Blockchains & Cryptocurrencies

Bitcoin Mechanics - II

Instructor: Abhishek Jain
Johns Hopkins University - Spring 2021

*Many slides based on NBFMG

Last Time: Errata

Last Time: Errata

- Miners indeed store a set of **unspent transactions** (UTXO) to check for double-spending

Last Time: Errata

- Miners indeed store a set of **unspent transactions** (UTXO) to check for double-spending

- Can potentially be kept in RAM

Today

Along the way, start identifying directions for improvements (or, motivation for altcoins)

Today

- Bitcoin Script Applications

Along the way, start identifying directions for improvements (or, motivation for altcoins)

Today

- Bitcoin Script Applications
- Bitcoin Network

Along the way, start identifying directions for improvements (or, motivation for altcoins)

Today

- Bitcoin Script Applications
- Bitcoin Network
- Soft/Hard forks

Along the way, start identifying directions for improvements (or, motivation for altcoins)

Today

- Bitcoin Script Applications
- Bitcoin Network
- Soft/Hard forks
- Mining (maybe…)

Along the way, start identifying directions for improvements (or, motivation for altcoins)

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

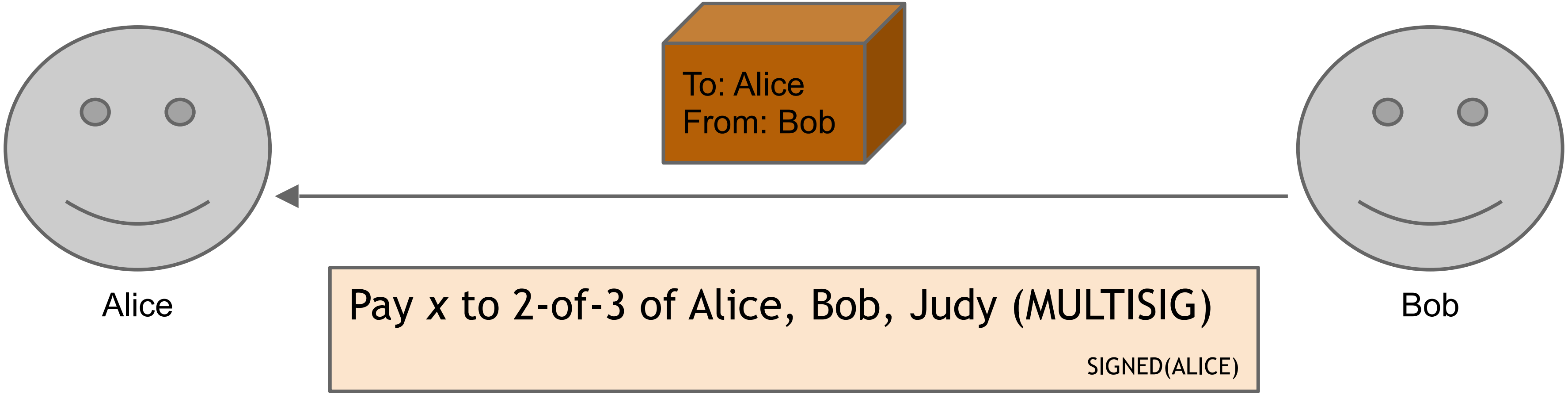
Alice

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

SIGNED(ALICE)

Bob

Example 1: Fair transactions via Escrow

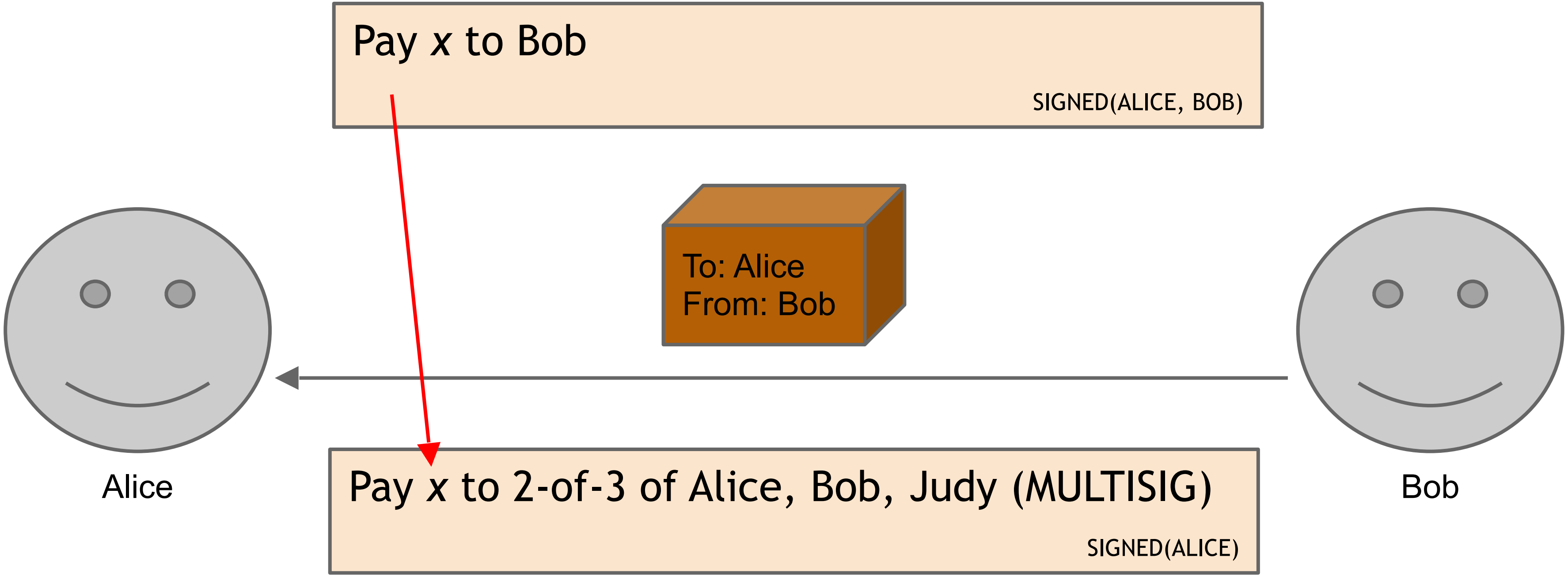


Pay *x* to 2-of-3 of Alice, Bob, Judy (MULTISIG)

SIGNED(ALICE)

Example 1: Fair transactions via Escrow

(normal case)



Pay *x* to Bob

SIGNED(ALICE, BOB)

To: Alice
From: Bob

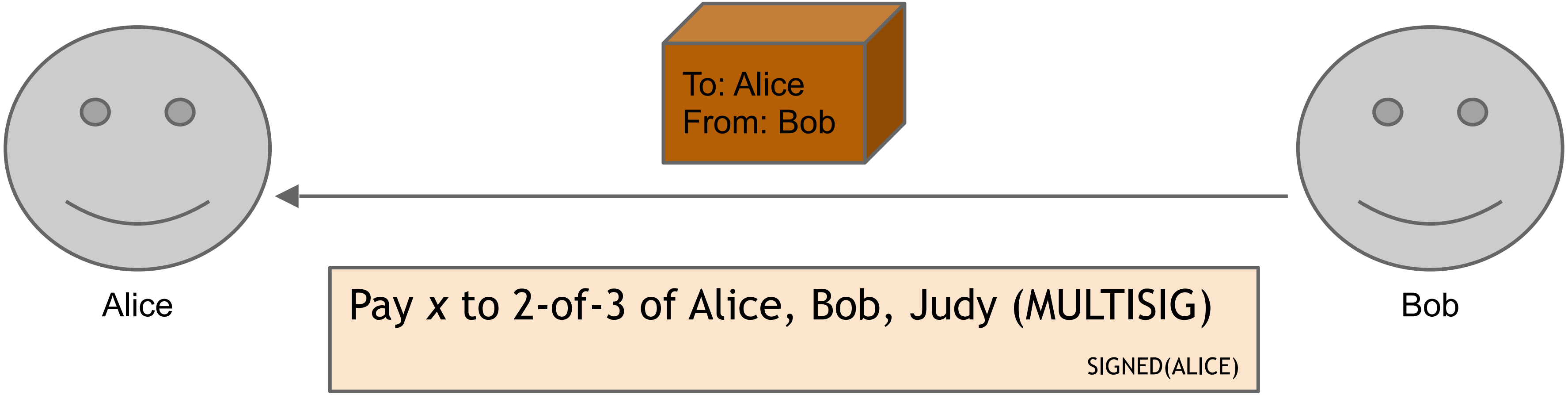
Alice

Bob

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

SIGNED(ALICE)

Example 1: Fair transactions via Escrow



To: Alice
From: Bob

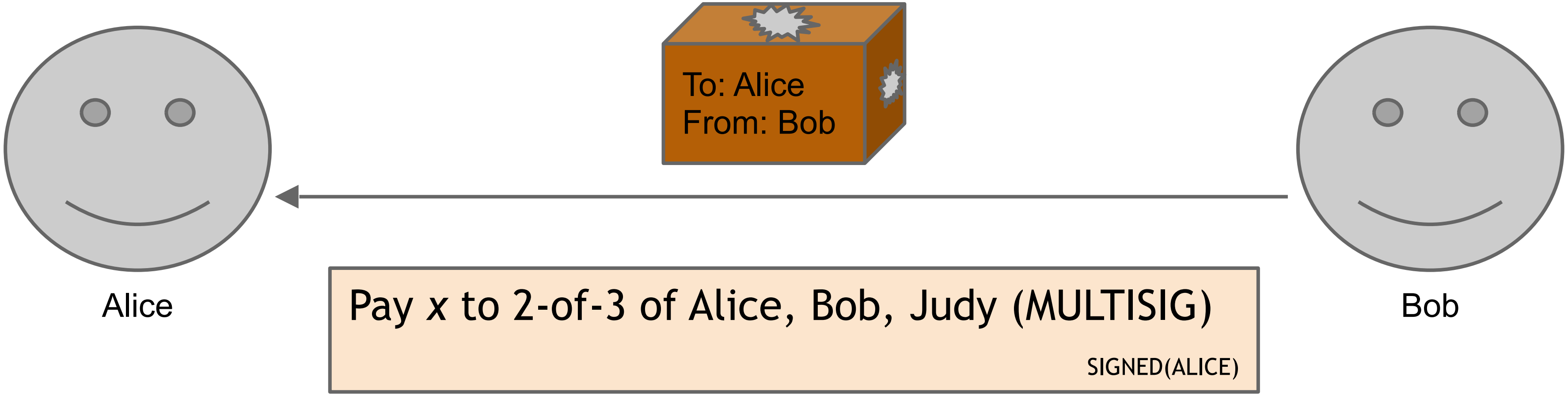
Alice

Bob

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

SIGNED(ALICE)

Example 1: Fair transactions via Escrow

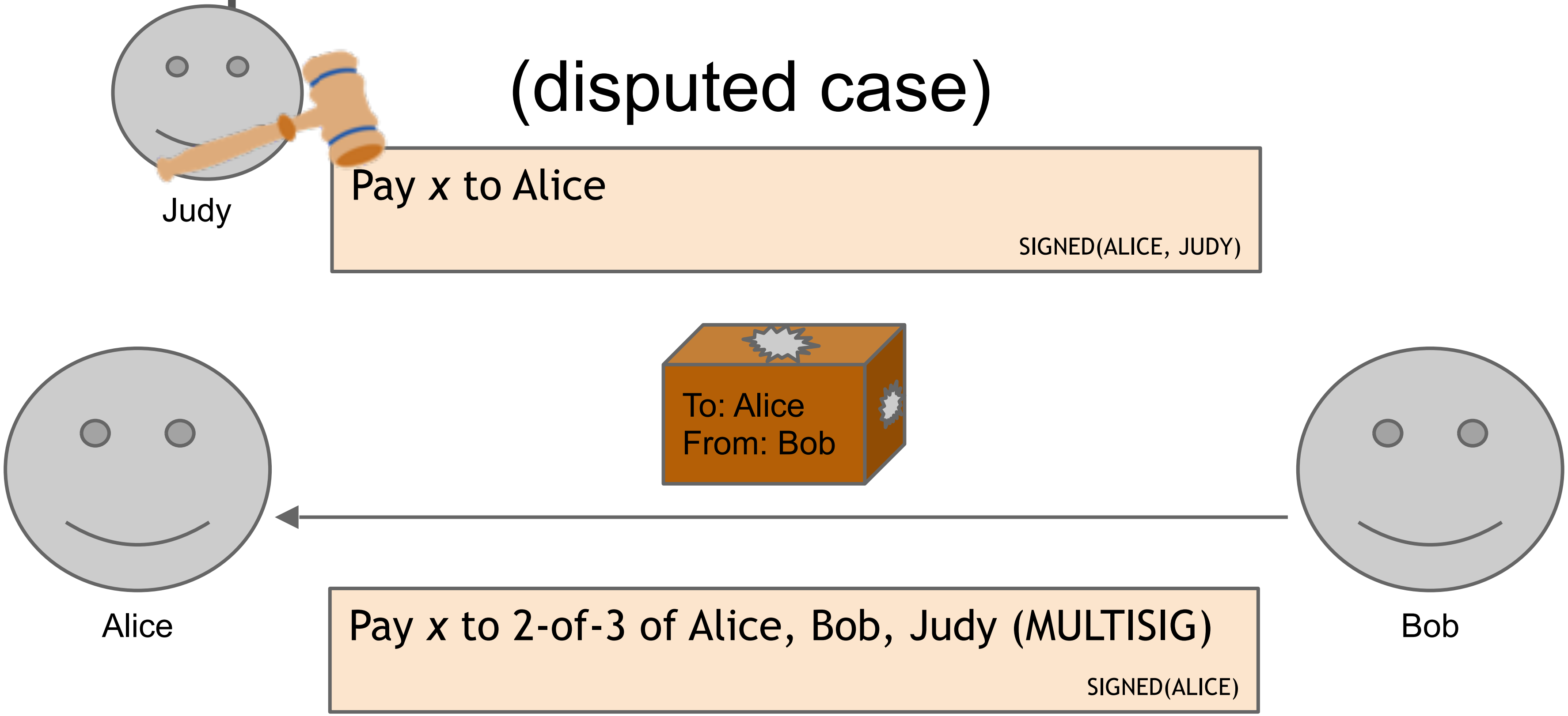


Pay *x* to 2-of-3 of Alice, Bob, Judy (MULTISIG)

