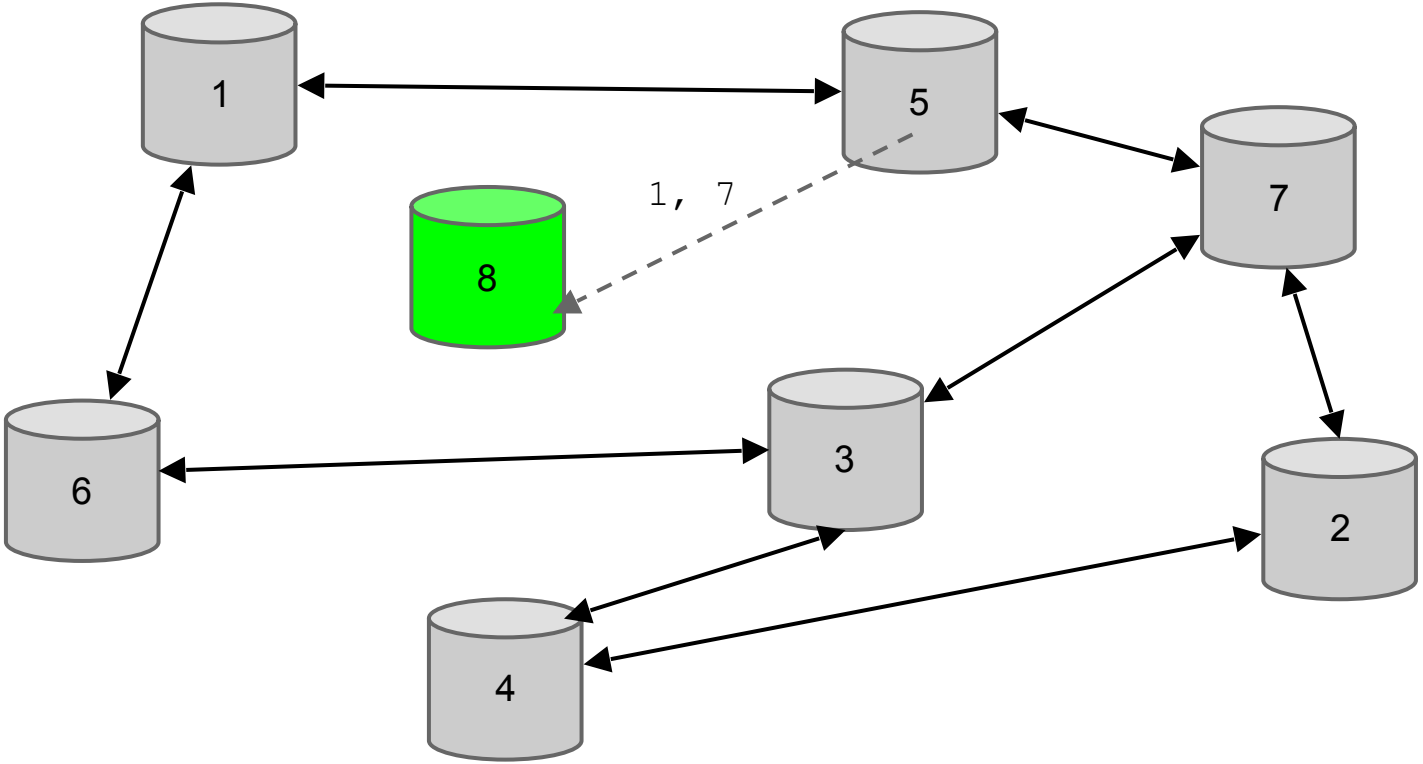
Joining the Bitcoin P2P network



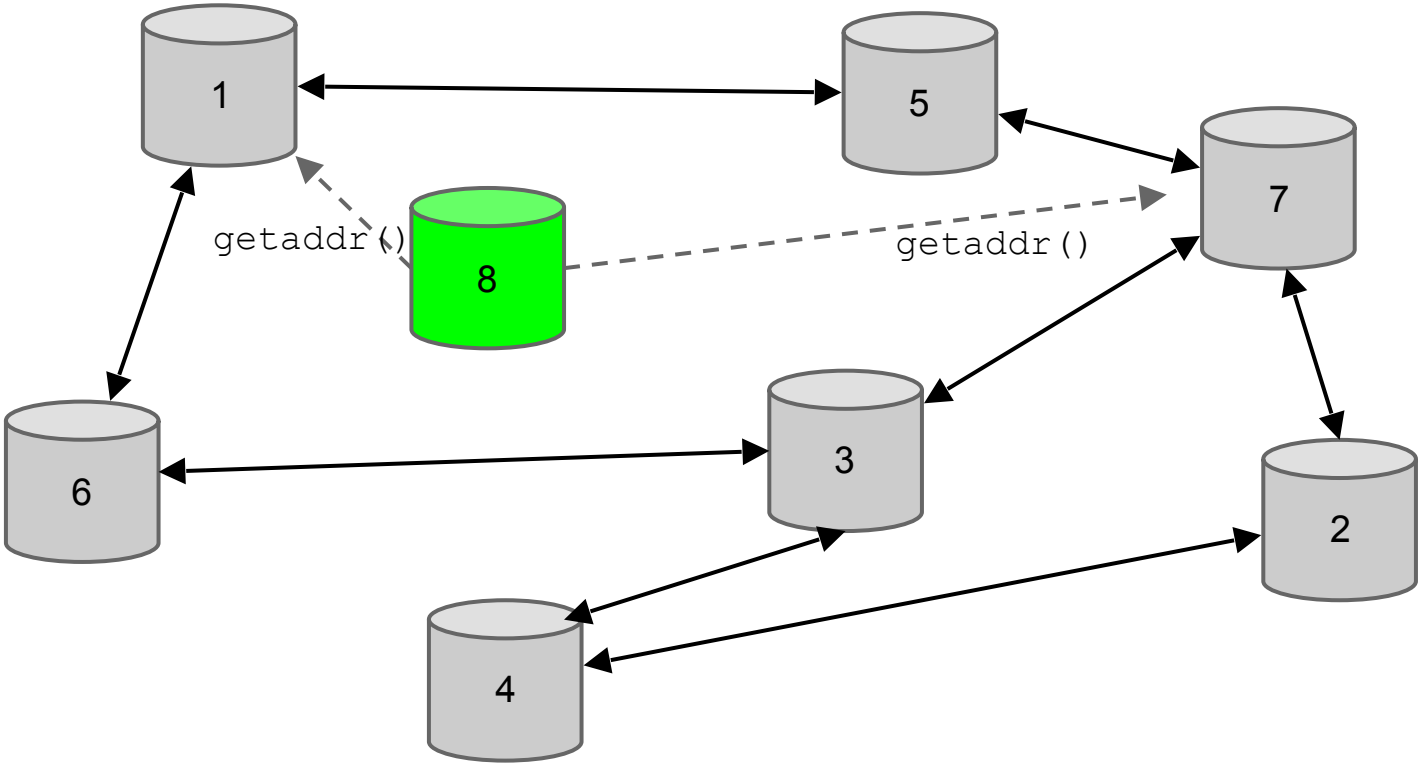
Joining the Bitcoin P2P network



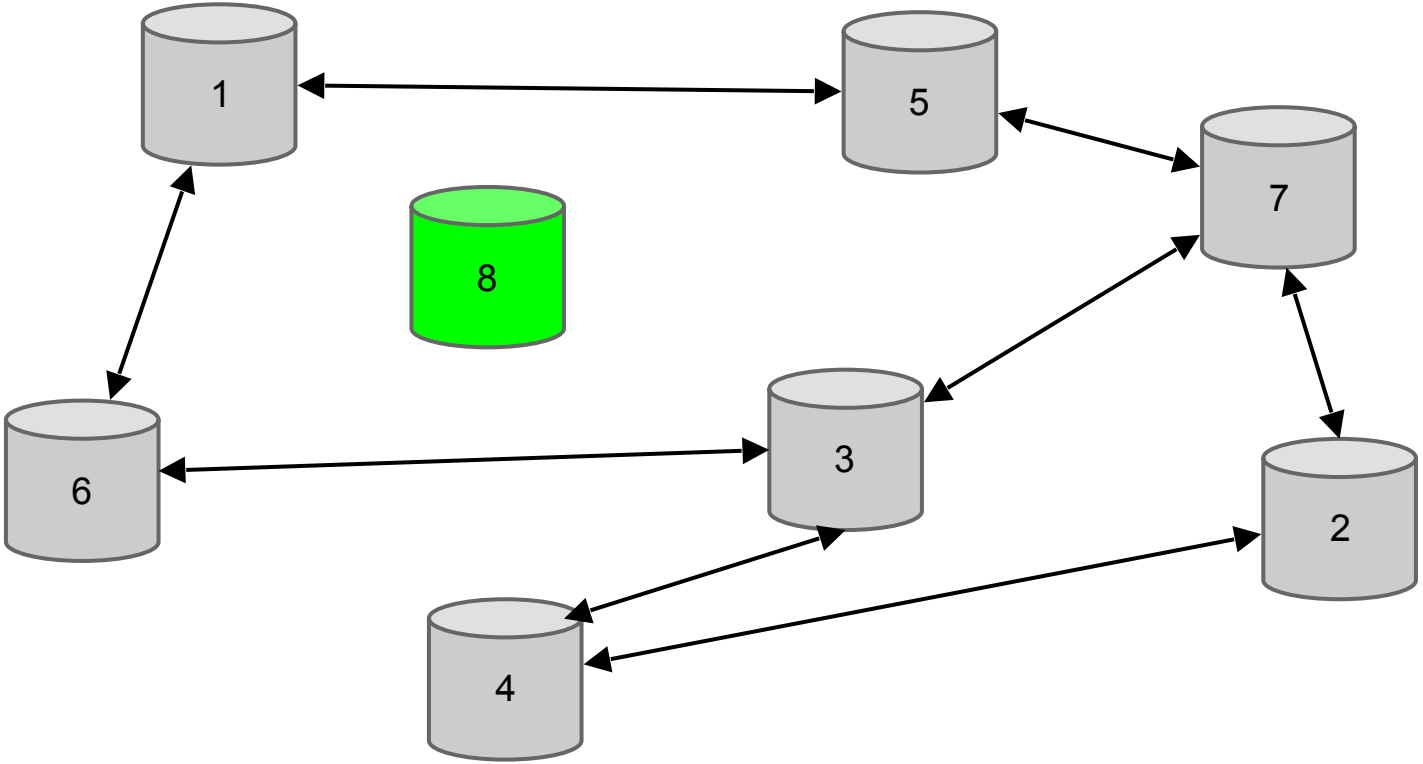
Joining the Bitcoin P2P network



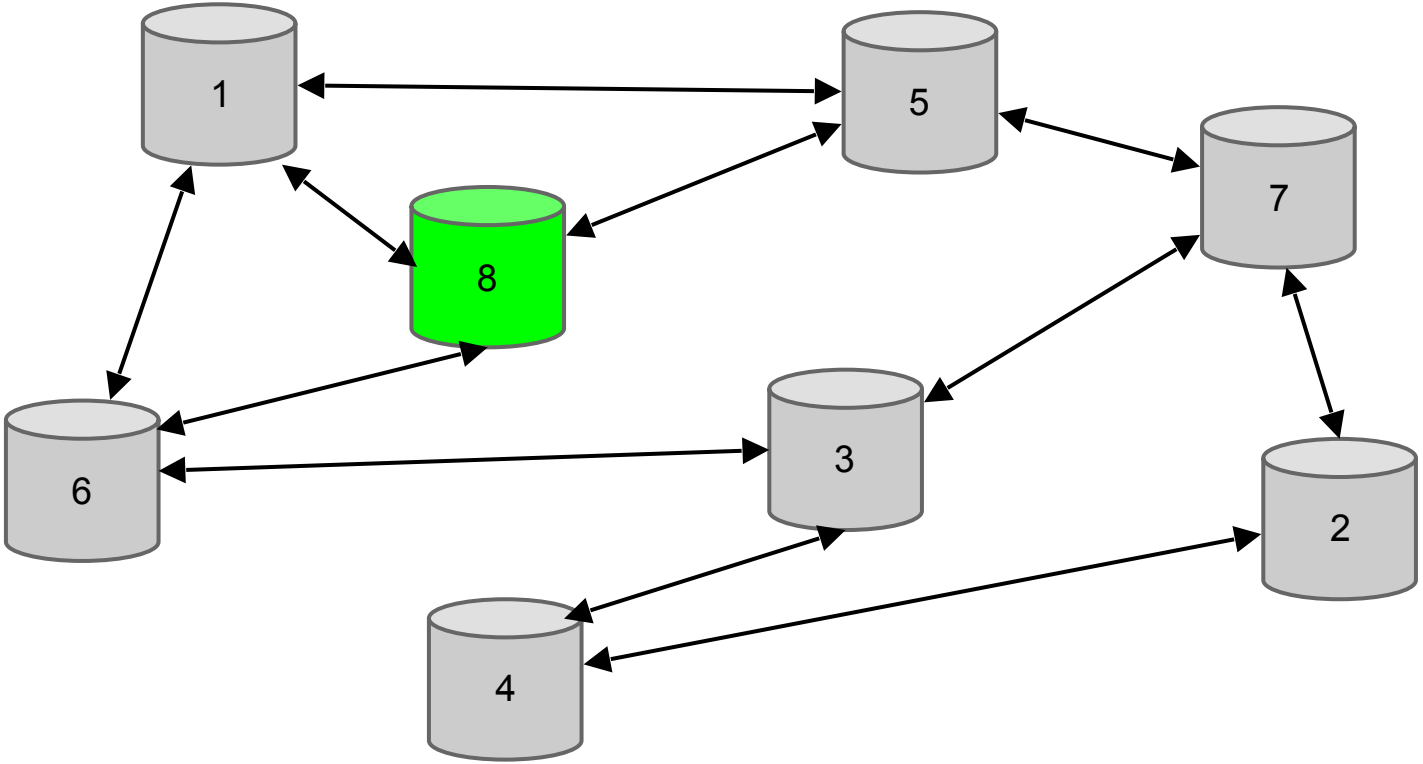