SIGNED(ALICE)

Example 1: Fair transactions via Escrow

(disputed case)



Pay *x* to Alice

SIGNED(ALICE, JUDY)

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 3: Micro-payments with Bitcoin

Example 3: Micro-payments with Bitcoin

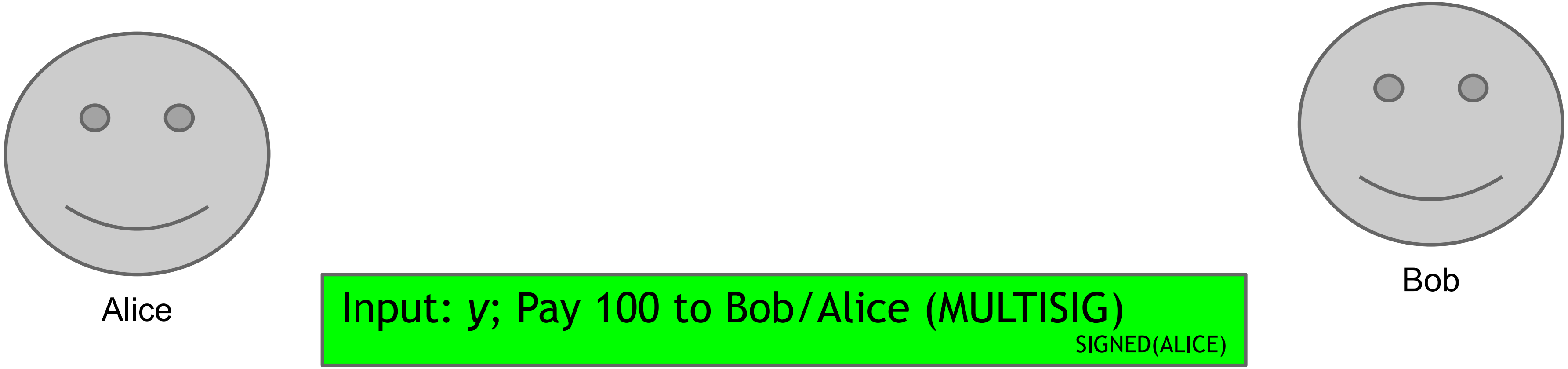
- Main Idea: Instead of doing several transactions, do a single transaction for total payment (and thus incur only a single transaction fee)
- *How to implement it?*

Example 3: Micro-payments with Bitcoin

Alice

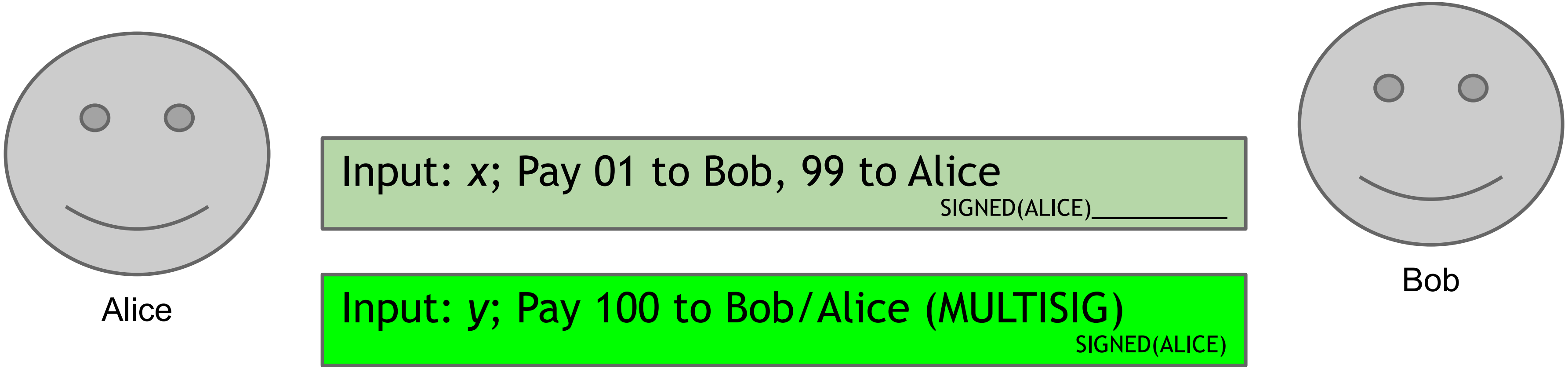
Bob

Example 3: Micro-payments with Bitcoin



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

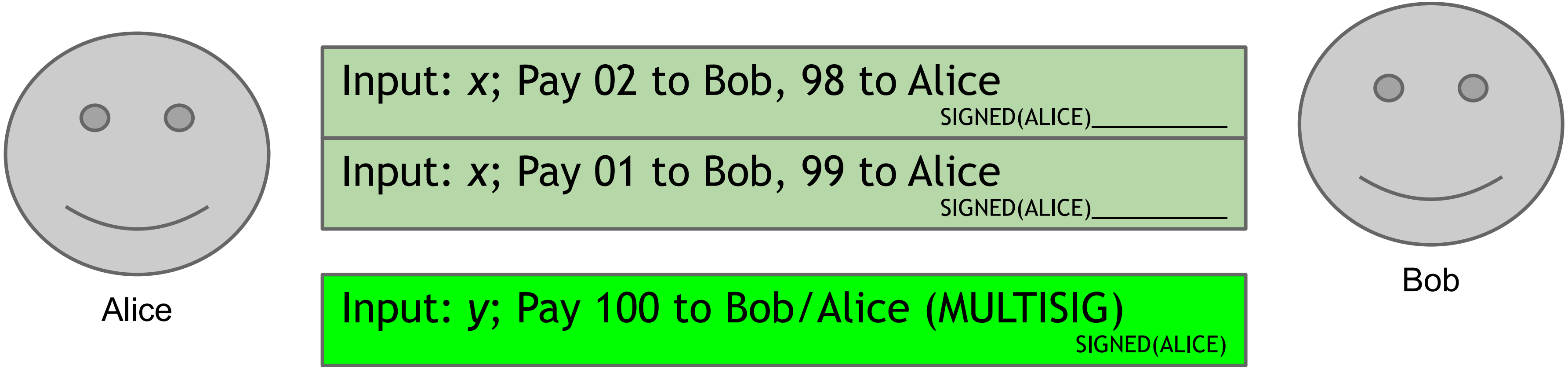
Example 3: Micro-payments with Bitcoin



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

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

Example 3: Micro-payments with Bitcoin

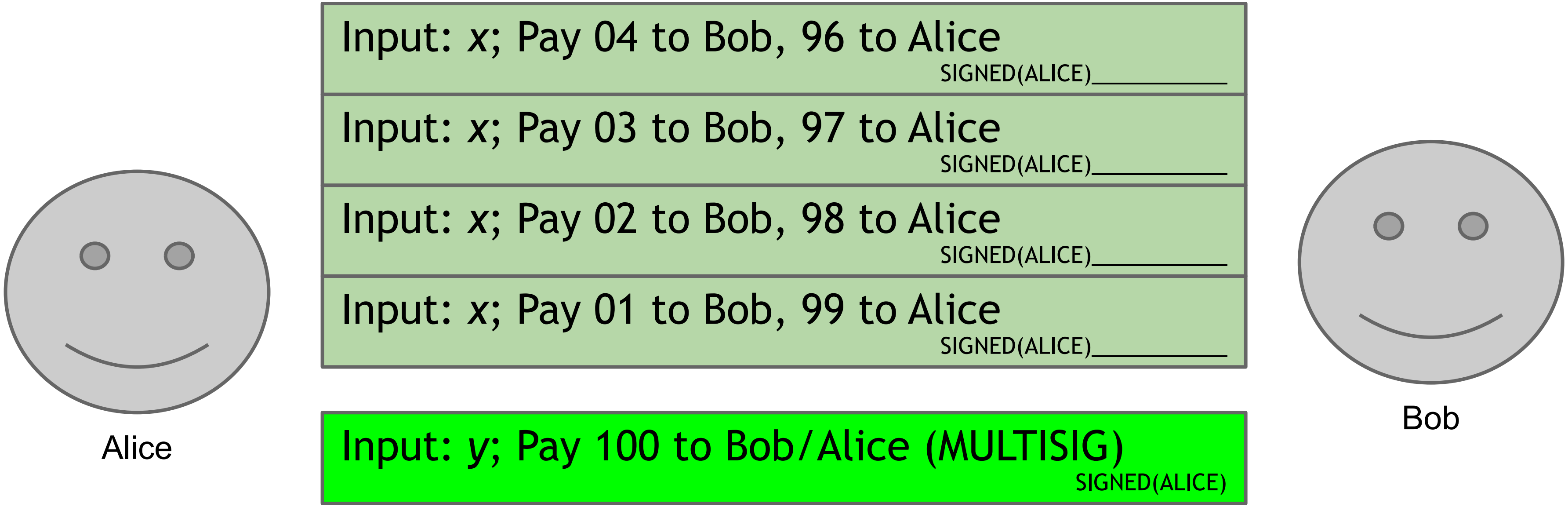


Input: *x*; Pay 02 to Bob, 98 to Alice
SIGNED(ALICE)___________

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

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

Example 3: Micro-payments with Bitcoin



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

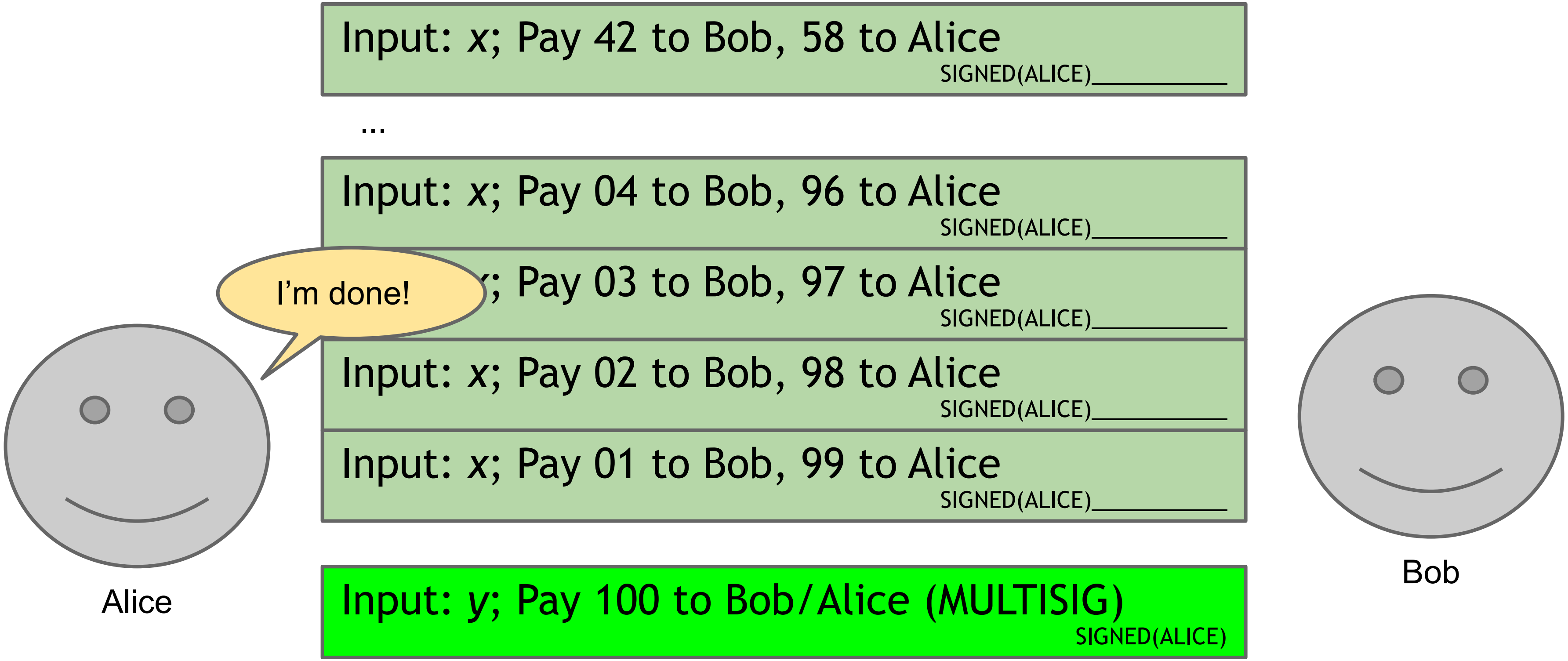
Input: *x*; Pay 03 to Bob, 97 to Alice
SIGNED(ALICE)___________

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)

Example 3: Micro-payments with Bitcoin



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

...

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

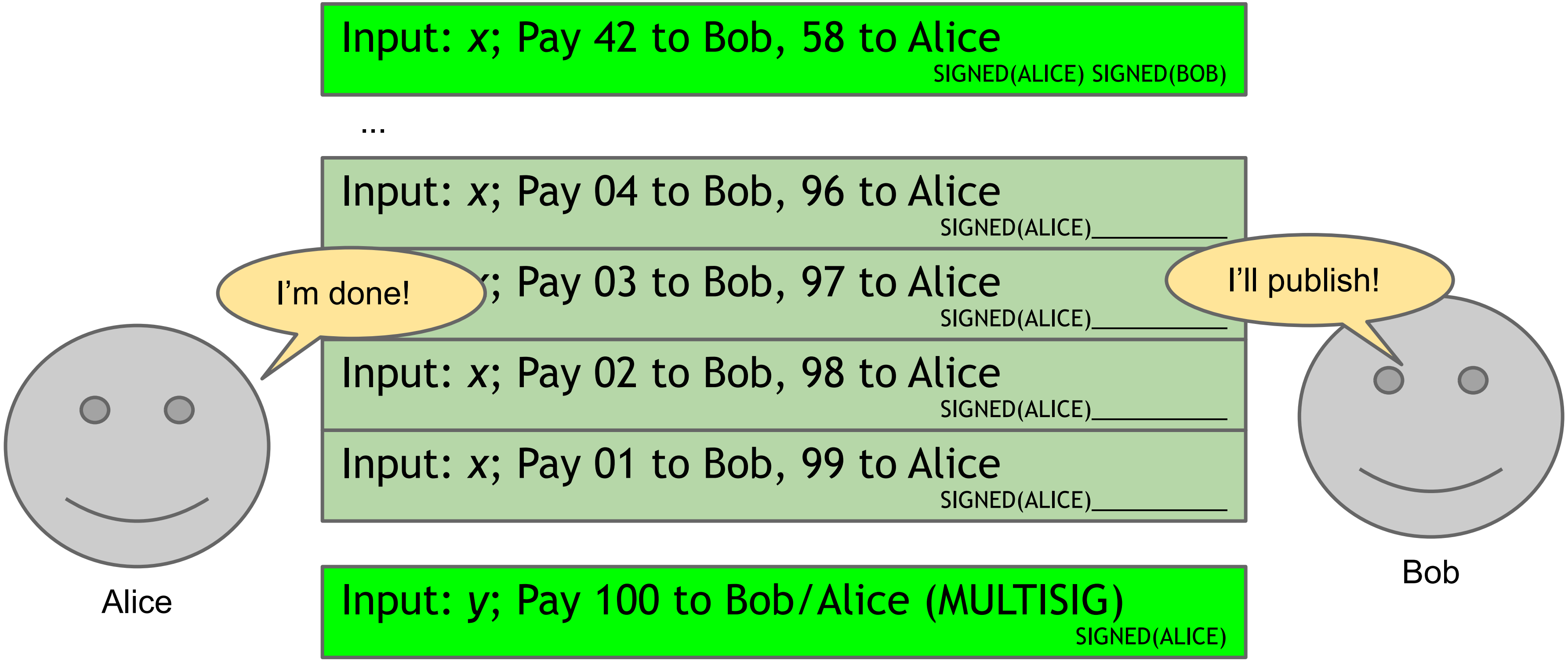
; Pay 03 to Bob, 97 to Alice
SIGNED(ALICE)___________

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)

Example 3: Micro-payments with Bitcoin



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

...

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

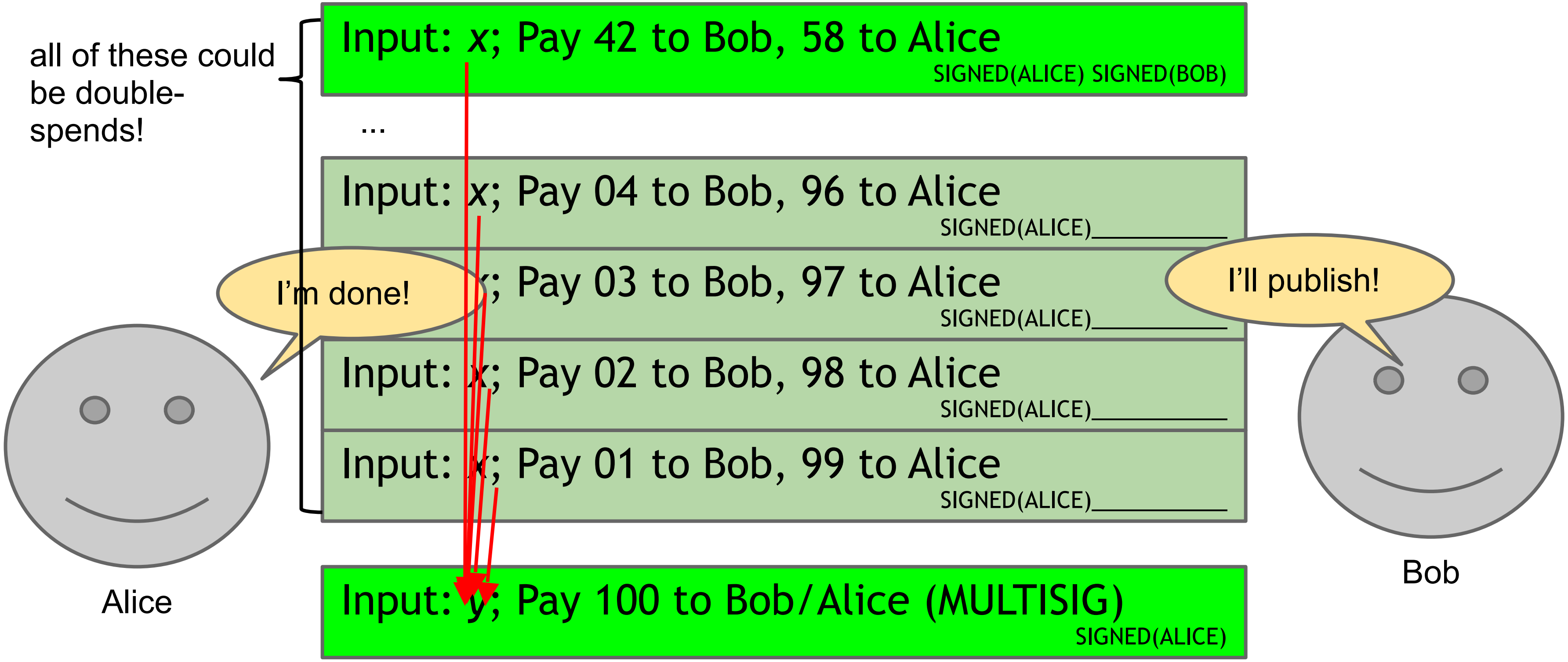
; Pay 03 to Bob, 97 to Alice
SIGNED(ALICE)__________

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)

Example 3: Micro-payments with Bitcoin



all of these could be double-spends!

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

...

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

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)

Example 3: 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 3: 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 3: 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

Some 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)