Joining the Bitcoin P2P network



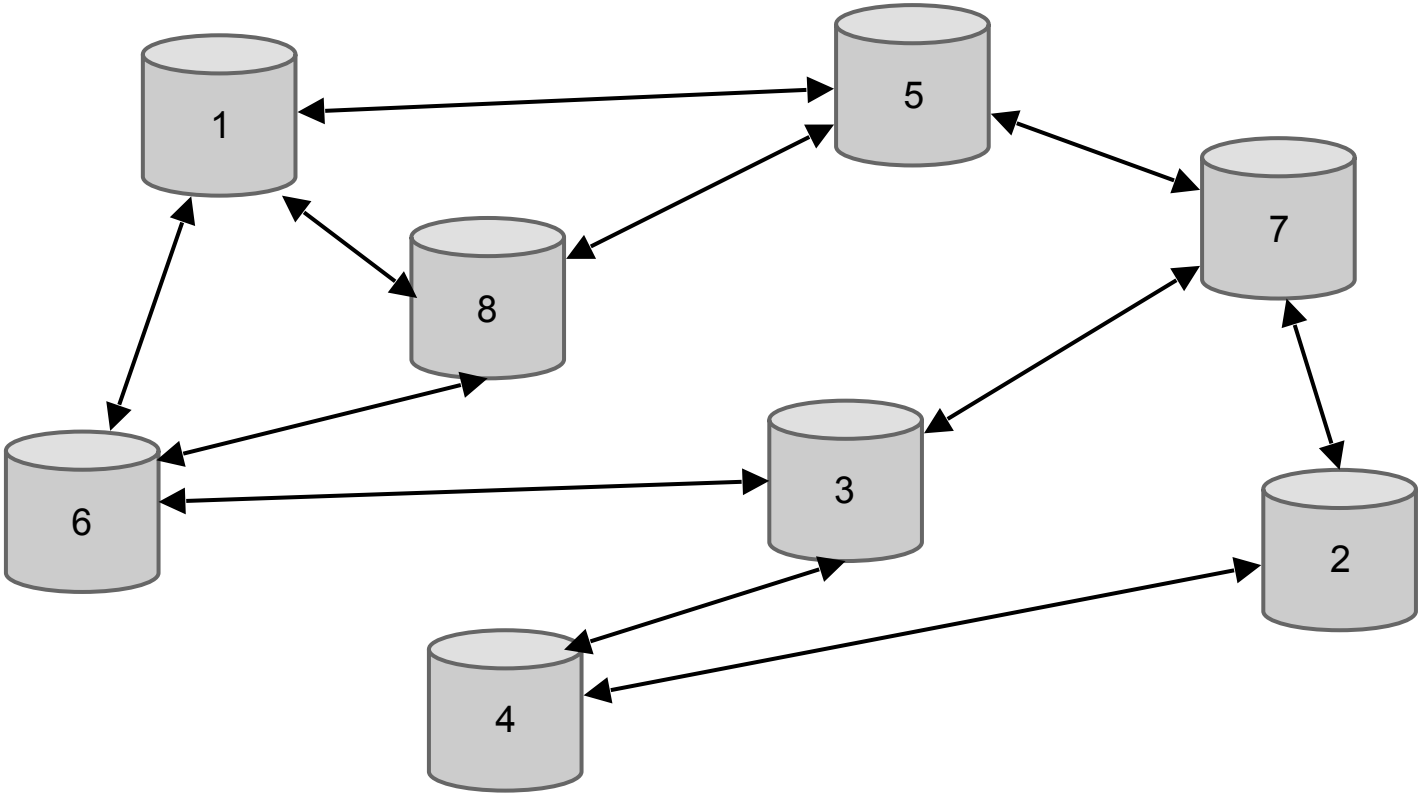
Joining the Bitcoin P2P network



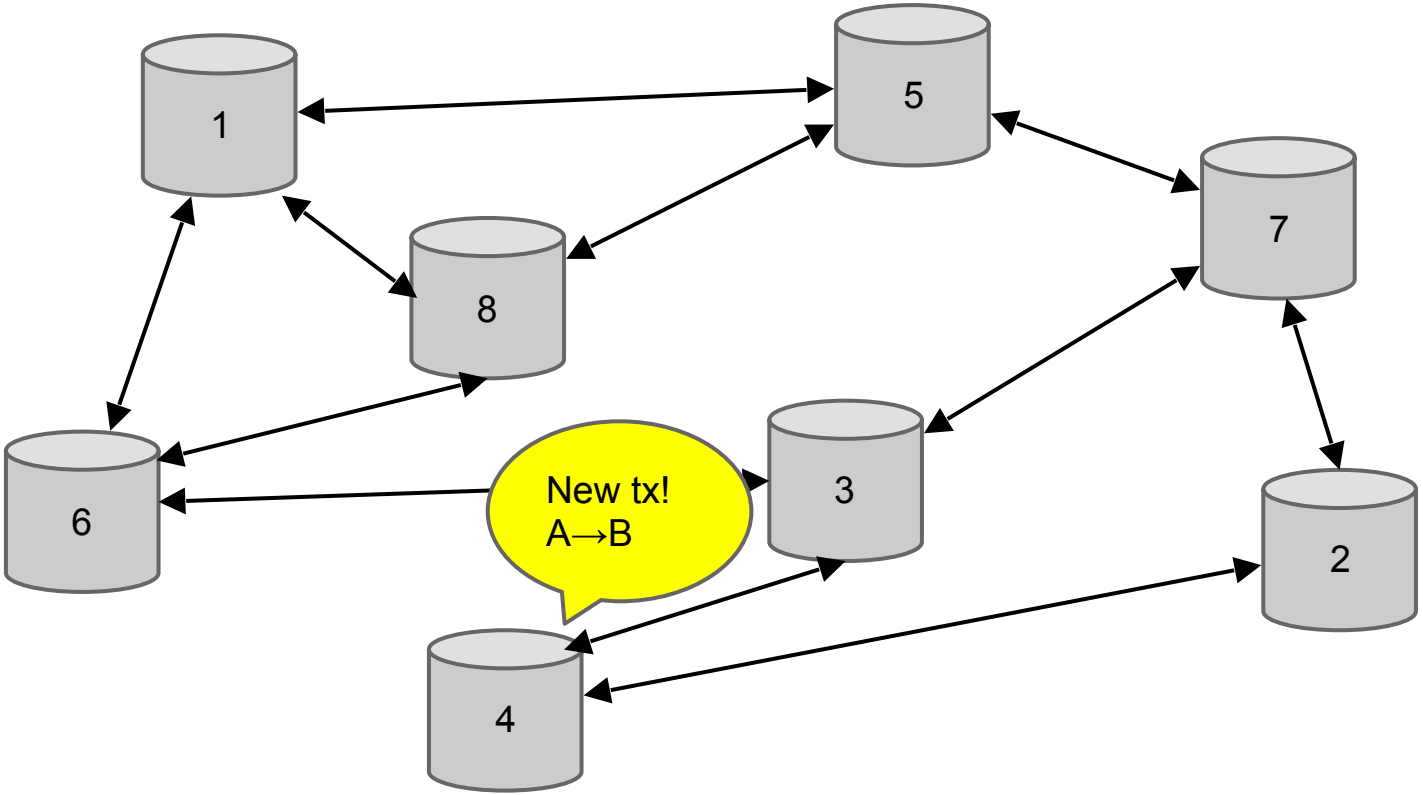
Joining the Bitcoin P2P network



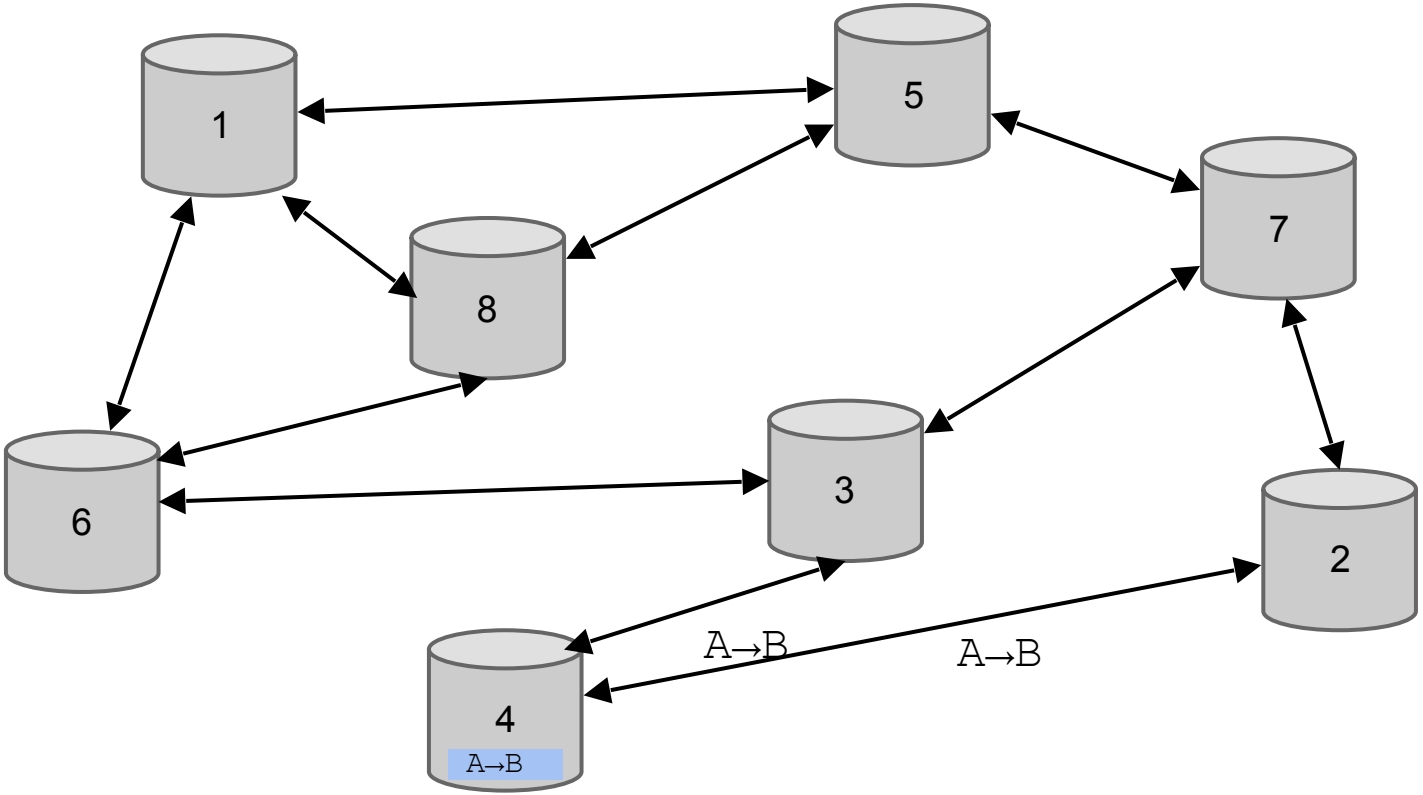
Transaction propagation (flooding)



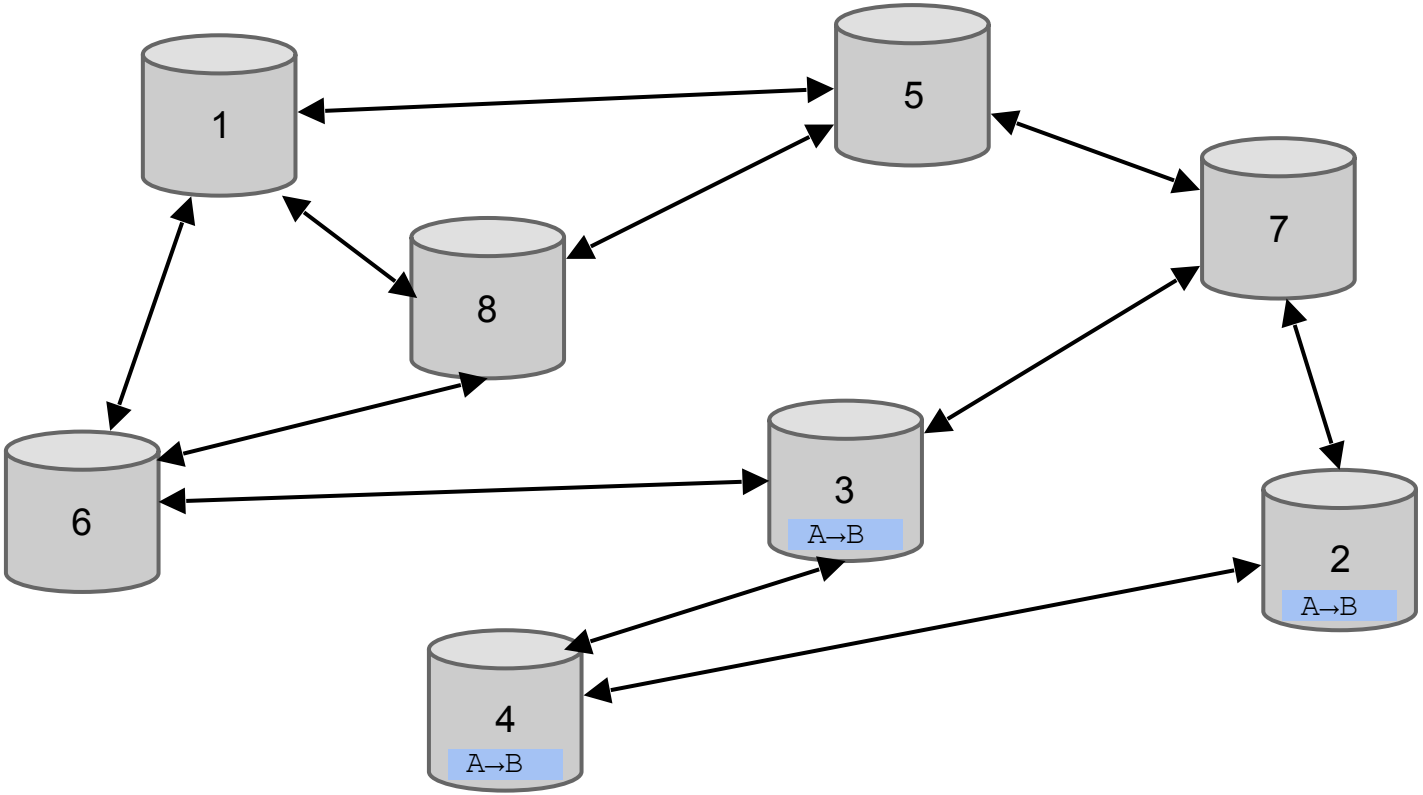
Transaction propagation (flooding)



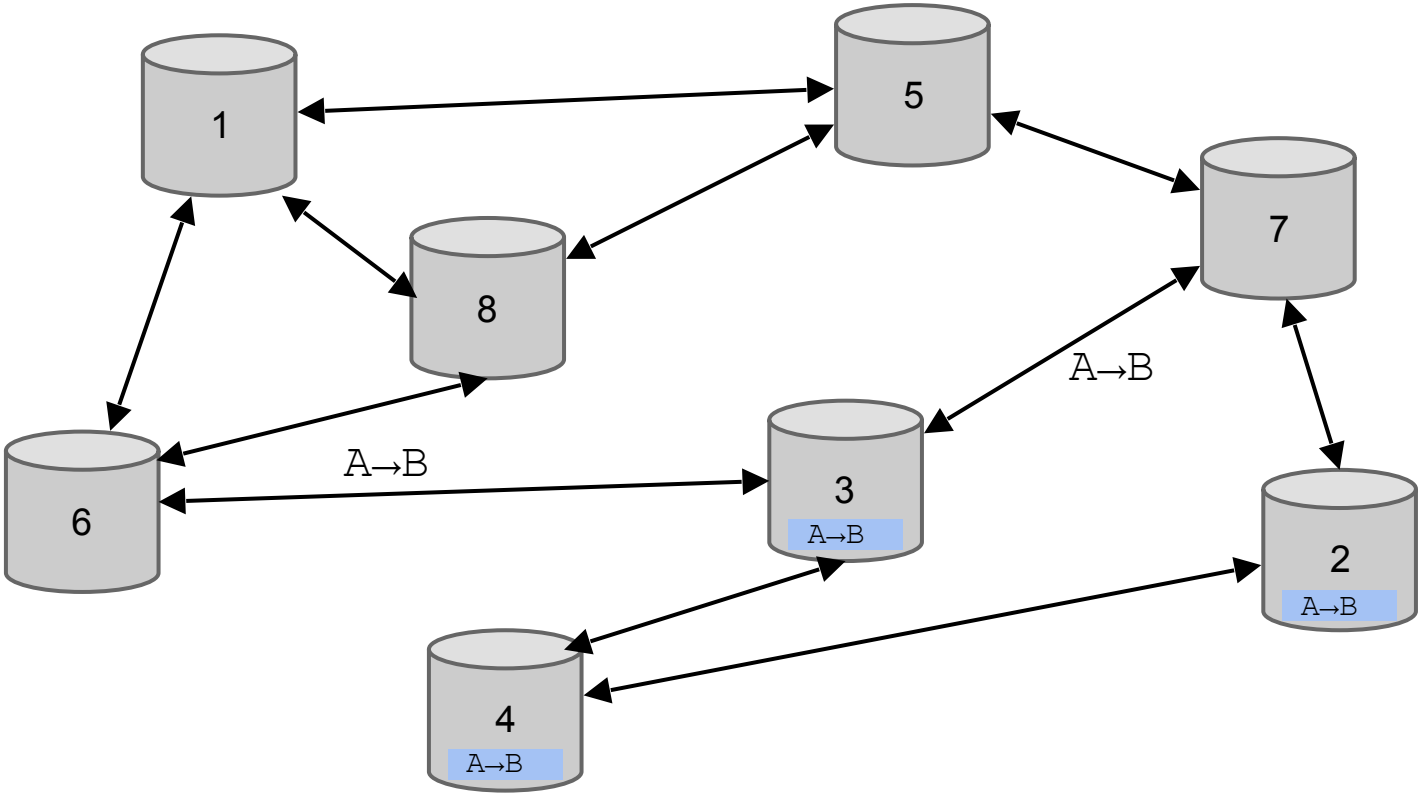
Transaction propagation (flooding)



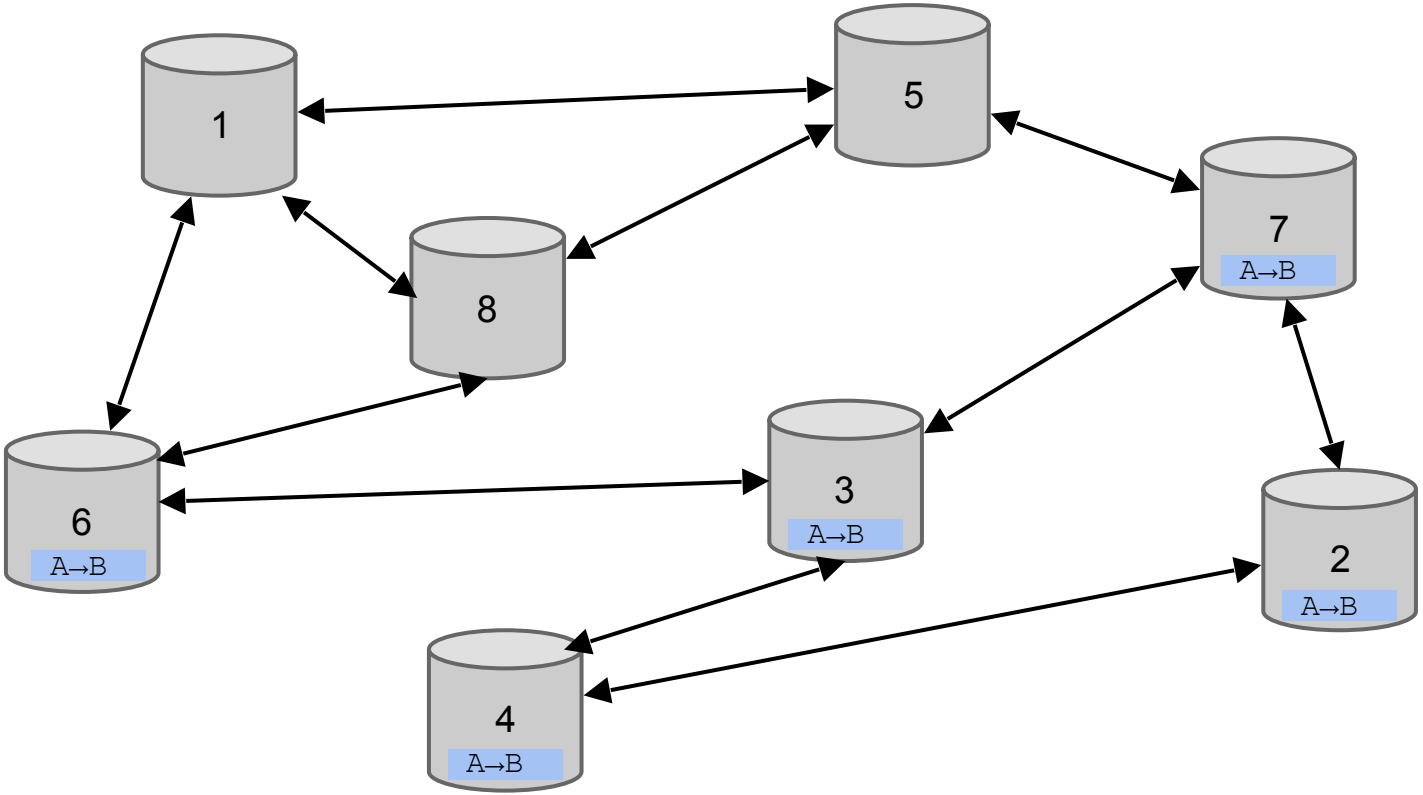
Transaction propagation (flooding)