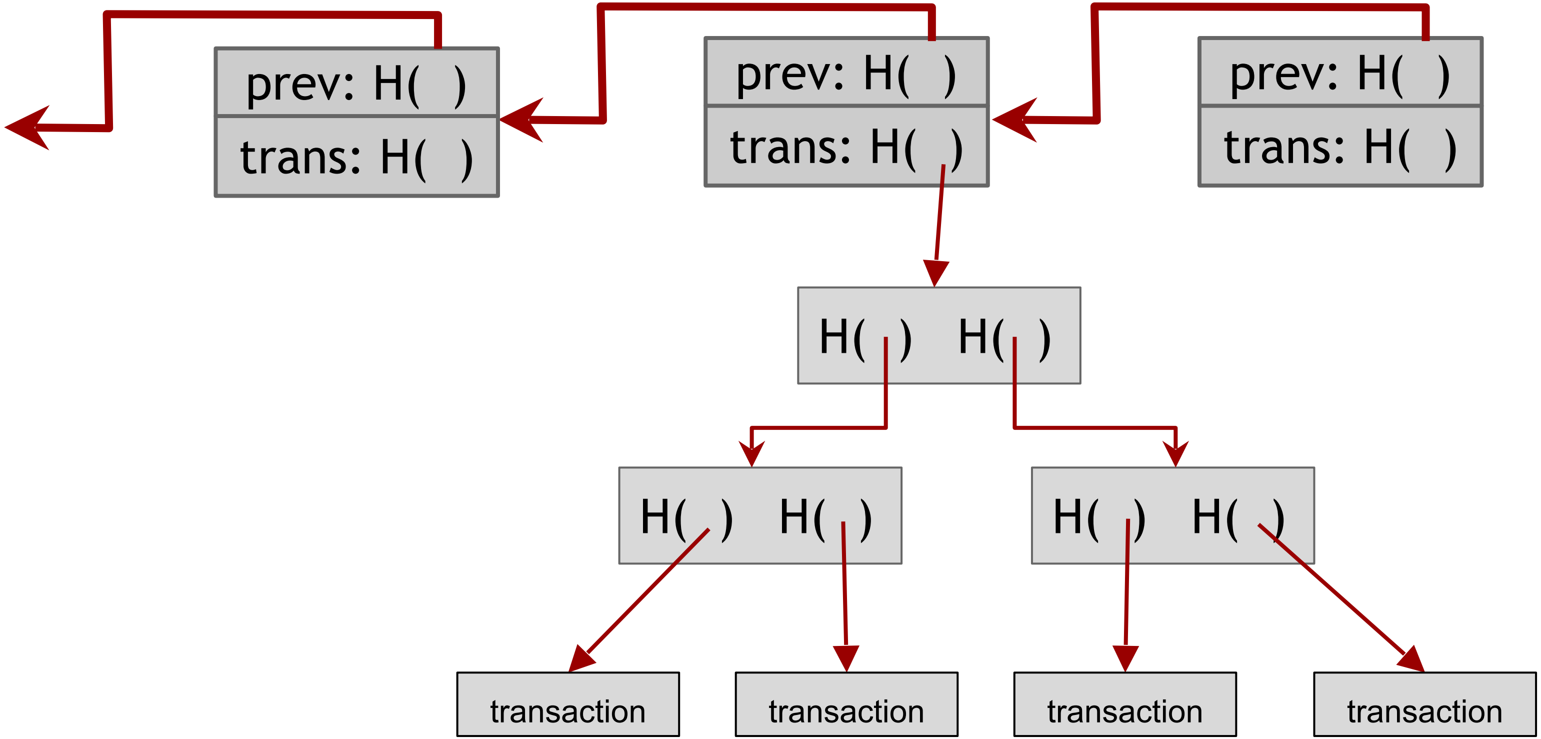
Bitcoin blocks

Bitcoin blocks

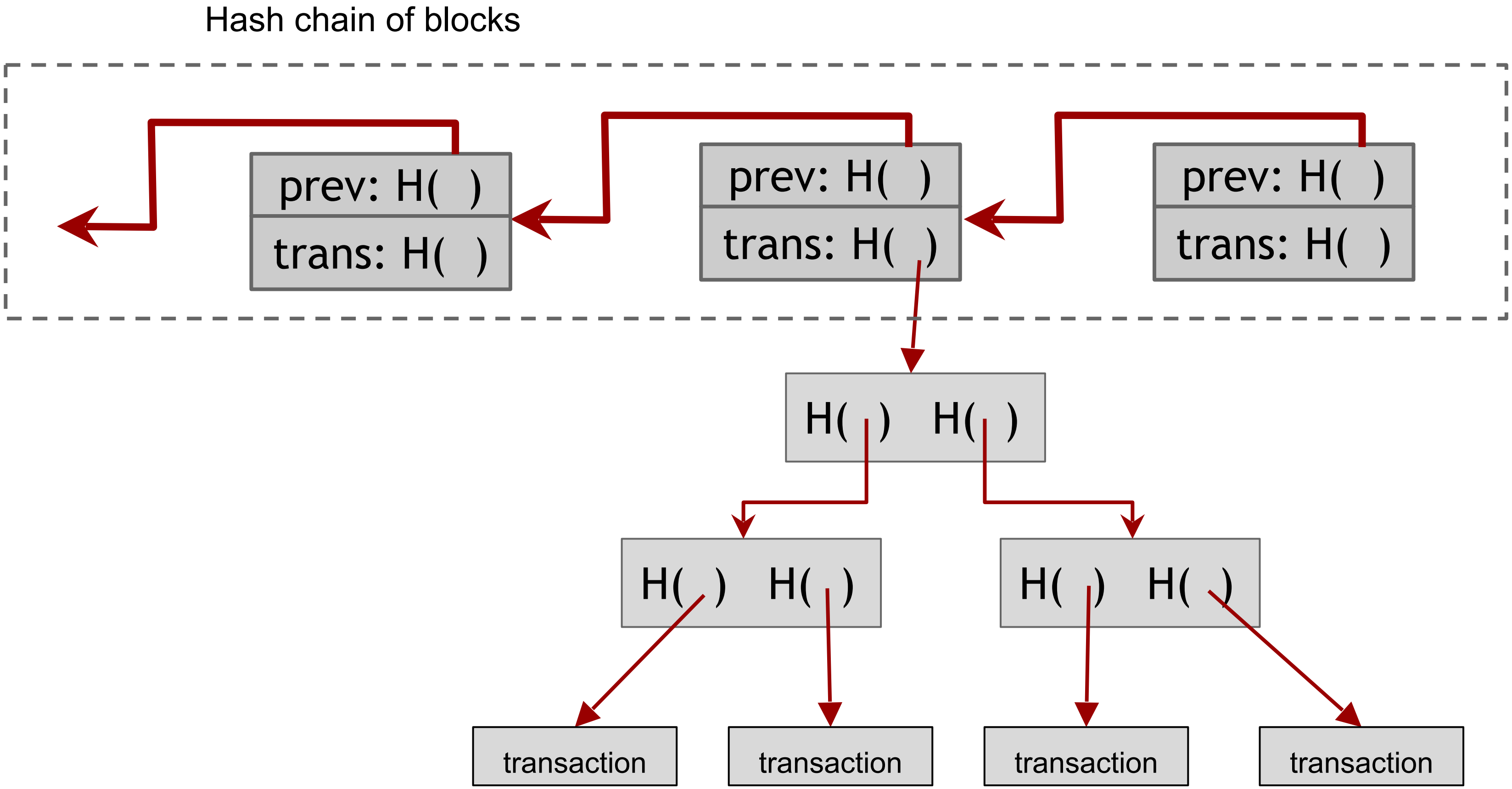
Why bundle transactions together?

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

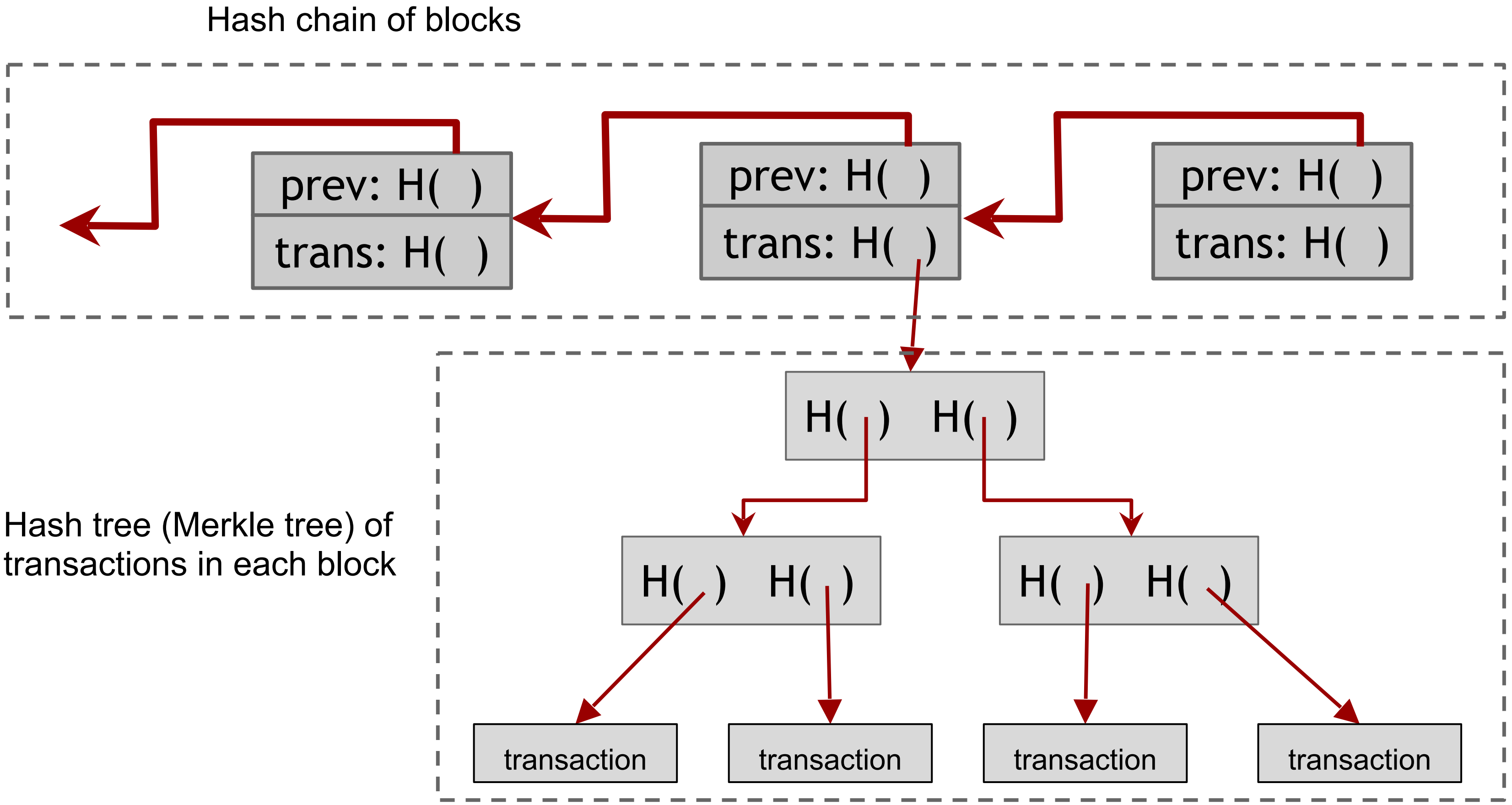
Bitcoin block structure



Bitcoin block structure



Bitcoin block structure



The real deal: a classical Bitcoin block

block header

```
{
  "hash":"00000000000000001aad2...",
  "ver":2,
  "prev_block":"00000000000000003043...",
  "time":1391279636,
  "bits":419558700,
  "nonce":459459841,
  "mrkl_root":"89776...",
```

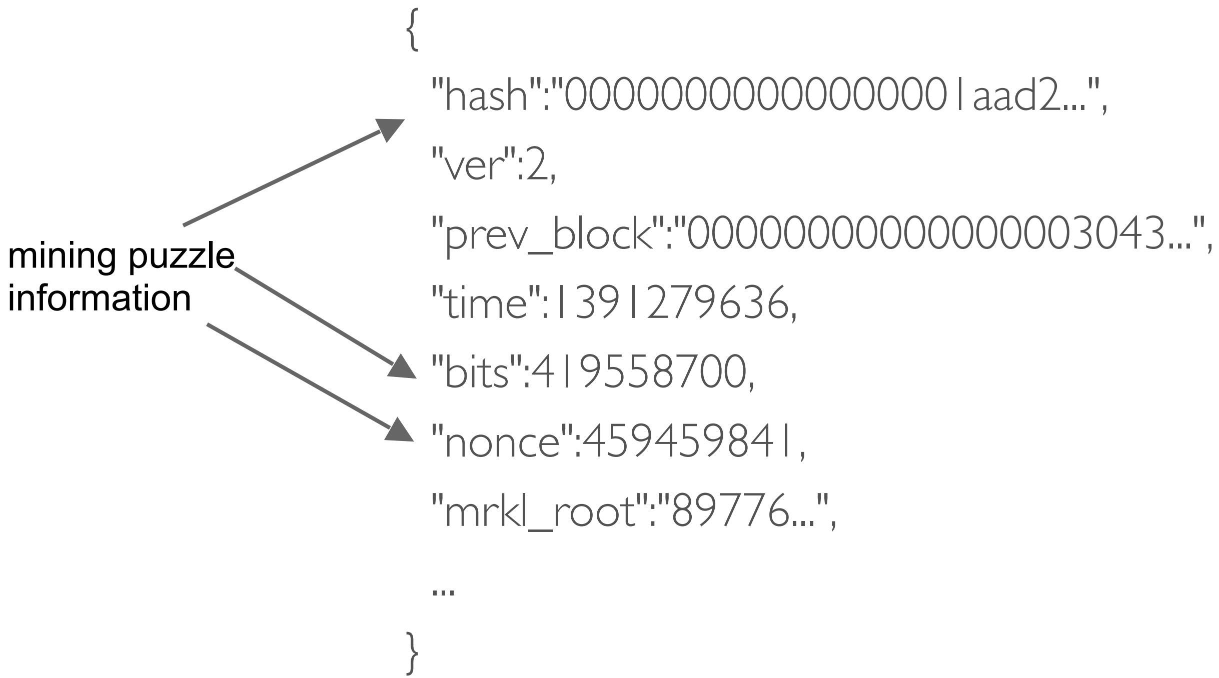
transaction data

```
  "n_tx":354,
  "size":181520,
  "tx":[
    ...
  ],
  "mrkl_tree":[
    "6bd5eb25...",
    ...
    "89776cdb..."
  ]
}
```

The real deal: block header

```
{
  "hash":"00000000000000001aad2...",
  "ver":2,
  "prev_block":"00000000000000003043...",
  "time":1391279636,
  "bits":419558700,
  "nonce":459459841,
  "mrkl_root":"89776...",
  ...
}
```

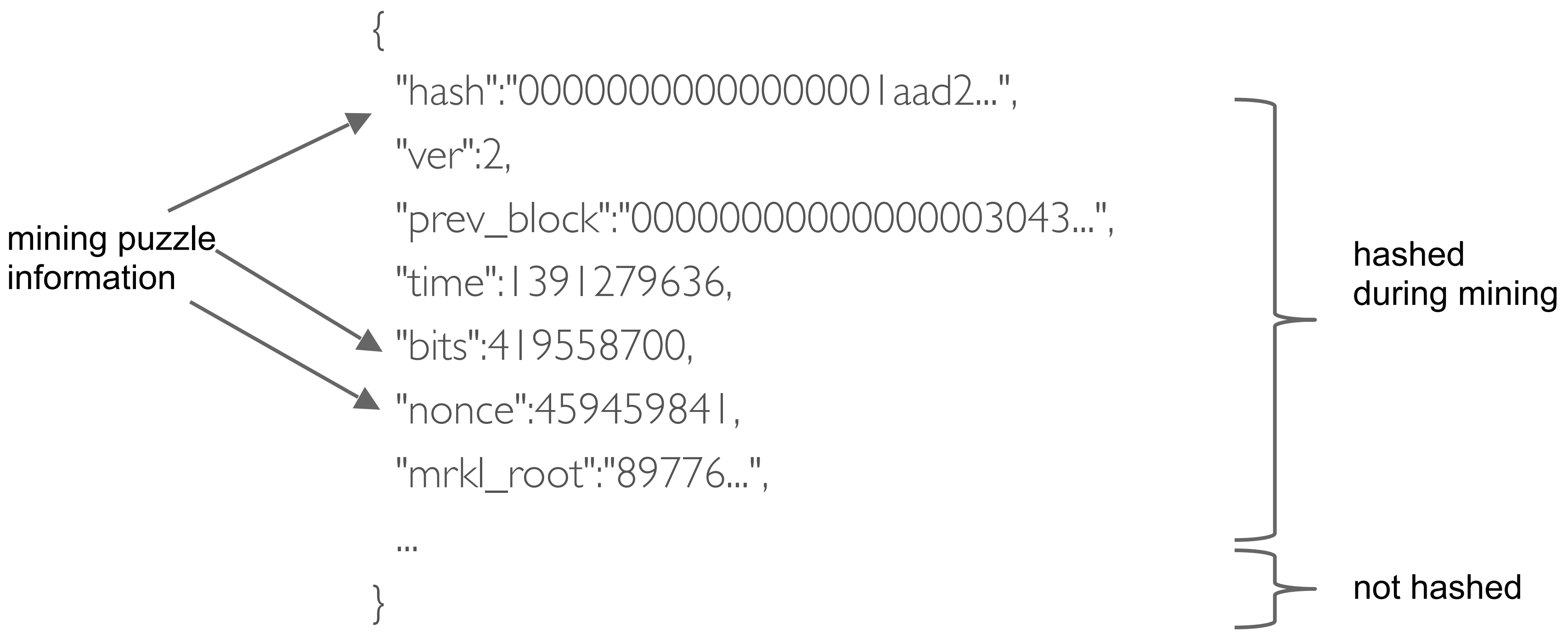
The real deal: block header



mining puzzle information

```
{
  "hash":"00000000000000001aad2...",
  "ver":2,
  "prev_block":"00000000000000003043...",
  "time":1391279636,
  "bits":419558700,
  "nonce":459459841,
  "mrkl_root":"89776...",
  ...
}
```