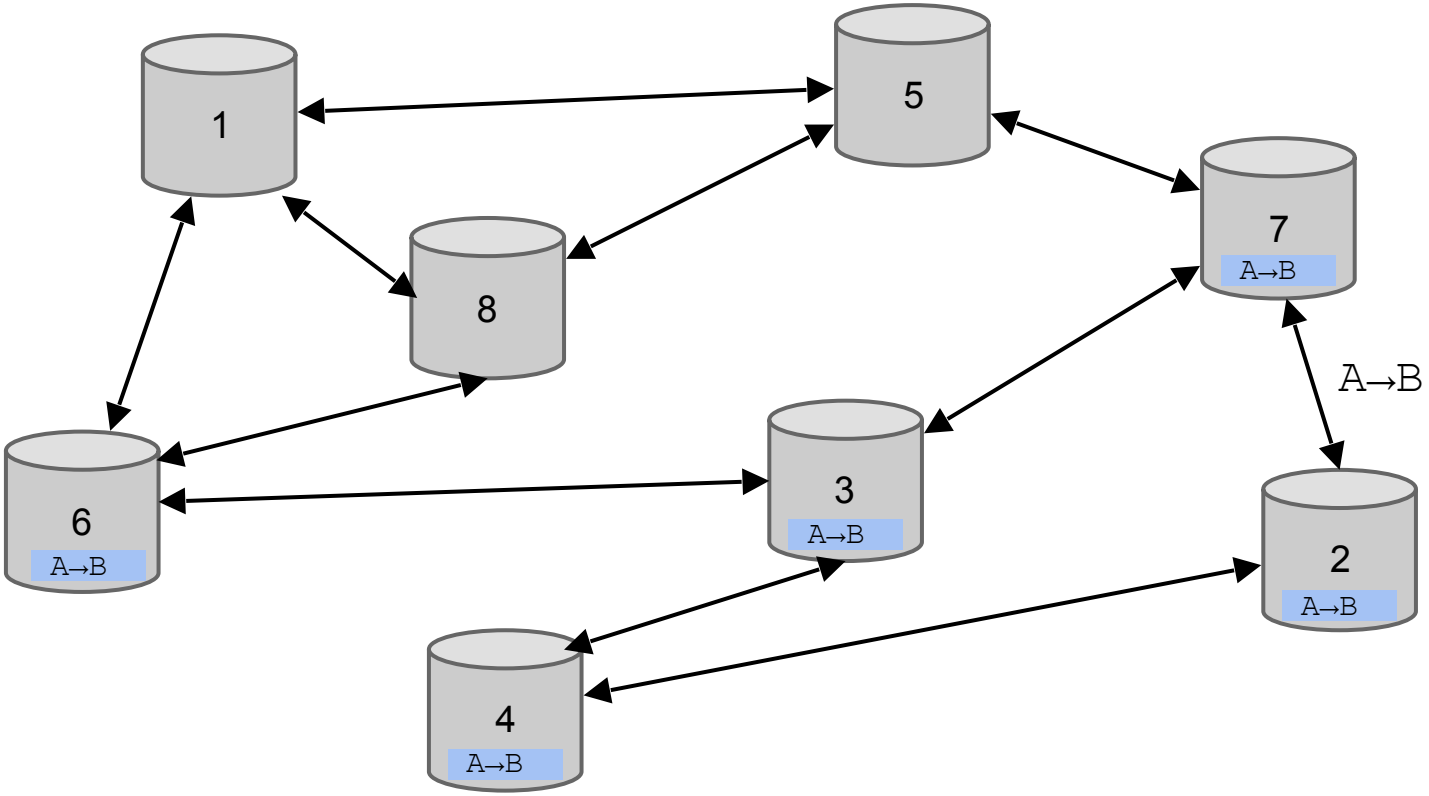
Transaction propagation (flooding)



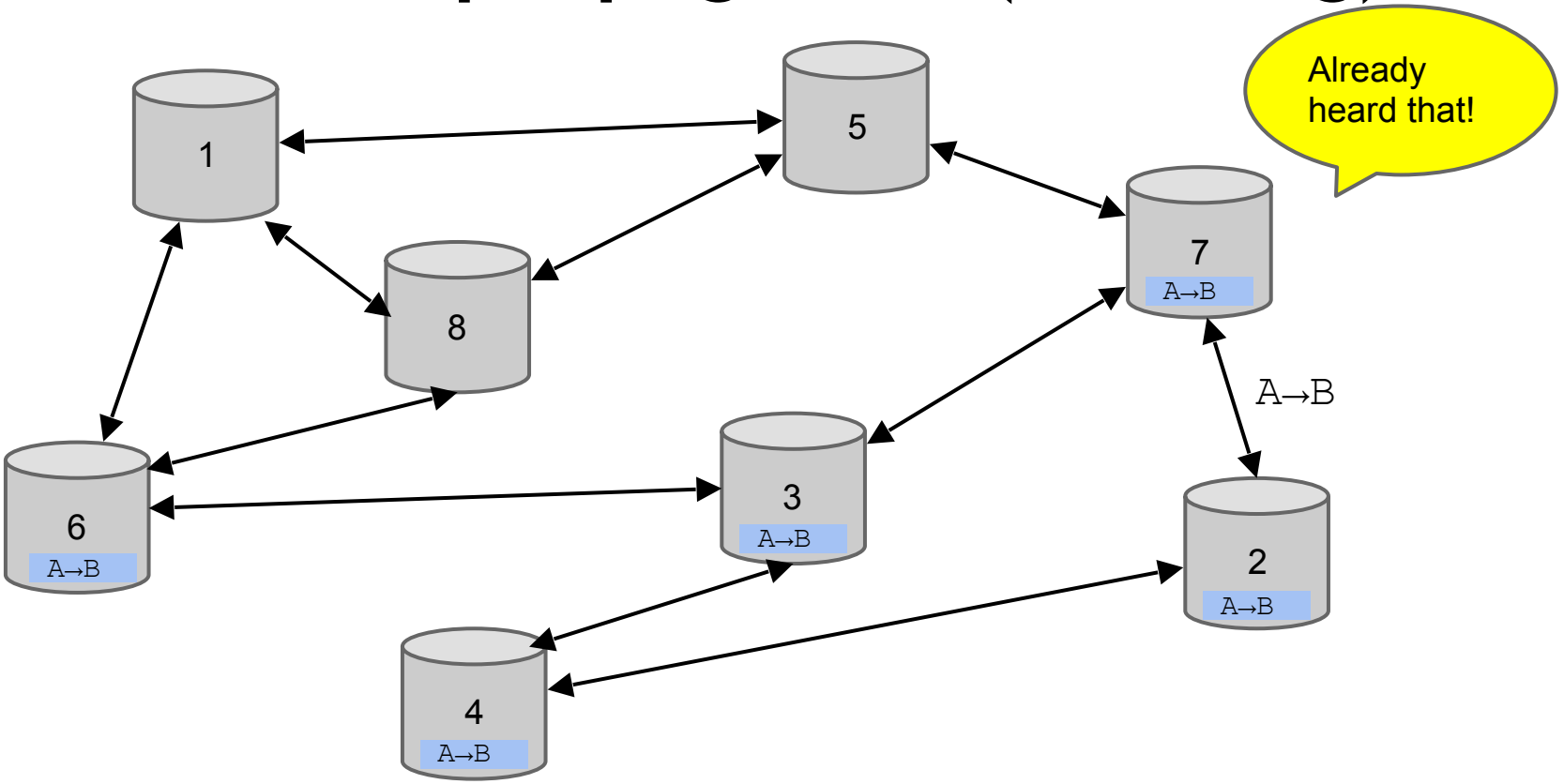
Transaction propagation (flooding)



Transaction propagation (flooding)



Transaction propagation (flooding)

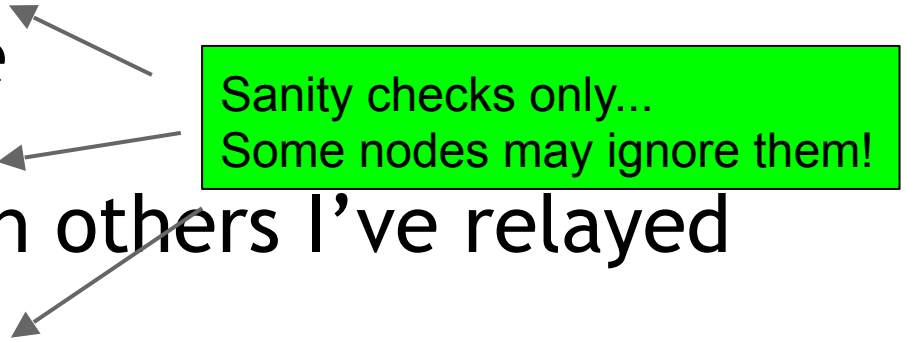


Should I relay a proposed transaction?

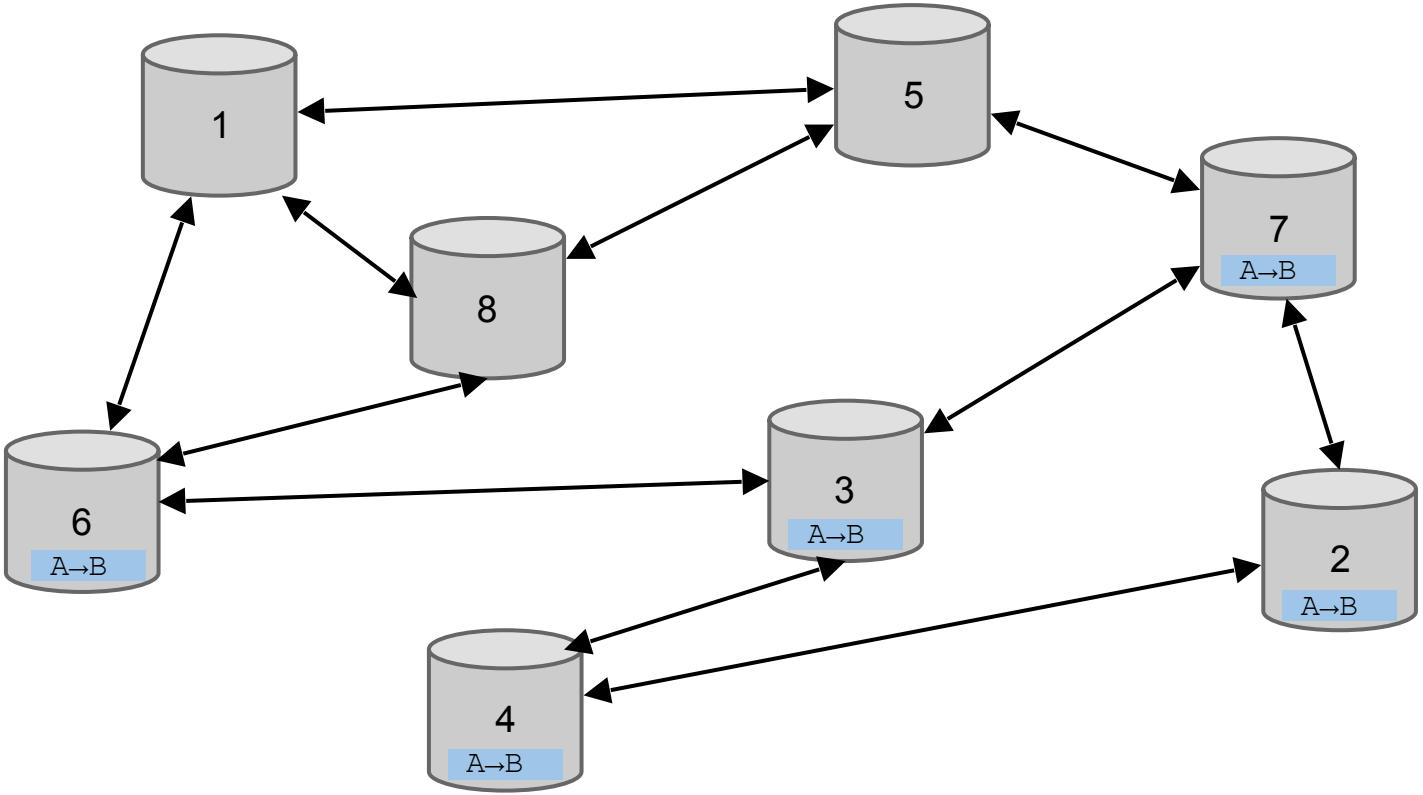
- Transaction valid with current block chain
- (default) script matches a whitelist
  - Avoid unusual scripts
- Haven’t seen before
  - Avoid infinite loops
- Doesn’t conflict with others I’ve relayed
  - Avoid double-spends

Should I relay a proposed transaction?

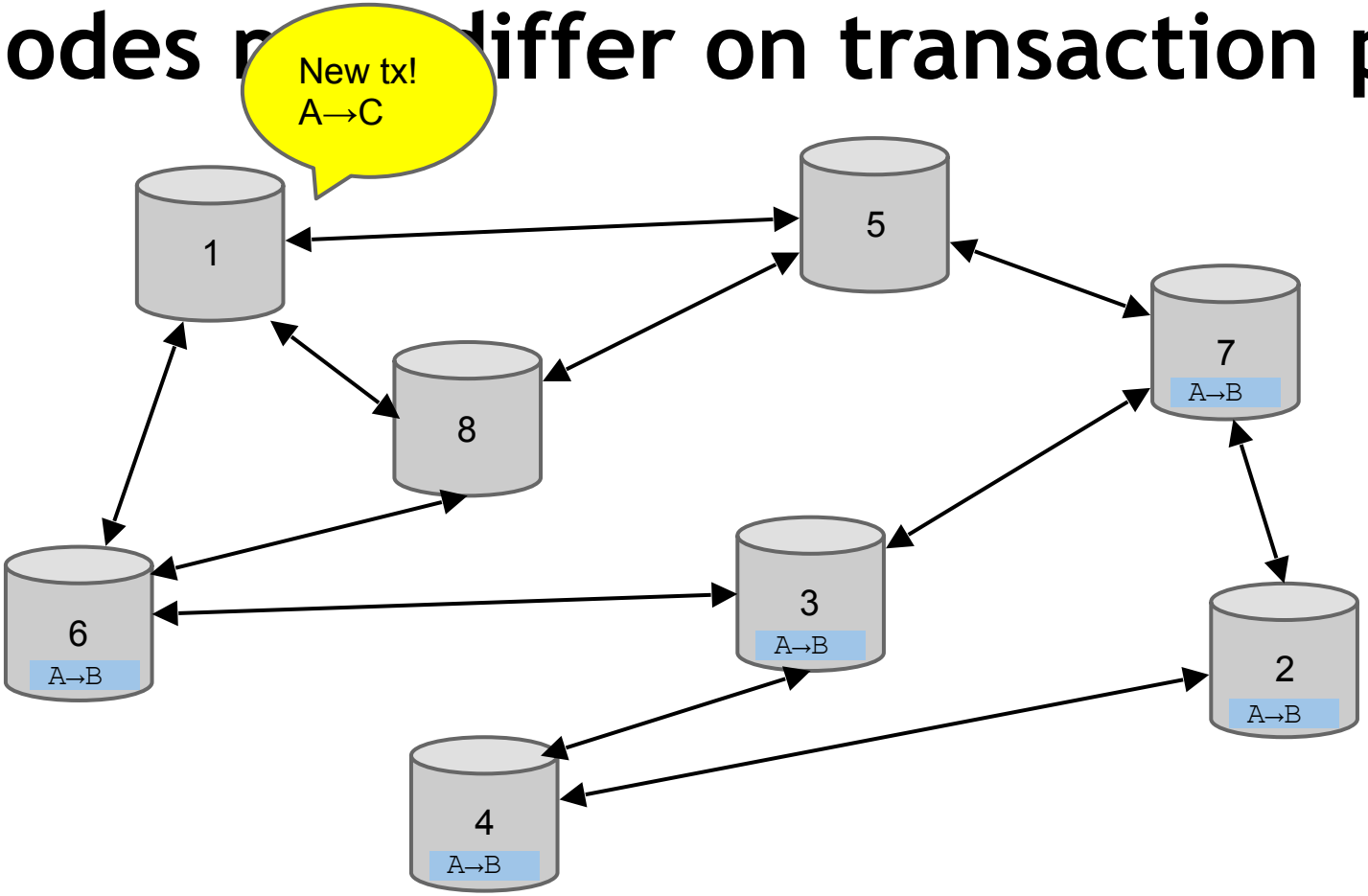
- Transaction valid with current block chain
- (default) script matches a whitelist
  - Avoid unusual scripts
- Haven’t seen before
  - Avoid infinite loops
- Doesn’t conflict with others I’ve relayed
  - Avoid double-spends



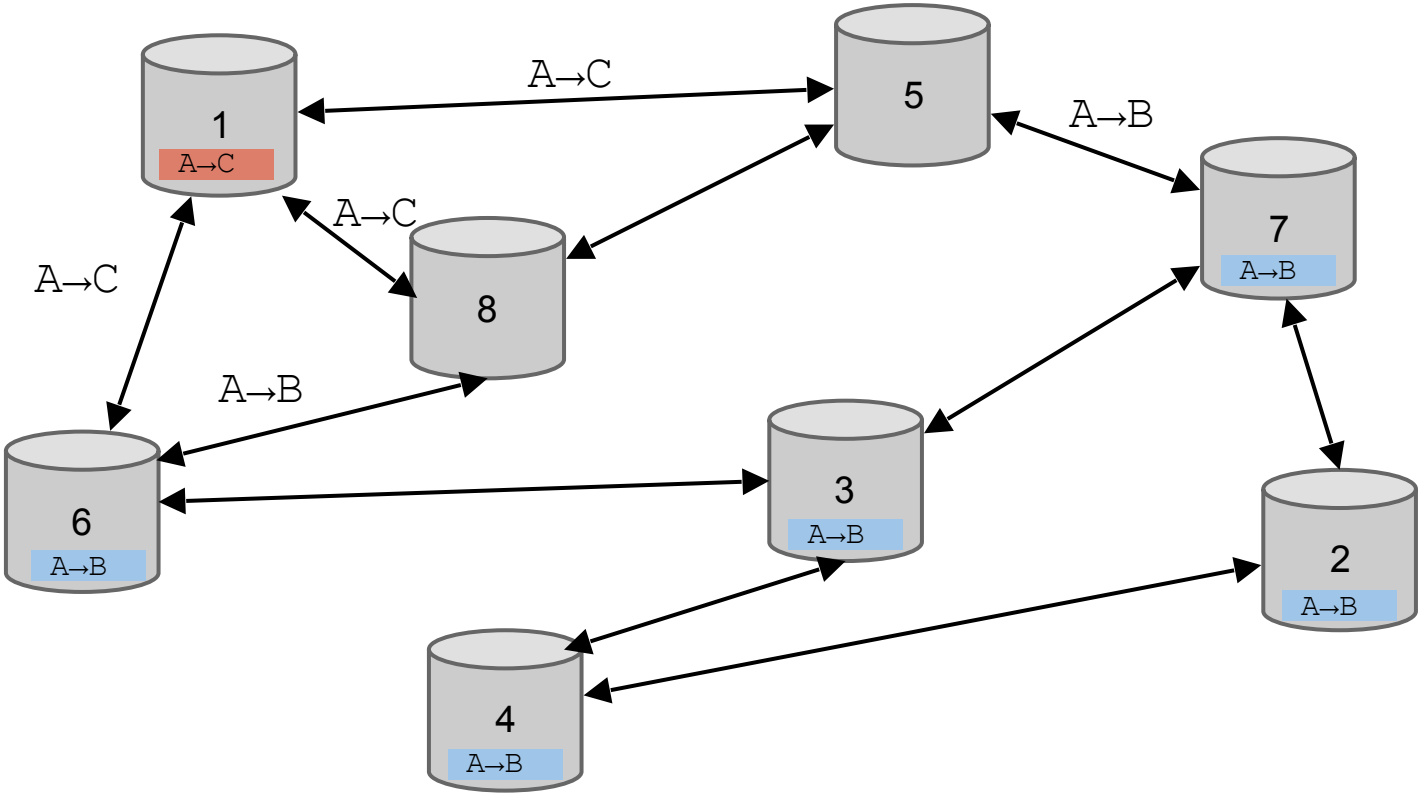
Nodes may differ on transaction pool



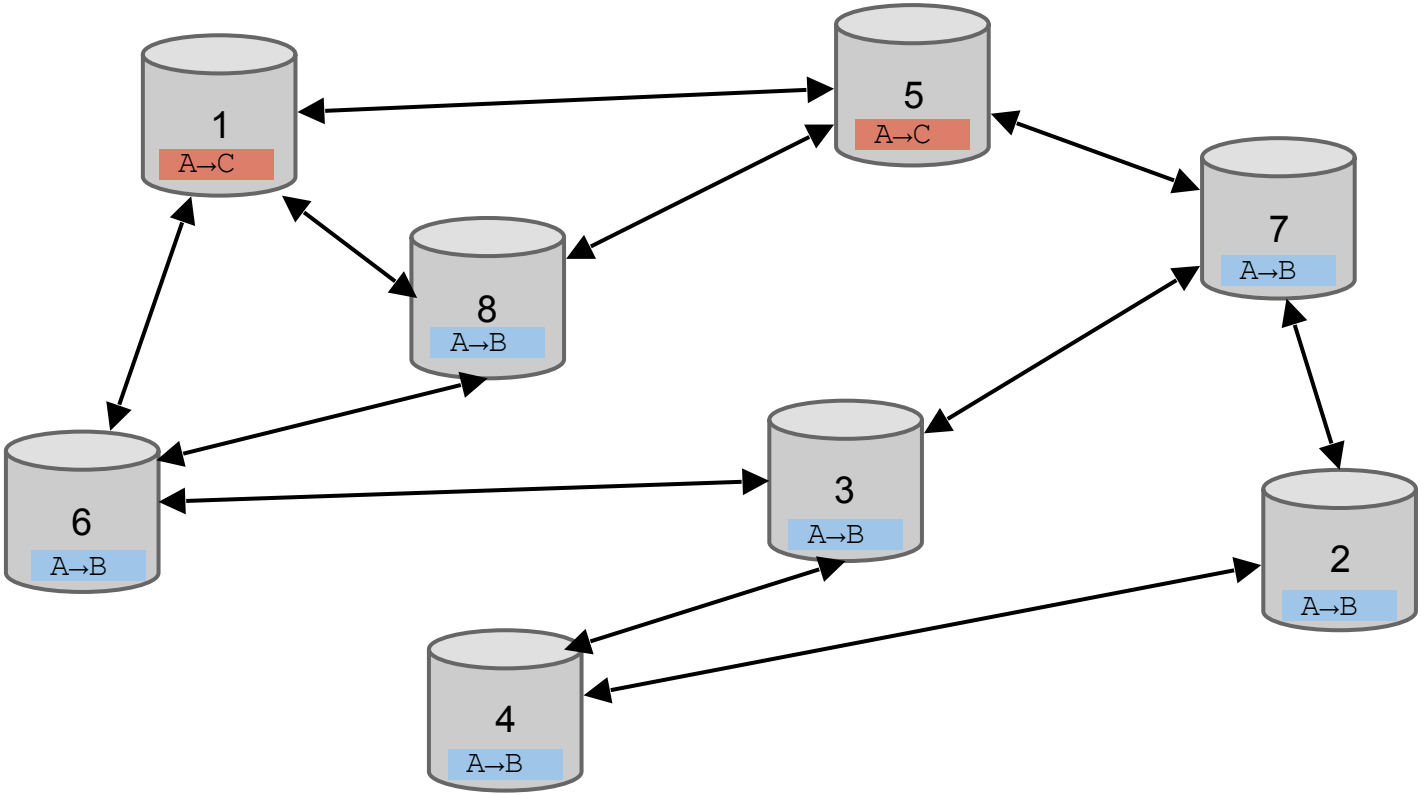
Nodes may differ on transaction pool



Nodes may differ on transaction pool



Nodes may differ on transaction pool



Race conditions

Transactions or blocks may *conflict*

- Default behavior: accept what you hear first
- Network position matters
- Miners may implement other logic!