The real deal: block header



```
{
  "hash":"0000000000000000 1aad2...",
  "ver":2,
  "prev_block":"0000000000000000003043...",
  "time":1391279636,
  "bits":419558700,
  "nonce":459459841,
  "mrkl_root":"89776...",
  ...
}
```

mining puzzle information

hashed during mining

not hashed

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

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

redeeming nothing

arbitrary

block reward

transaction fees

The real deal: coinbase transaction

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

Null hash pointer

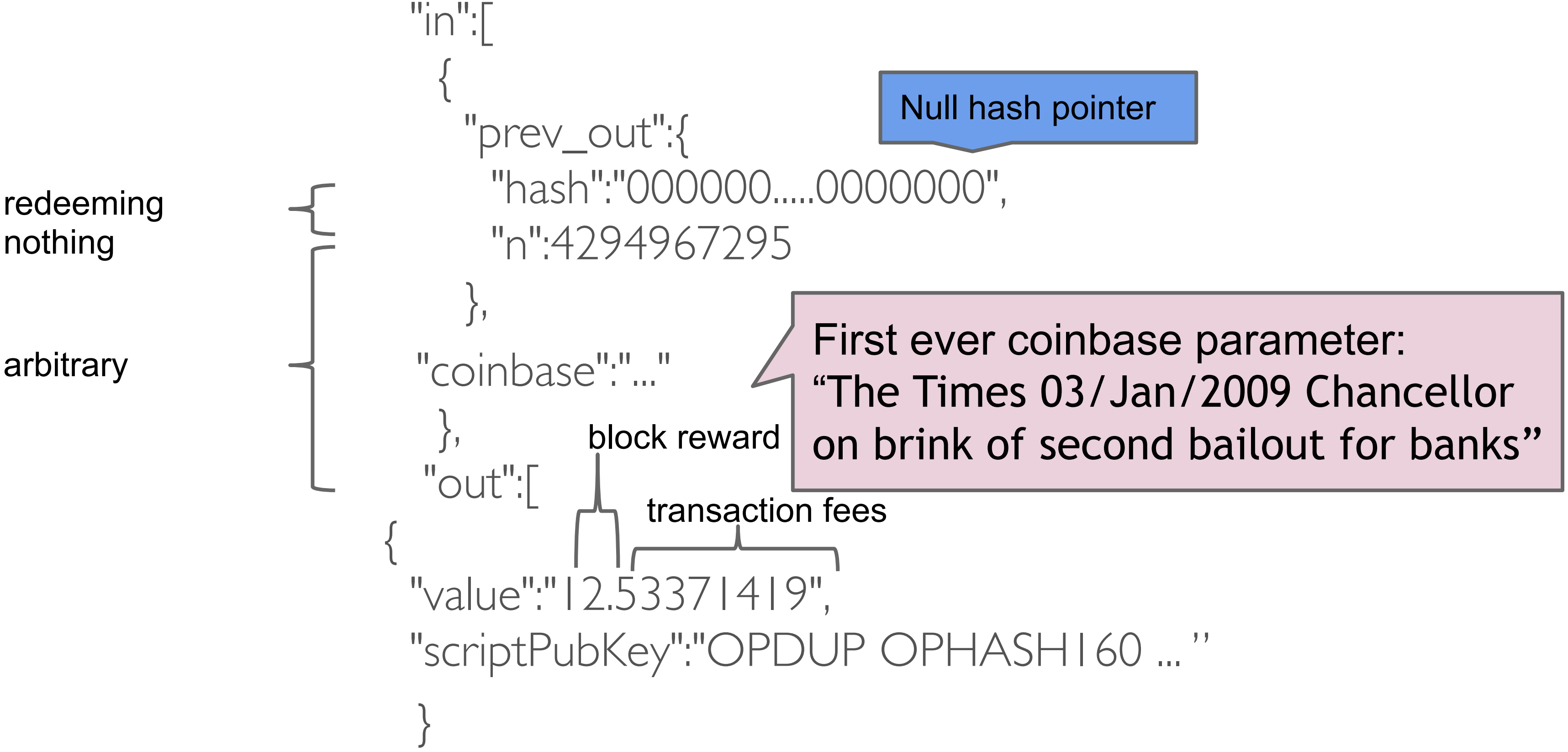
redeeming nothing

arbitrary

block reward

transaction fees

The real deal: coinbase transaction

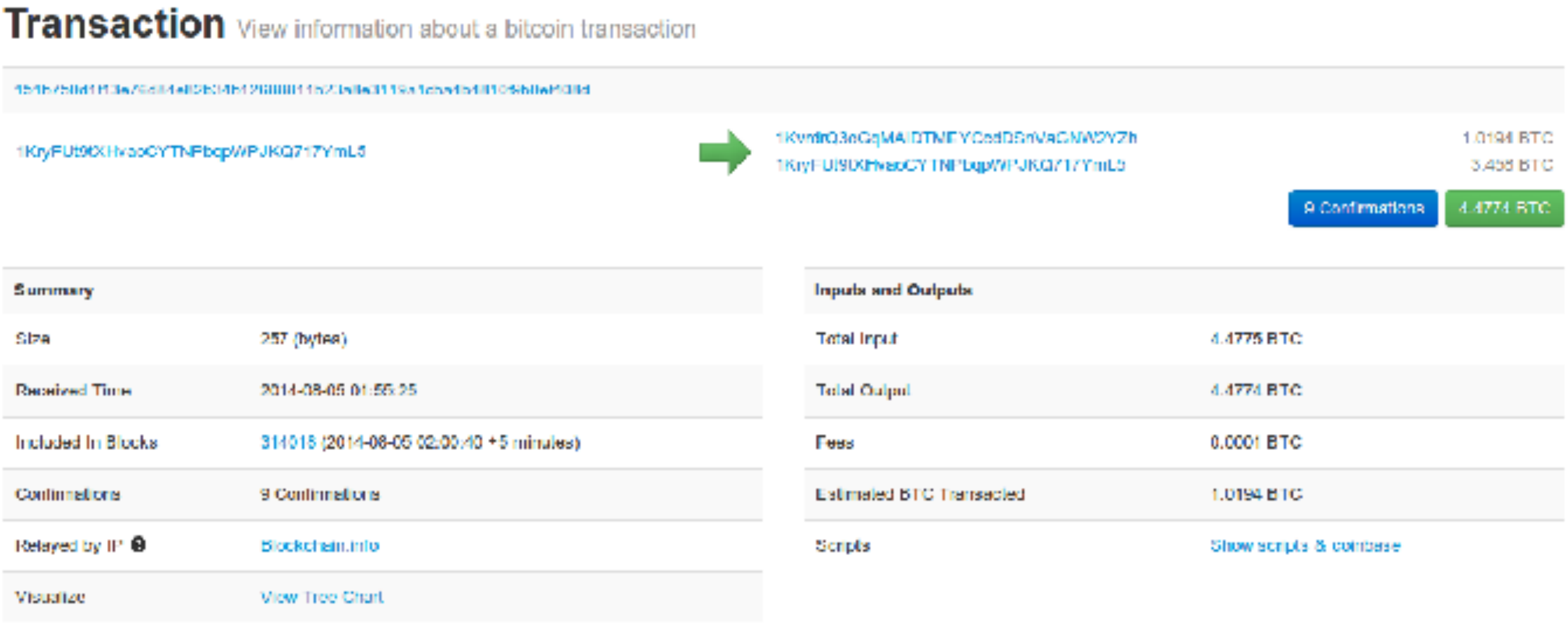


```
"in":[
  {
    "prev_out":{
      "hash":"000000.....0000000",
      "n":4294967295
    },
 "coinbase":"..."
  },
 "out":[
{
 "value":"12.53371419",
 "scriptPubKey":"OPDUP OPHASH160 ... ’’
 }
```

Null hash pointer

First ever coinbase parameter:
“The Times 03/Jan/2009 Chancellor on brink of second bailout for banks”

See for yourself!



Transaction View information about a bitcoin transaction

[illegible]

1KryFUt9tXHvaoCYTNFbcpWPJKQ717YmL5 → 1KvmfrQ3oCqMAIDTNFYCedDSnVaGNW2YZh 1.0194 BTC
1KryFUt9tXHvaoCYTNFbcpWPJKQ717YmL5 3.458 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

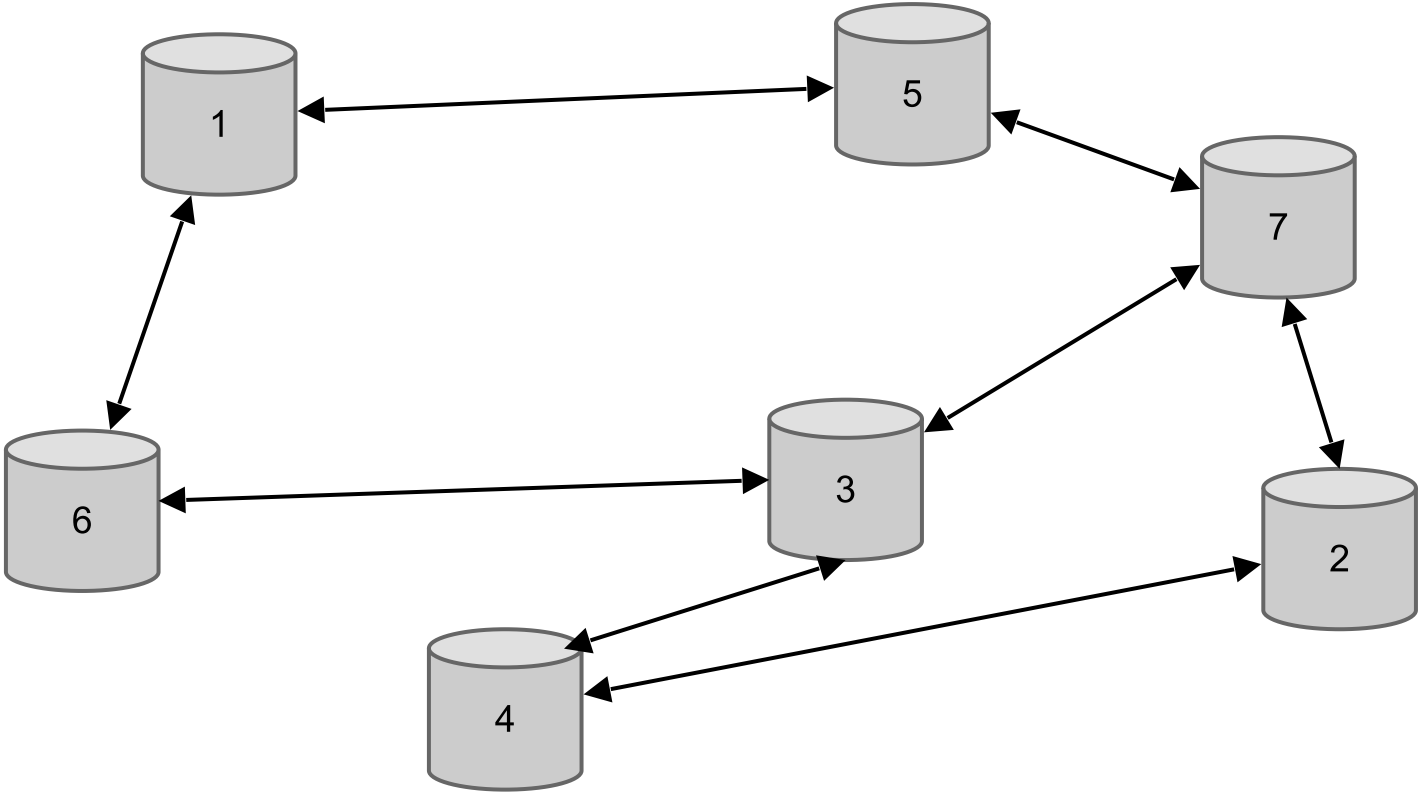
blockchain.info (and many other sites)

The Bitcoin network

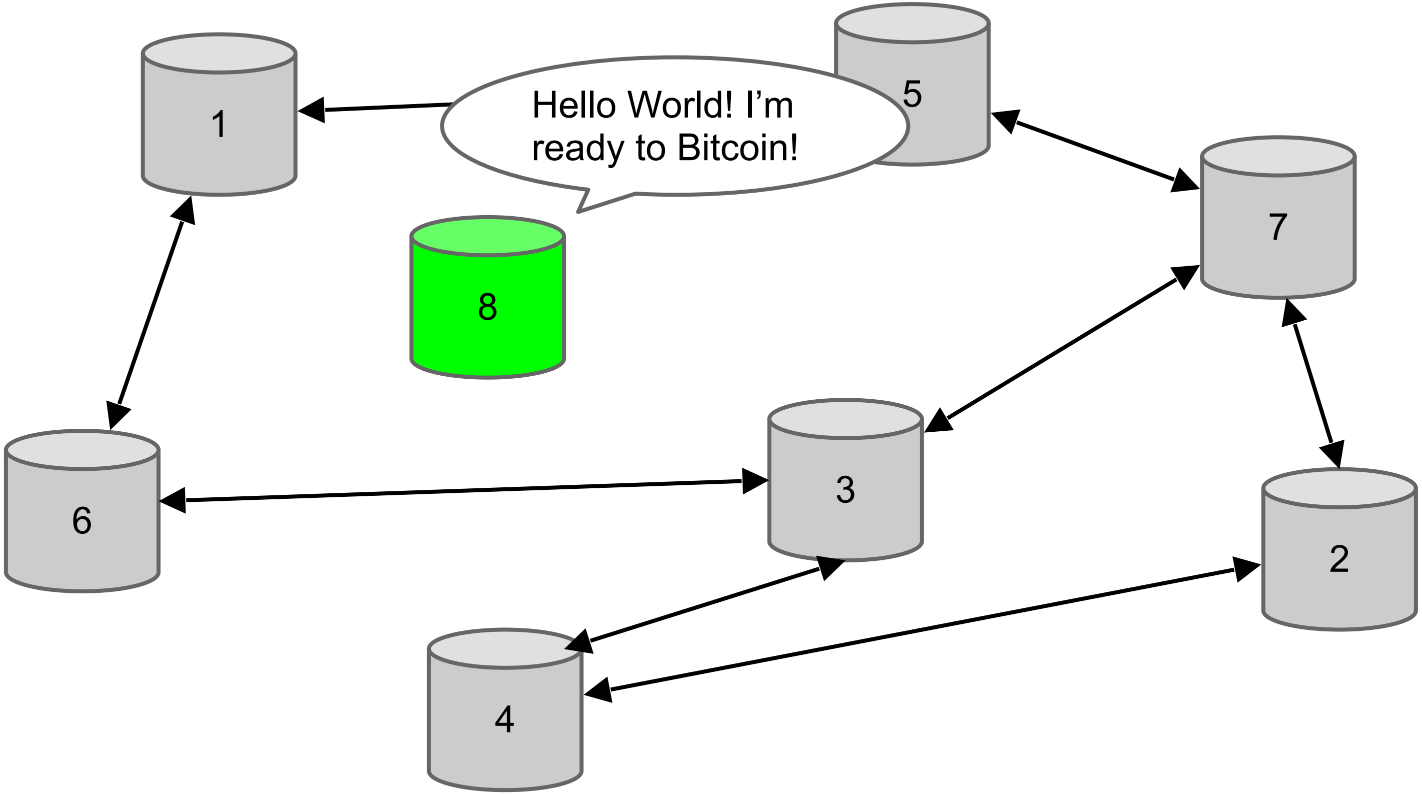
Bitcoin P2P network

- 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

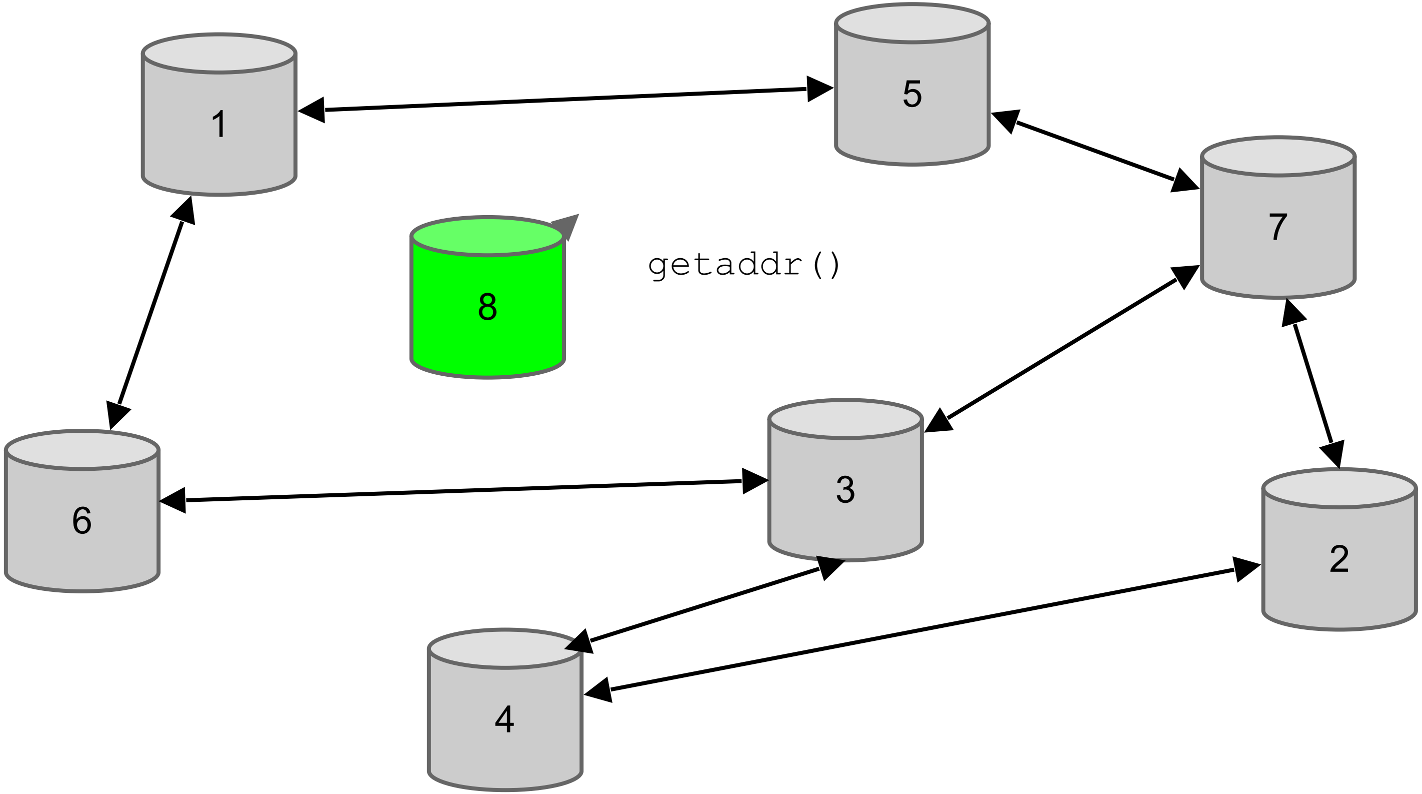
Joining the Bitcoin P2P network



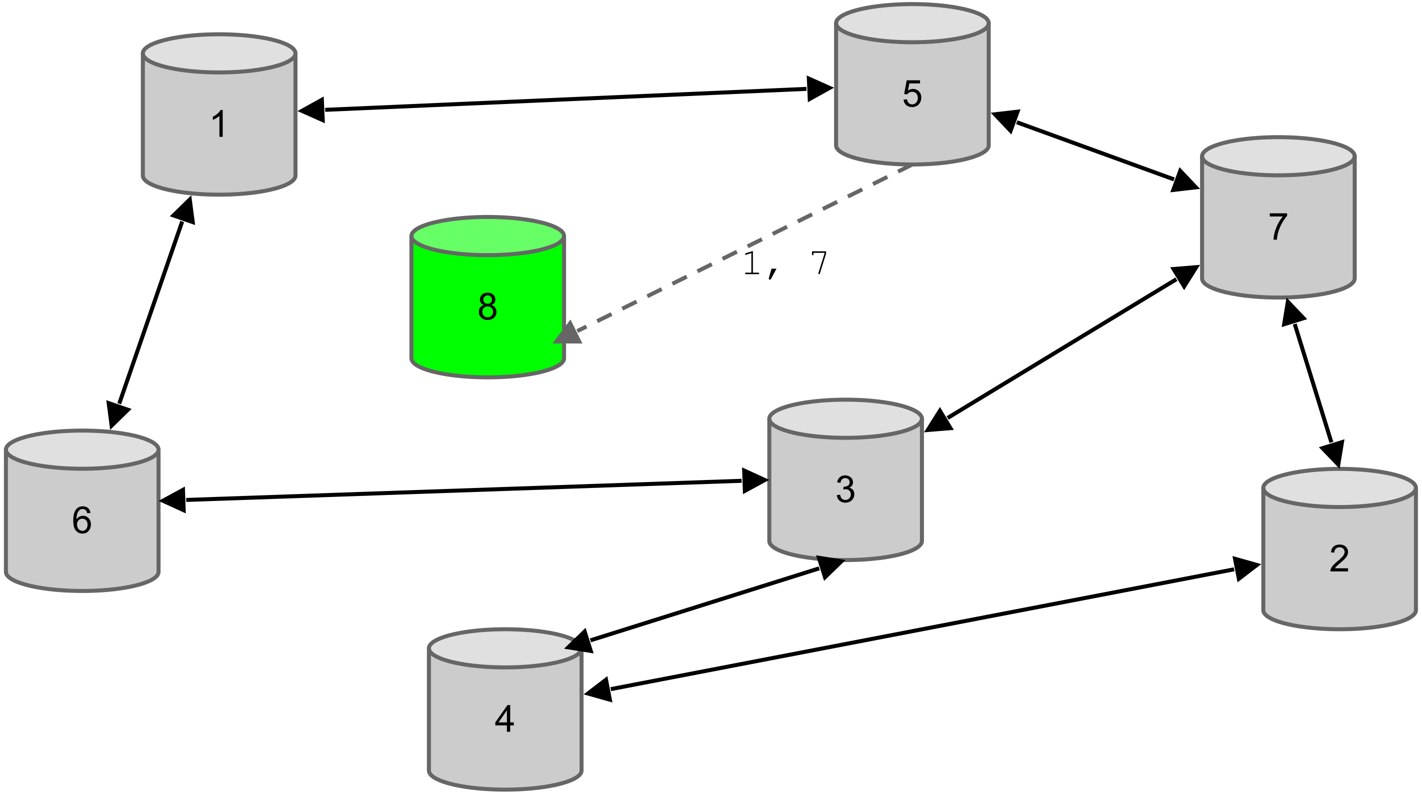
Joining the Bitcoin P2P network