Block propagation nearly identical

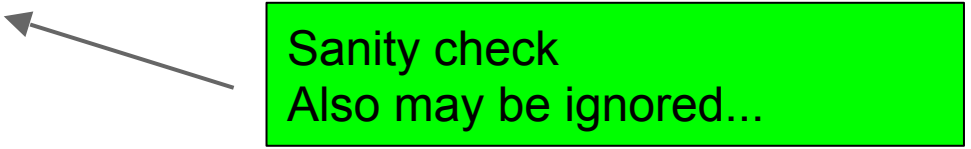
Relay a new block when you hear it if:

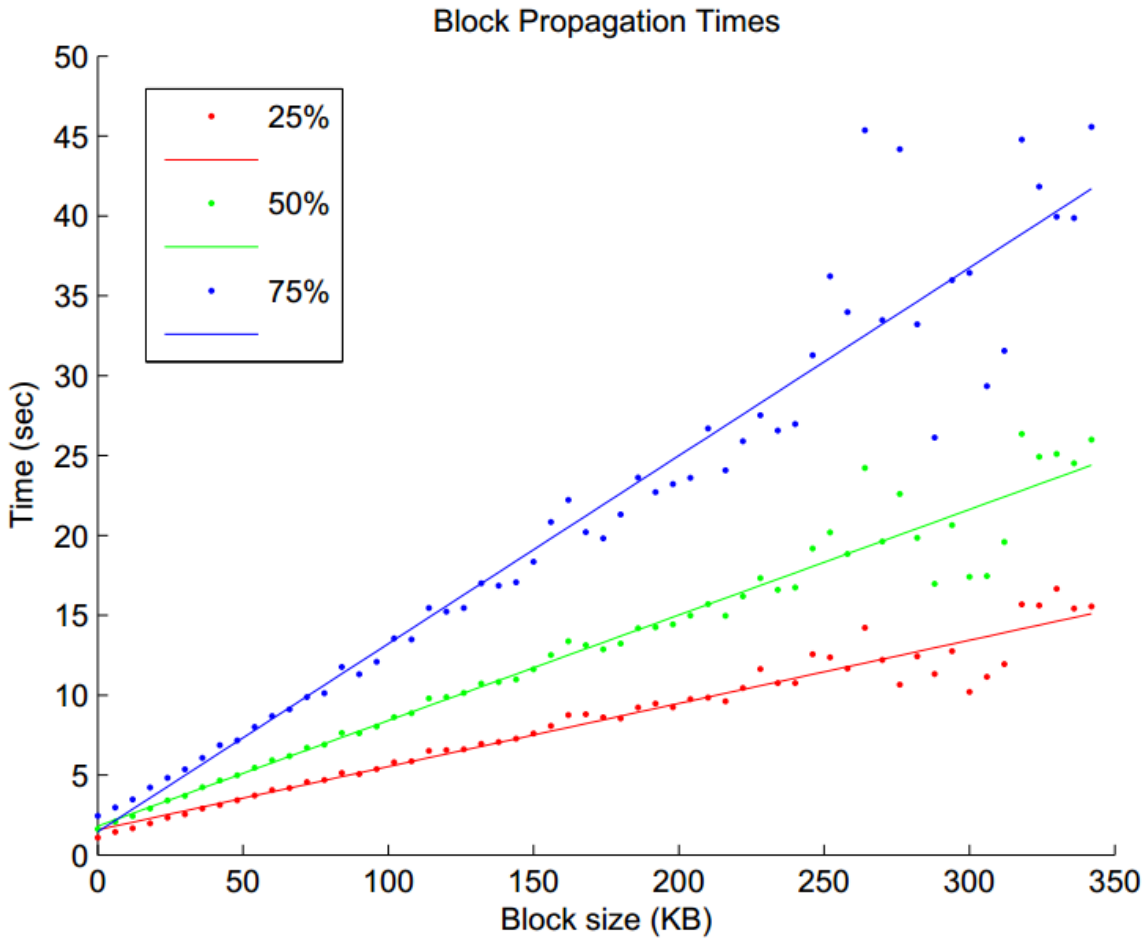
- Block meets the hash target
- Block has all valid transactions
  - Run *all* scripts, even if you wouldn't relay
- Block builds on current longest chain
  - Avoid forks

Block propagation nearly identical

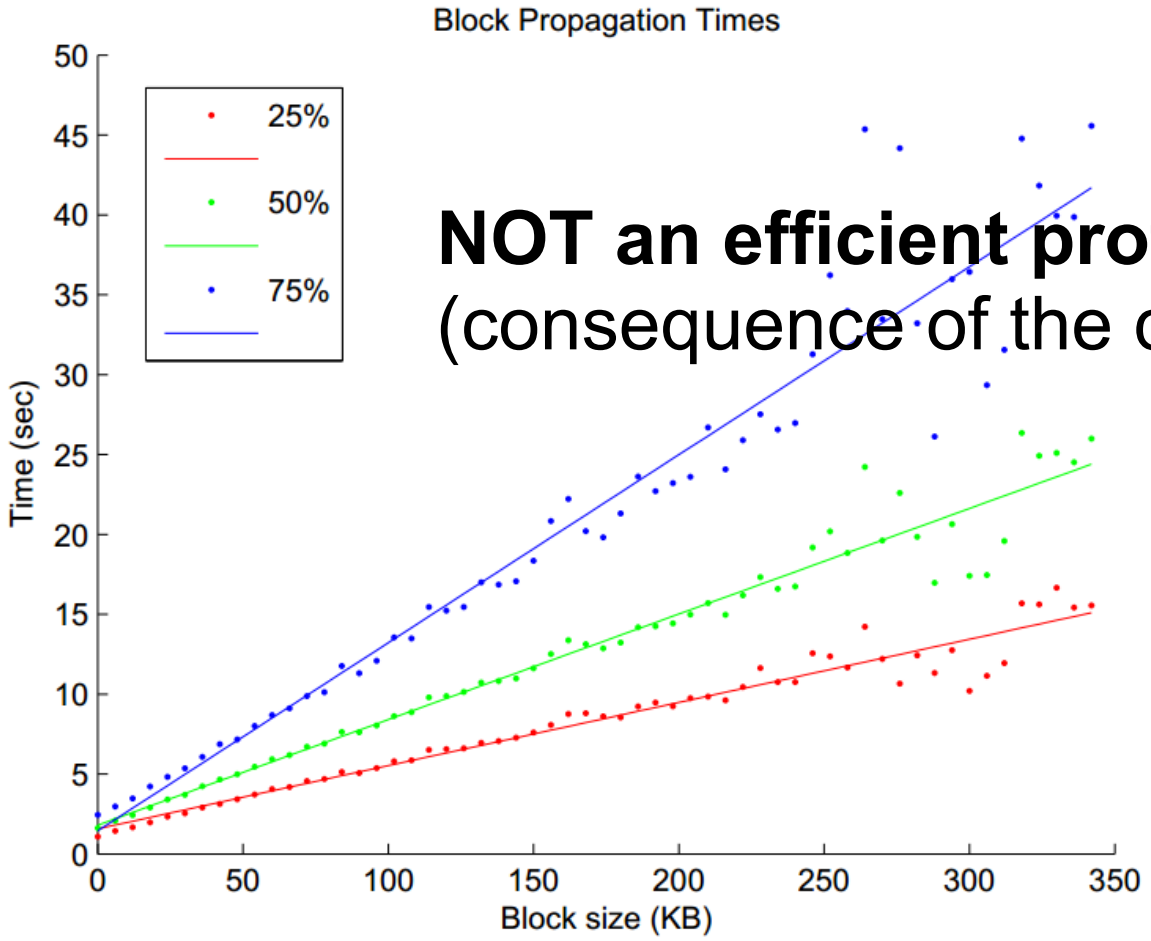
Relay a new block when you hear it if:

- Block meets the hash target
- Block has all valid transactions
  - Run *all* scripts, even if you wouldn't relay
- Block builds on current longest chain
  - Avoid forks





Source: Yonatan Sompolinsky and Aviv Zohar: “Accelerating Bitcoin’s Transaction Processing” 2014



NOT an efficient protocol
(consequence of the design)

Source: Yonatan Sompolinsky and Aviv Zohar: “Accelerating Bitcoin’s Transaction Processing” 2014

How big is the network?

- Unclear how to measure exactly
- Estimates-up to 1M IP addresses/month
- Only about 5-10k “full nodes”
    - Permanently connected
    - Fully-validate
- This number may be dropping!

Fully-validating nodes

- Permanently connected
- Store entire block chain
- Hear and forward every node/transaction

Thin/SPV clients (not fully-validating)

Idea: don’t store everything

- Store block headers only
- Request transactions as needed
  - To verify incoming payment
- Trust fully-validating nodes

Limitations & improvements

Hard-coded limits in Bitcoin

- 10 min. average creation time per block
- 1 M bytes in a block
- 20,000 signature operations per block
- 23M total bitcoins maximum
- 50,25,12.5... bitcoin mining reward

Hard-coded limits in Bitcoin

- 10 min. average creation time per block
- 1 M bytes in a block
- 20,000 signature operations per block
- 23M total bitcoins maximum
- 50,25,12.5... bitcoin mining reward

These affect economic balance of power too much to change now

Throughput limits in Bitcoin

- 1 M bytes/block (10 min)
- >250 bytes/transaction
- 7 transactions/sec ☹️

Compare to:

- VISA: 2,000-10,000 transactions/sec
- PayPal: 50-100 transaction/sec

Throughput limits in Bitcoin

- 1 M bytes/block (10 min)
- >250 bytes/transaction
- 7 tra

Improving throughput: A strong motivation for Altcoins

Compare

- VISA: 2,000-10,000 transactions/sec
- PayPal: 50-100 transaction/sec

Cryptographic limits in Bitcoin

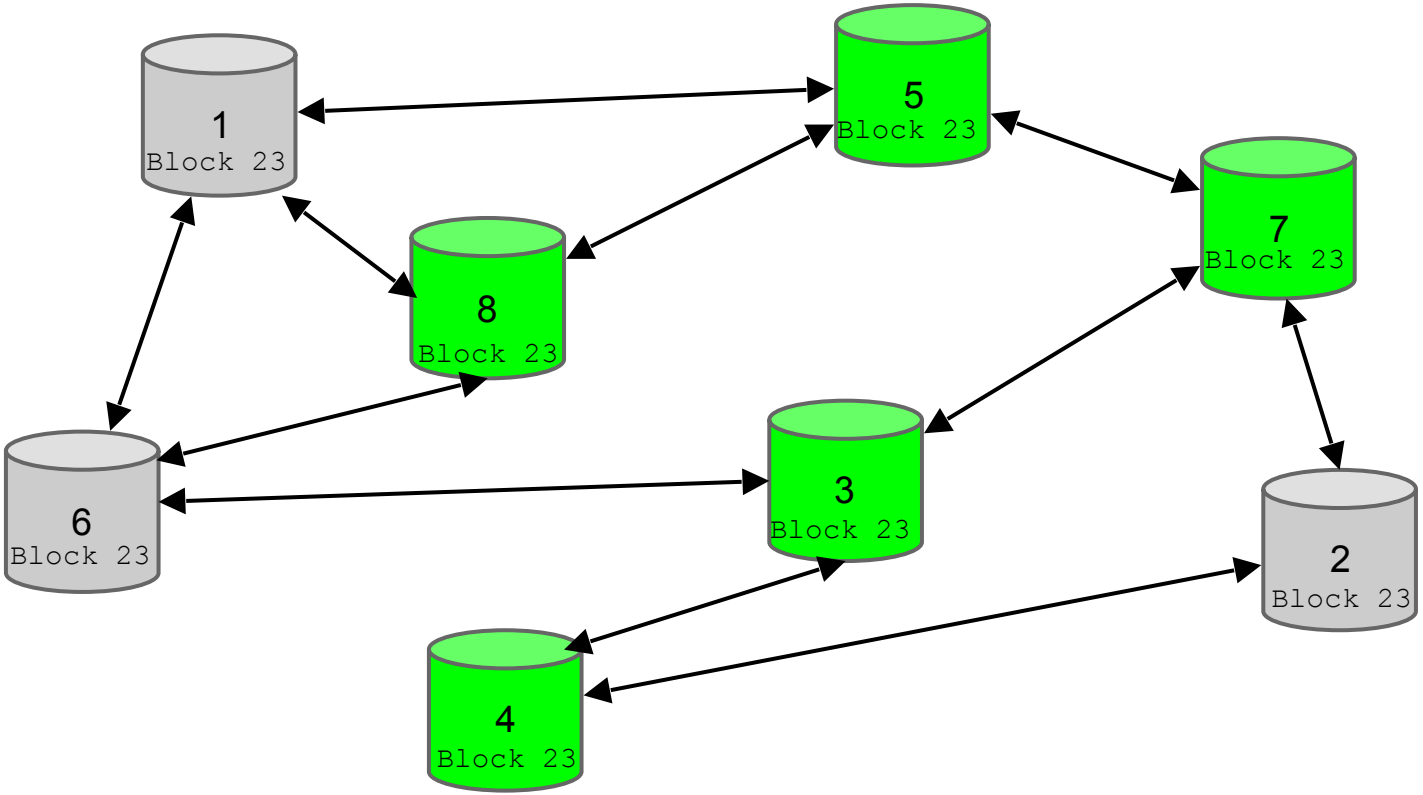
- Only 1 signature algorithm (ECDSA/P256)
- Hard-coded hash functions

Some of these crypto primitives used here might break by 2040 (e.g., collision-found in hash function, or powerful quantum computer breaks ECDSA)...

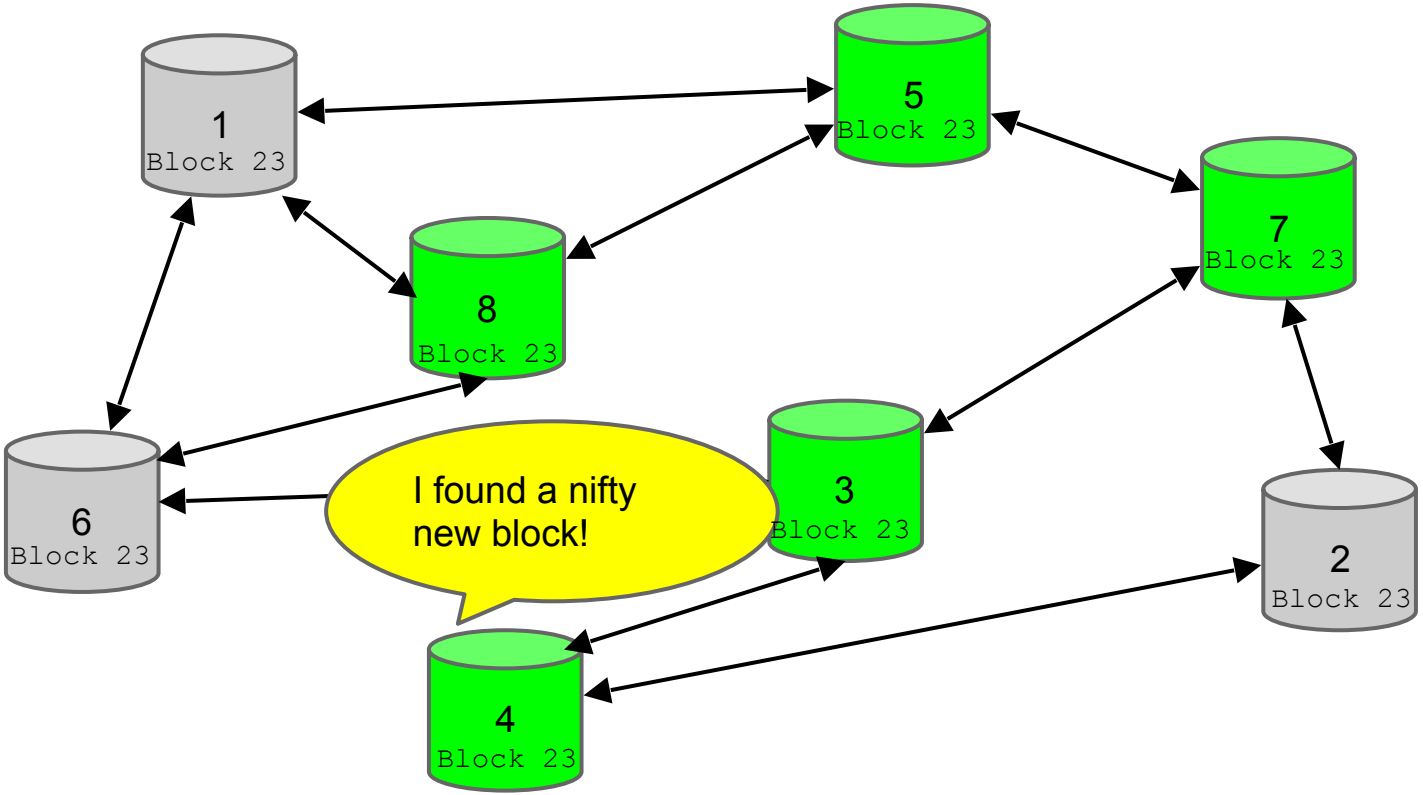
Why not update Bitcoin software to overcome these limitations?

- Many of these changes require “hard forks”, which are currently considered unacceptable