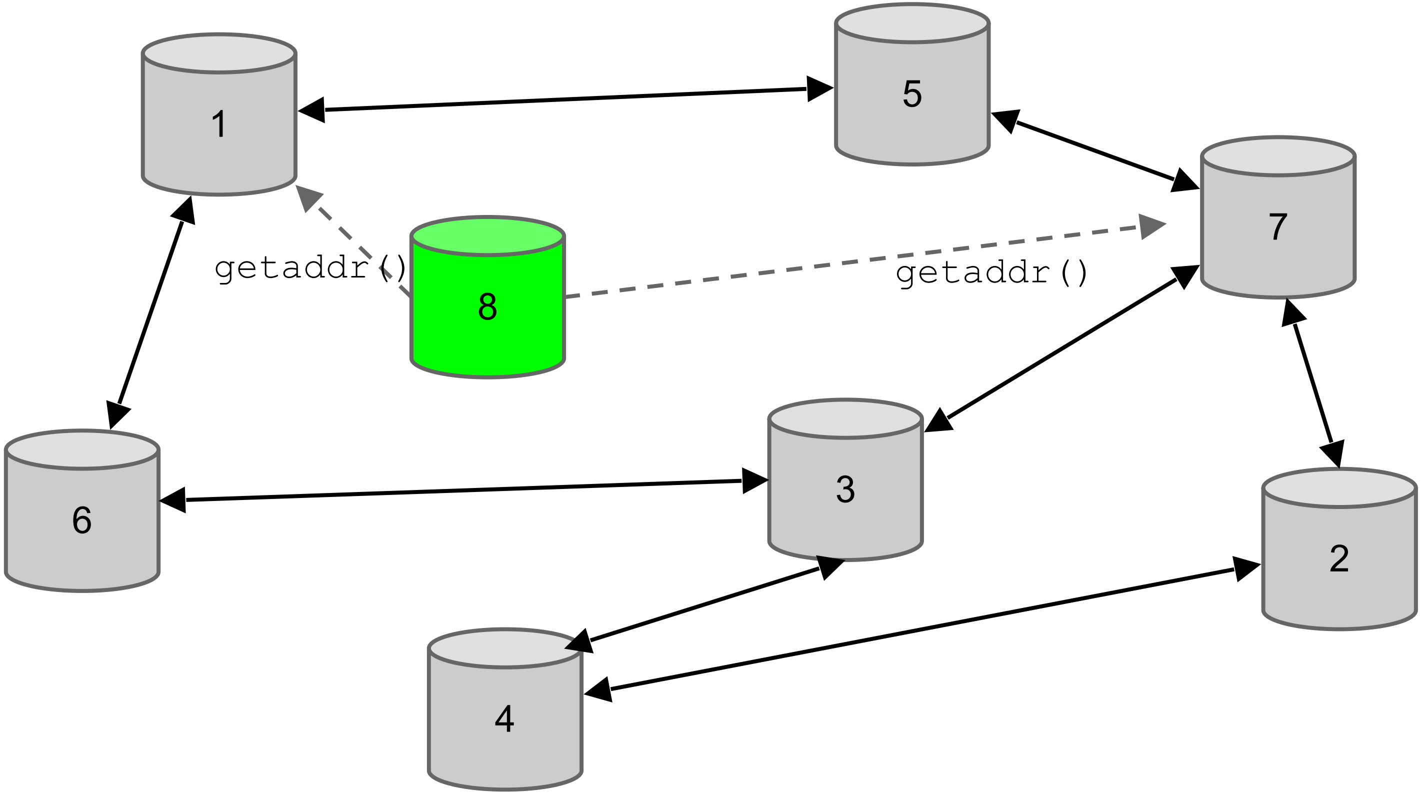
Joining the Bitcoin P2P network



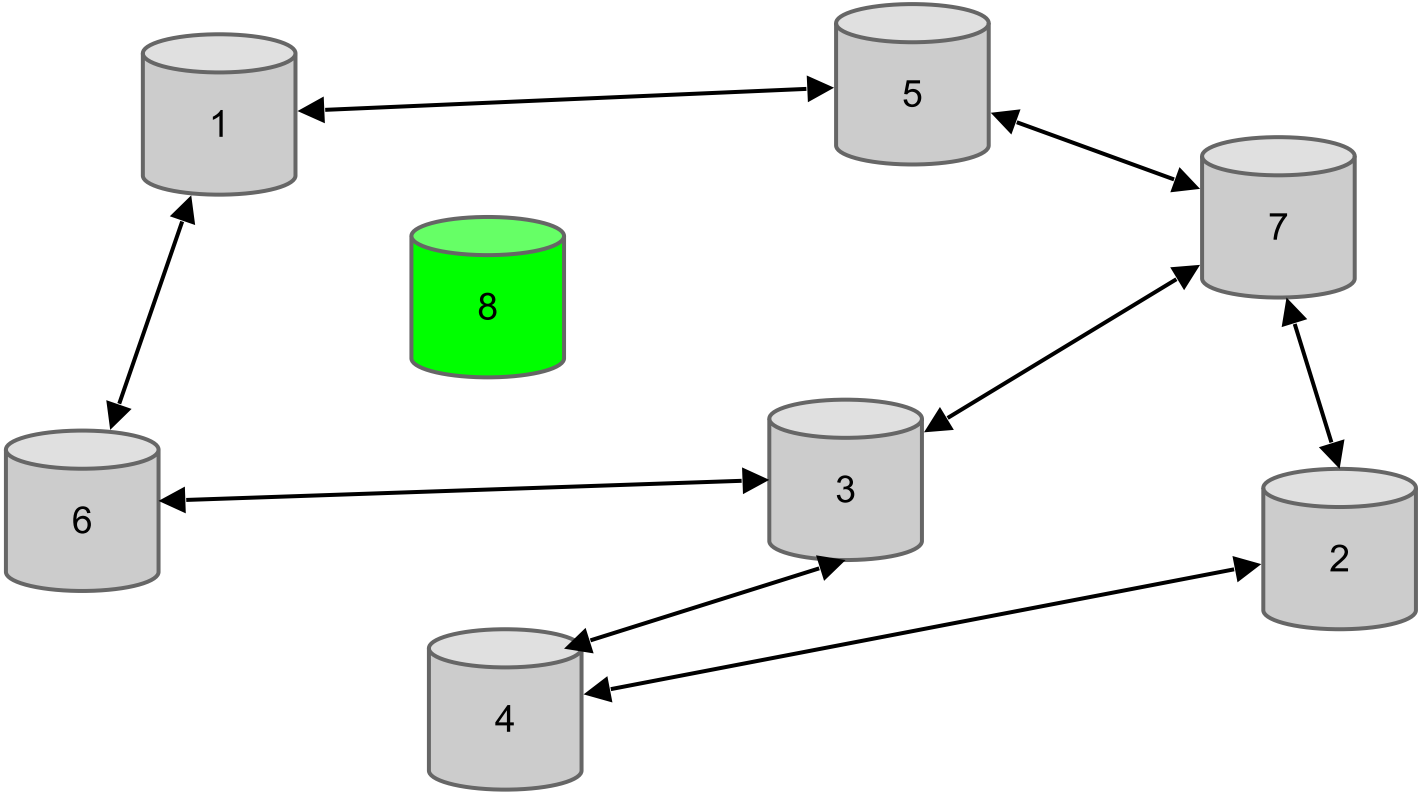
Joining the Bitcoin P2P network



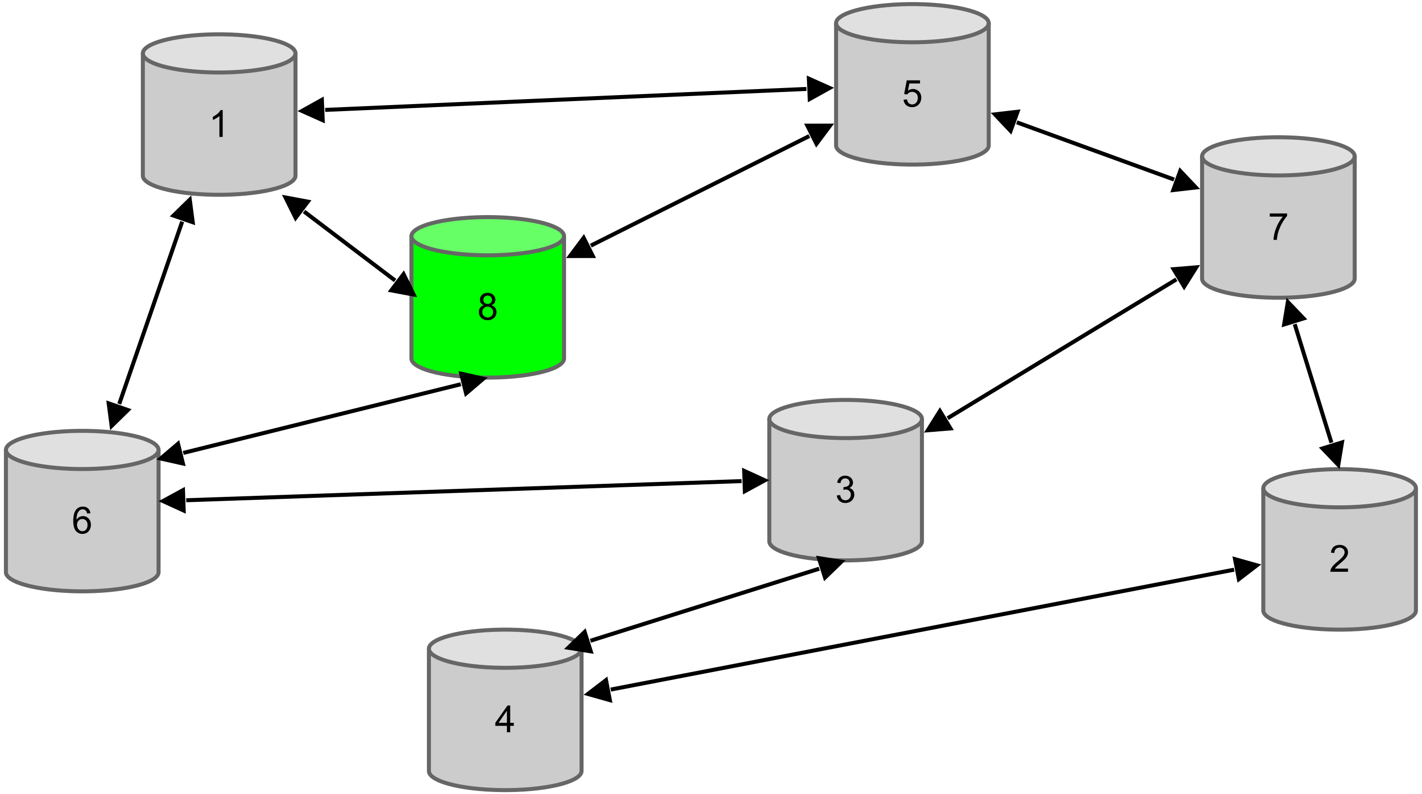
Joining the Bitcoin P2P network



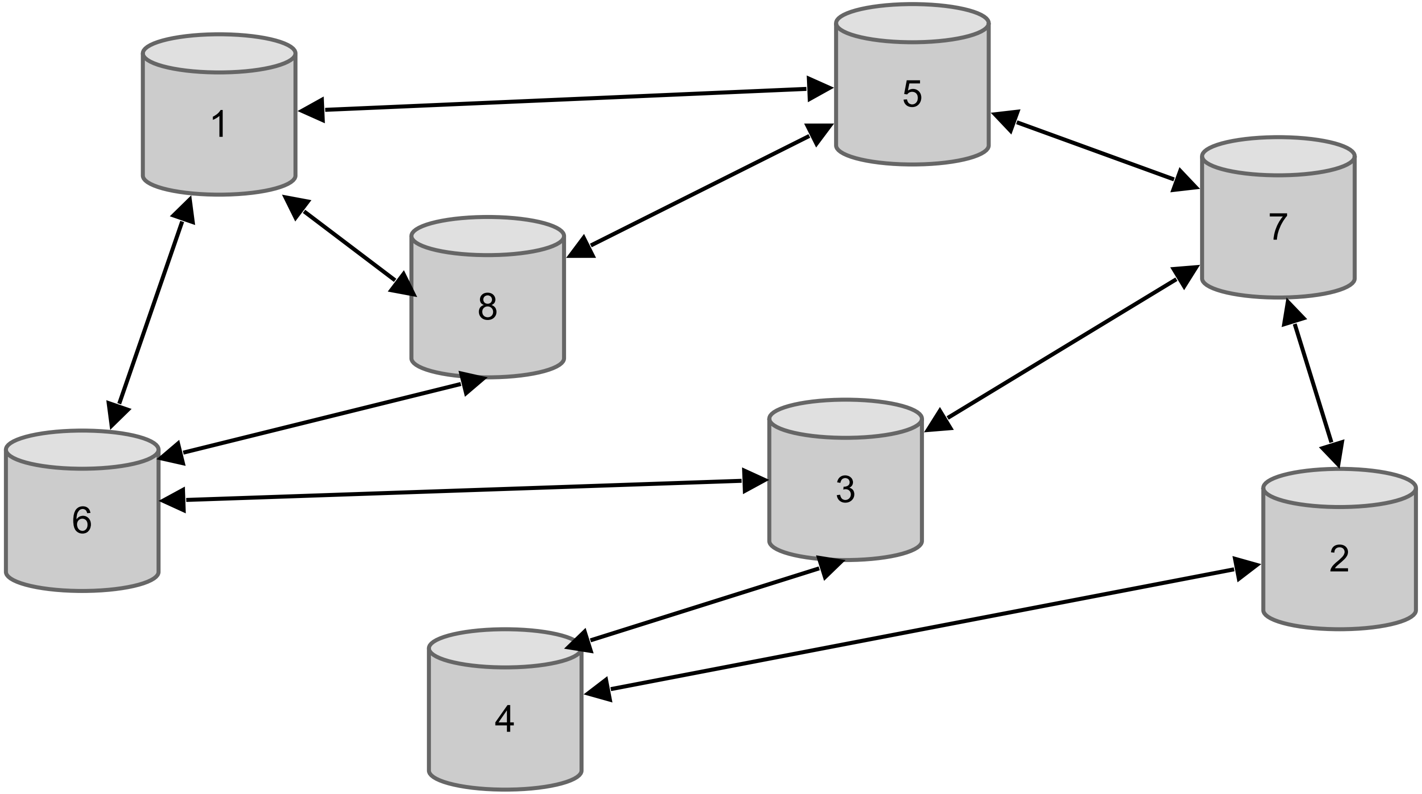
Joining the Bitcoin P2P network