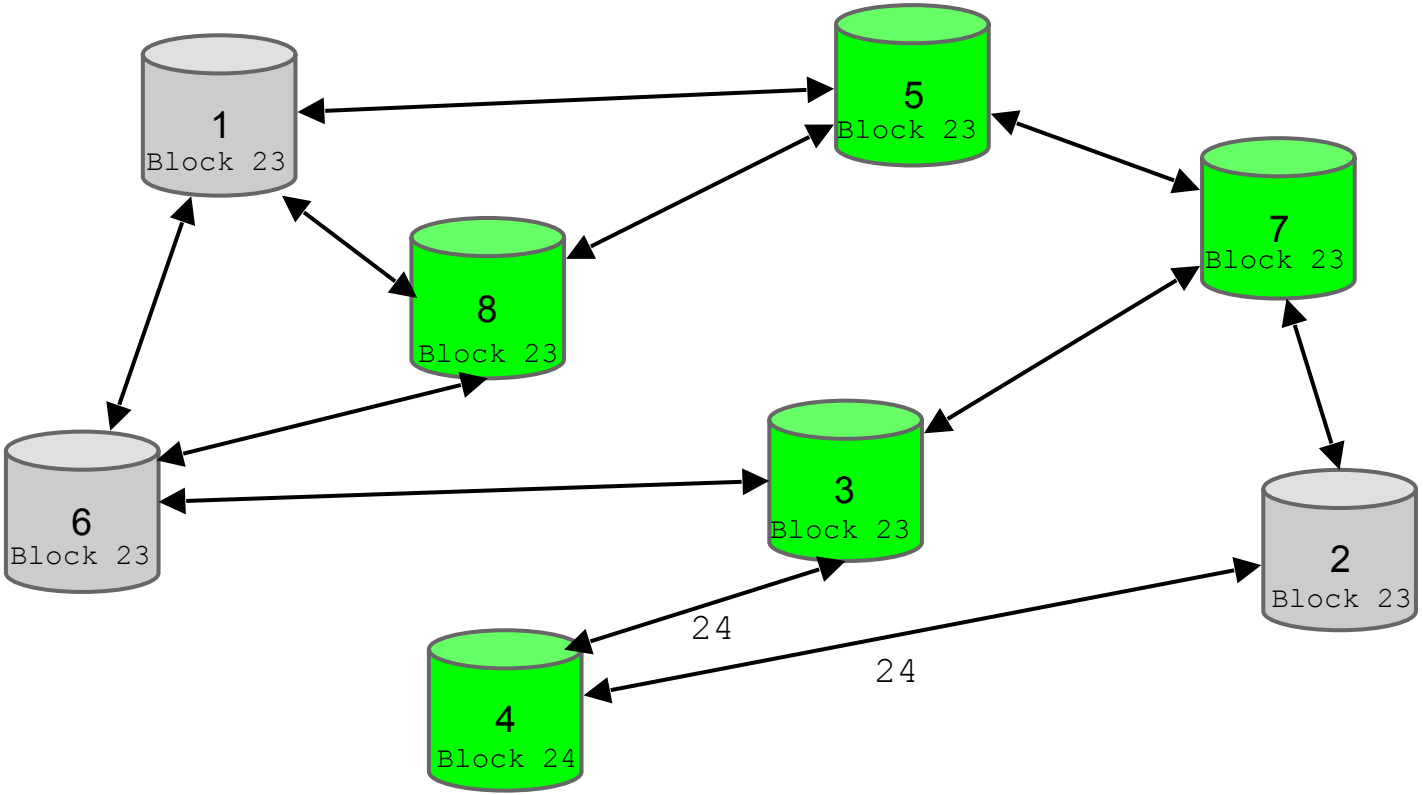
“Hard-forking” changes to Bitcoin



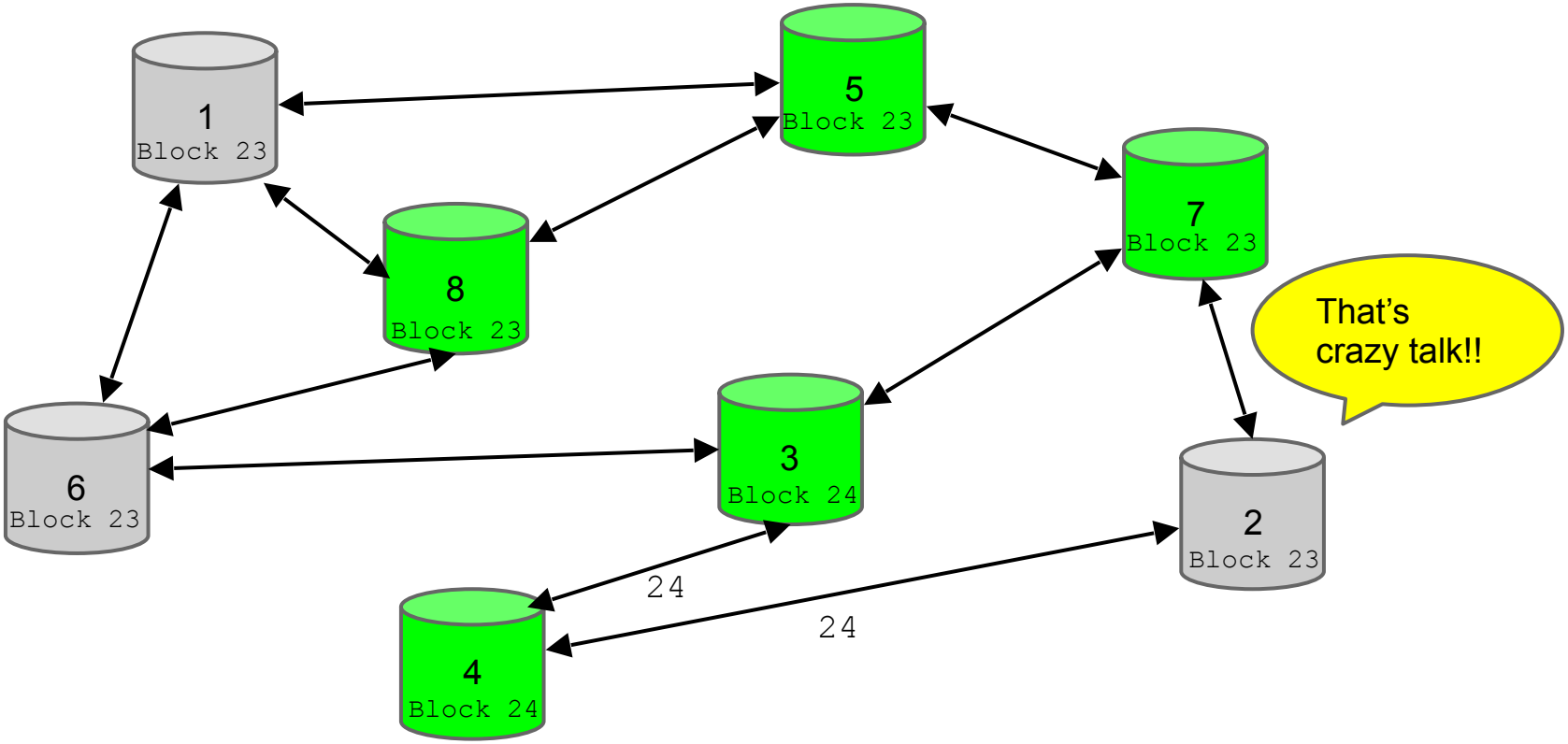
“Hard-forking” changes to Bitcoin



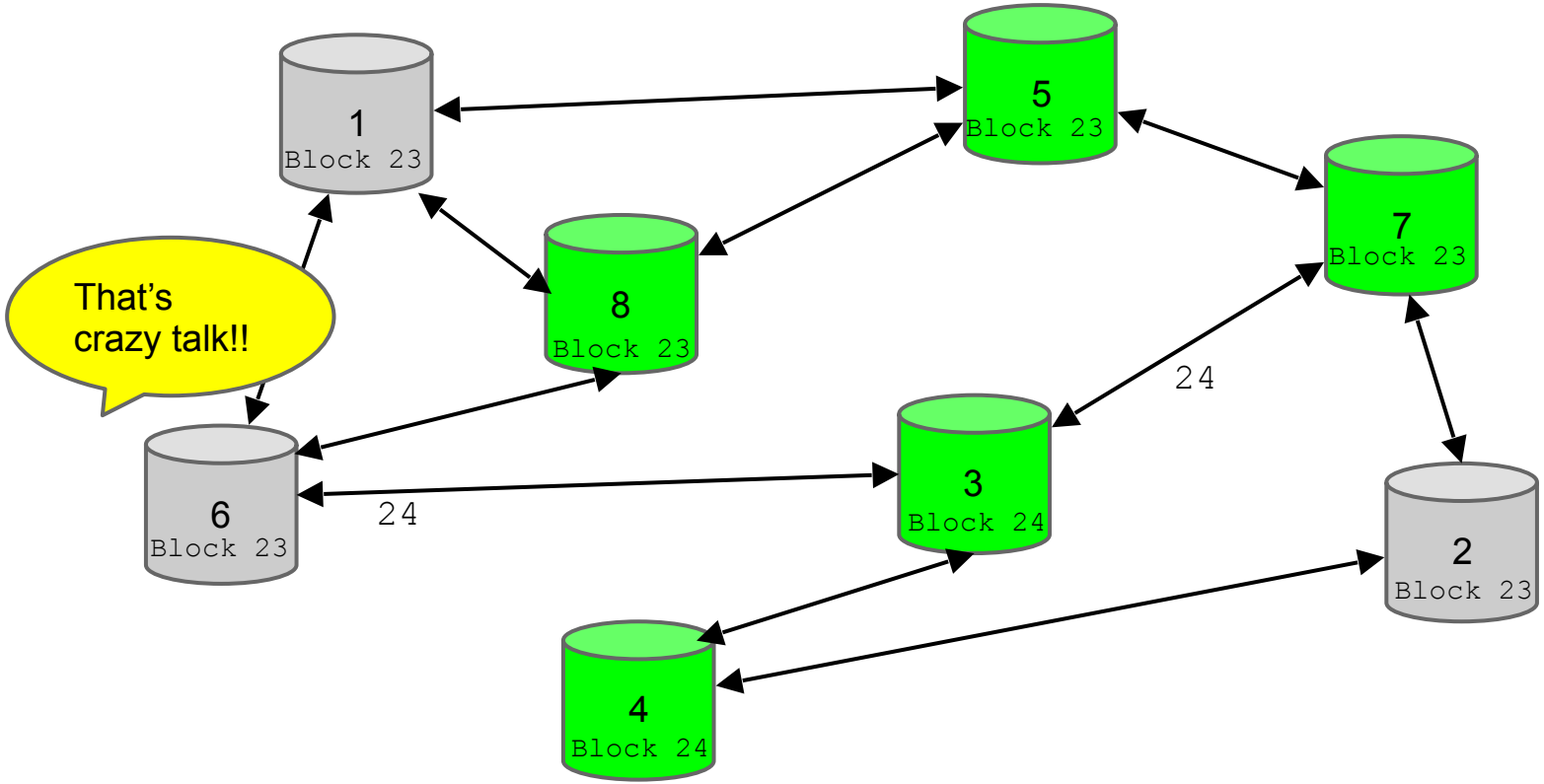
“Hard-forking” changes to Bitcoin



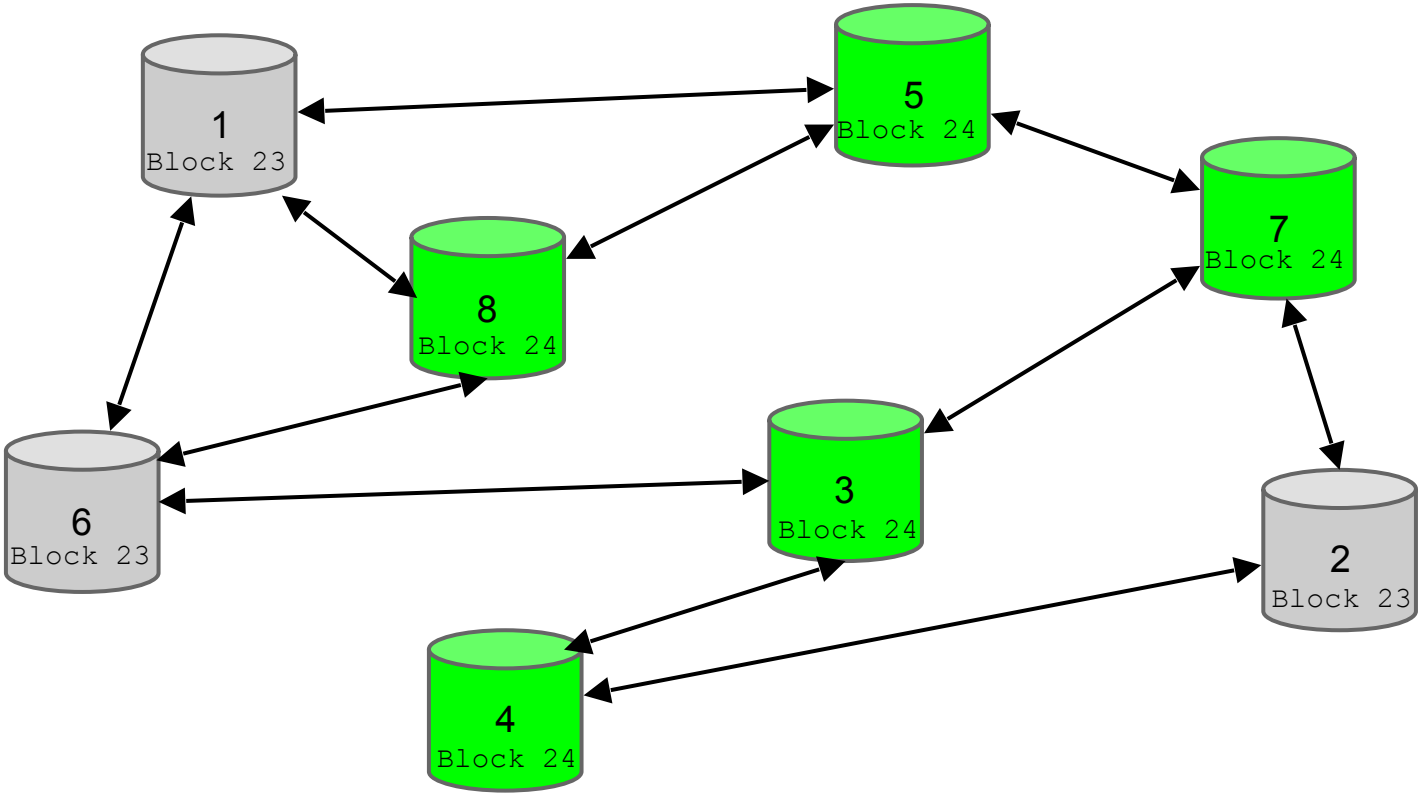
“Hard-forking” changes to Bitcoin



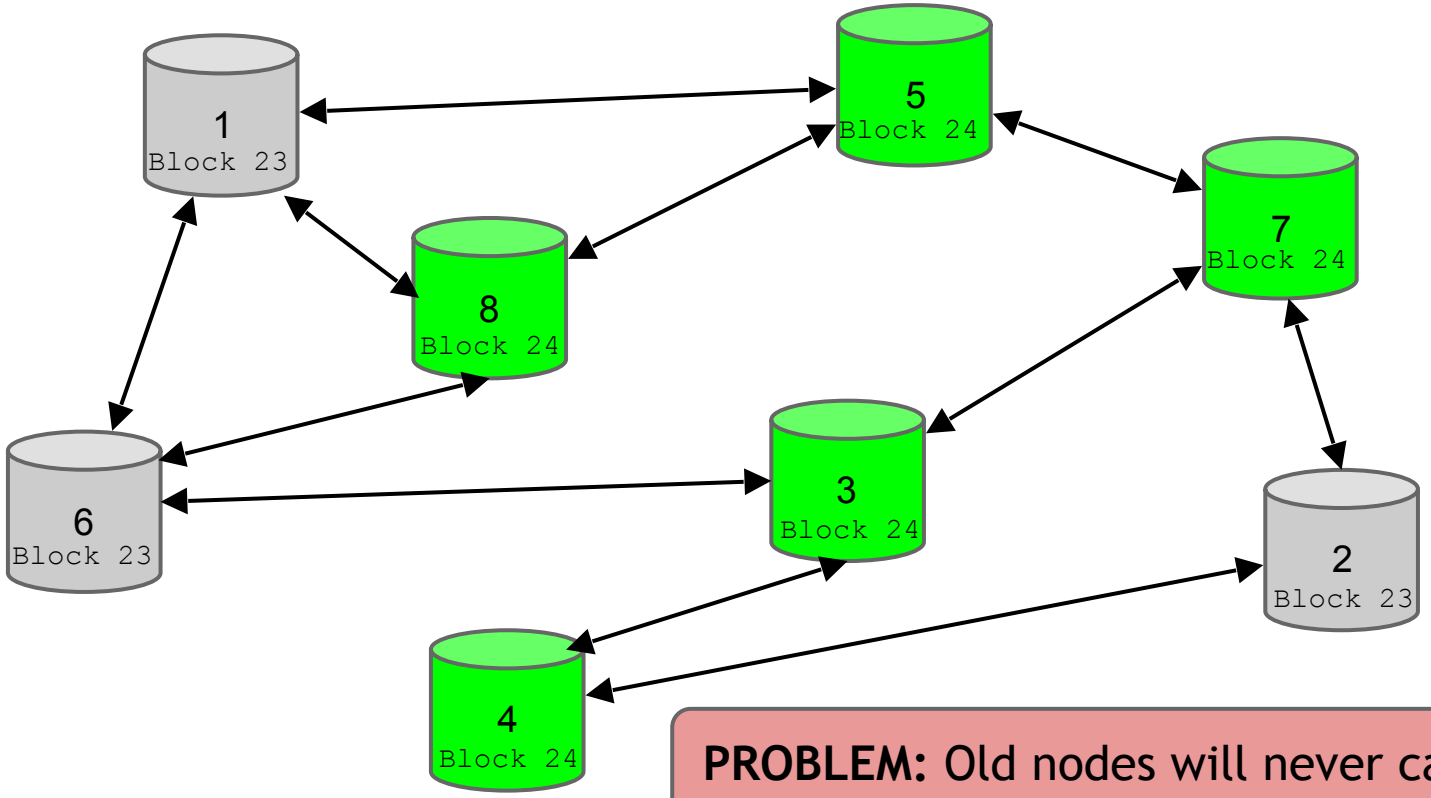
“Hard-forking” changes to Bitcoin



“Hard-forking” changes to Bitcoin



“Hard-forking” changes to Bitcoin



PROBLEM: Old nodes will never catch up

Soft forks

Observation: we can add new features which only *limit* the set of valid transactions

Need majority of nodes to enforce new rules

Old nodes will approve

Soft forks

Observation: we can add new features which only *limit* the set of valid transactions

Need majority of nodes to enforce new rules

Old nodes will approve

RISK: Old nodes might mine now-invalid blocks

Soft fork example: pay to script hash

<signature>
<<pubkey> OP_CHECKSIG>

OP_HASH160
<hash of redemption script>
OP_EQUAL

Old nodes will just approve the hash, not run the embedded script

Soft fork possibilities

- New signature schemes
- Extra per-block metadata
  - Shove in the coinbase parameter
  - Commit to unspent transaction tree in each block

Hard forks

- New op codes
- Changes to size limits
- Changes to mining rate
- Many small bug fixes

Hard forks

- New op codes
- Changes to size limits
- Changes to mining rate
- Many small bug fixes

Currently seem unlikely to happen

Hard forks

- New op codes
- Changes to size limits
- Changes to mining rate
- Many small bug fixes

Currently seem unlikely to happen

Many of these issues addressed by Altcoins