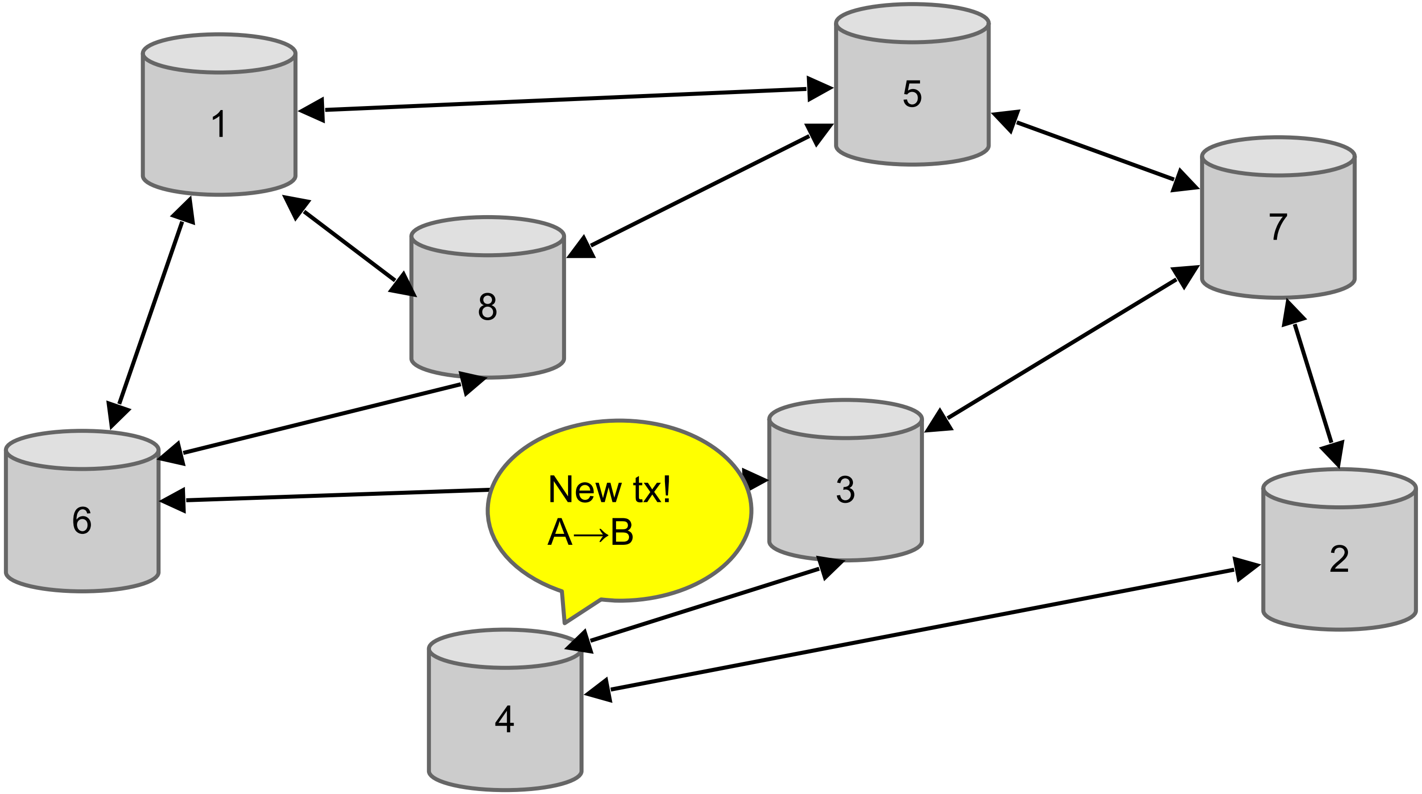
Joining the Bitcoin P2P network



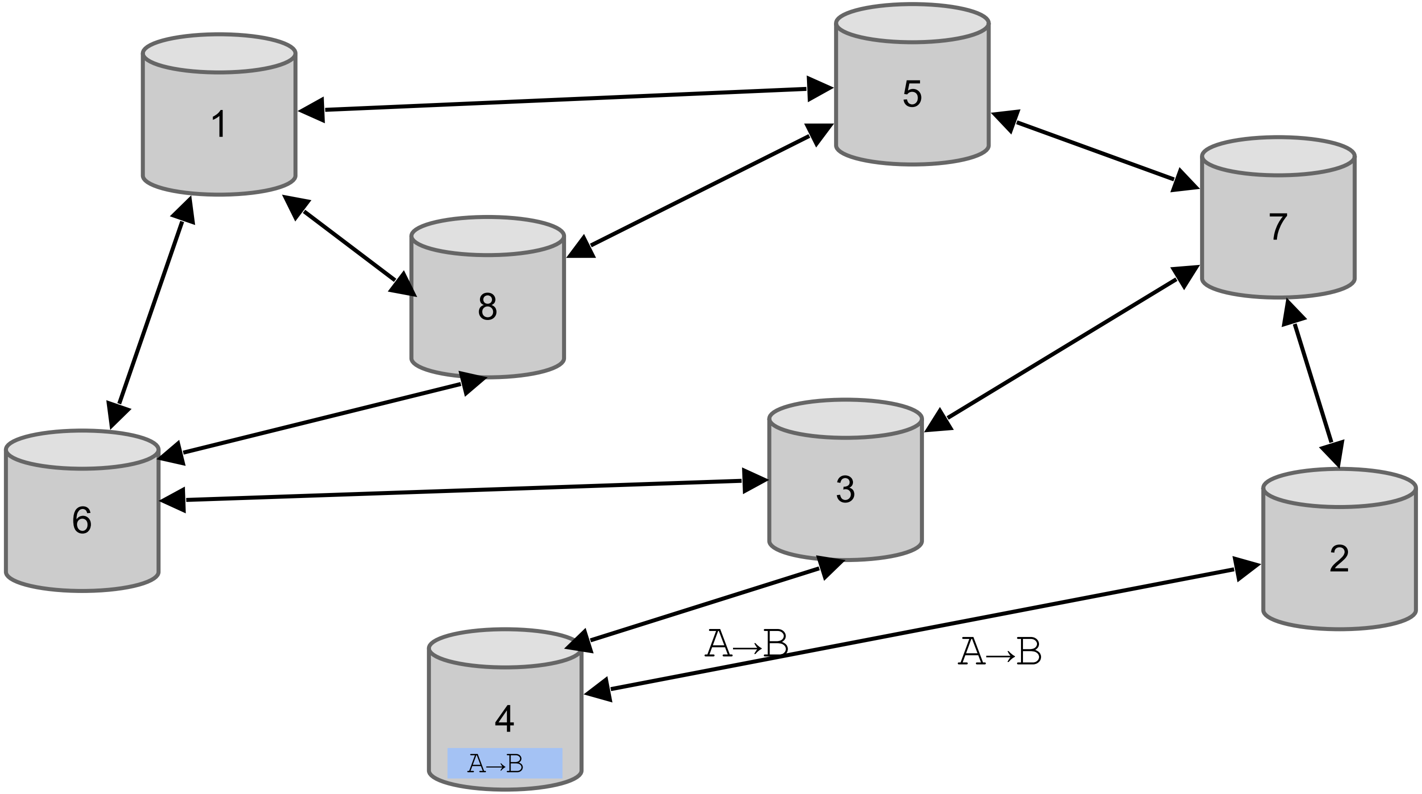
Transaction propagation (flooding)



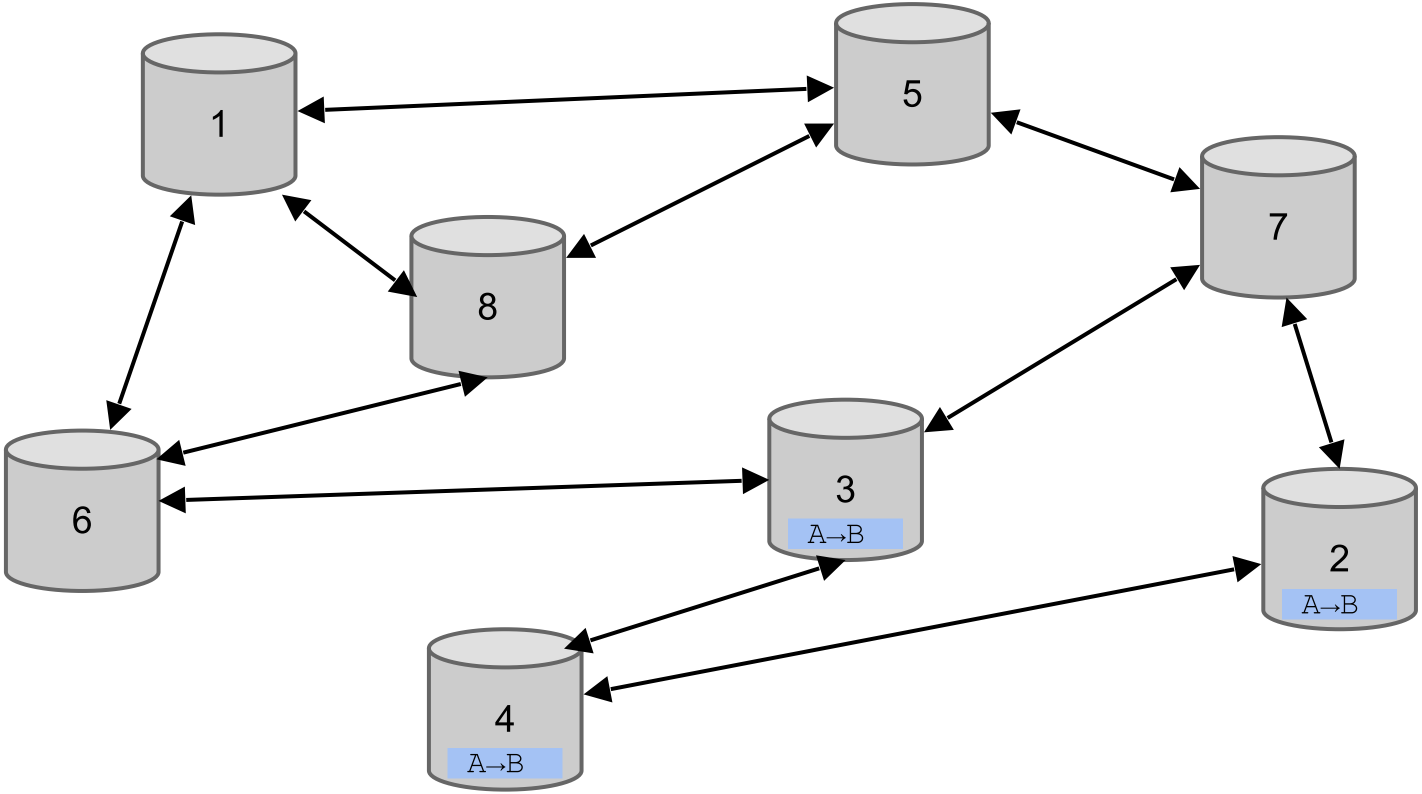
Transaction propagation (flooding)



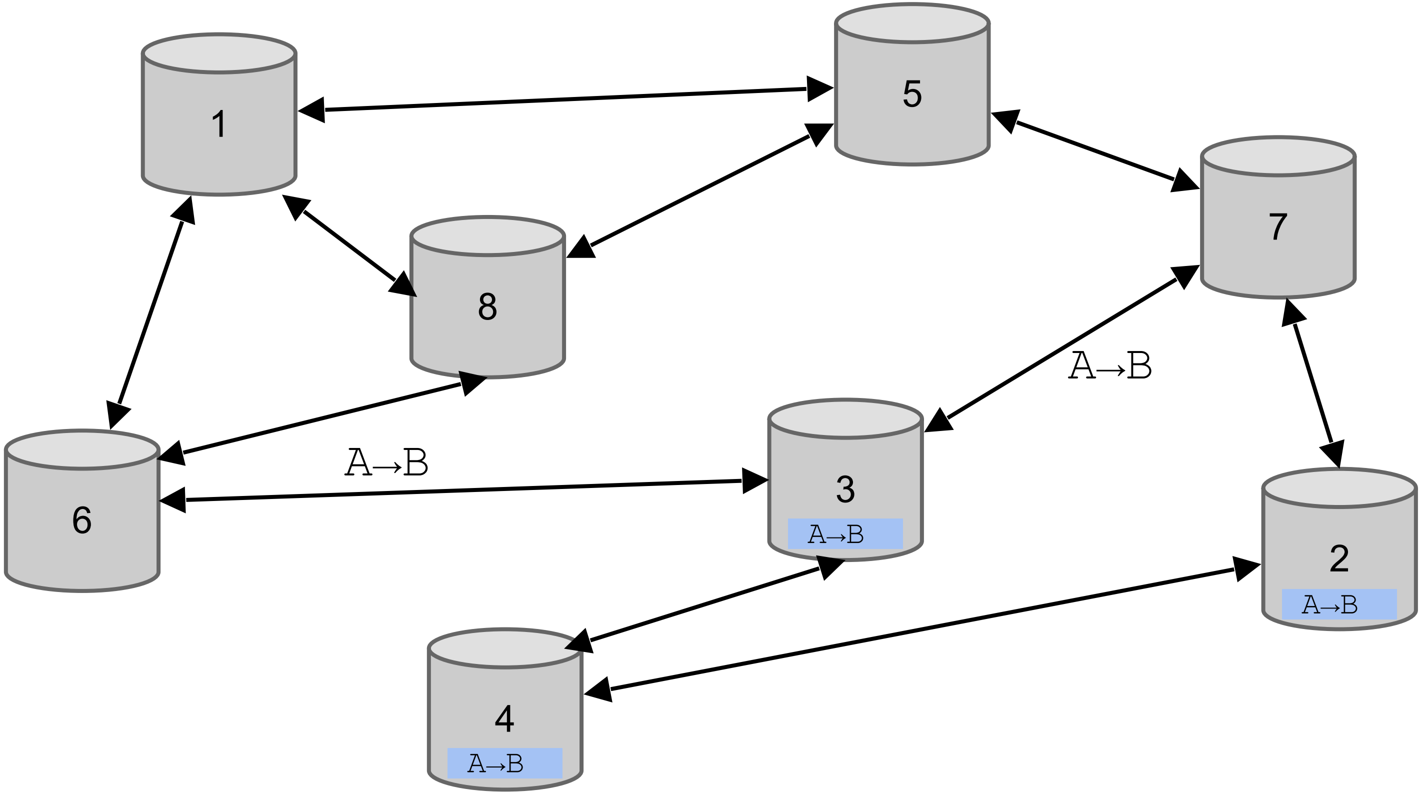
Transaction propagation (flooding)



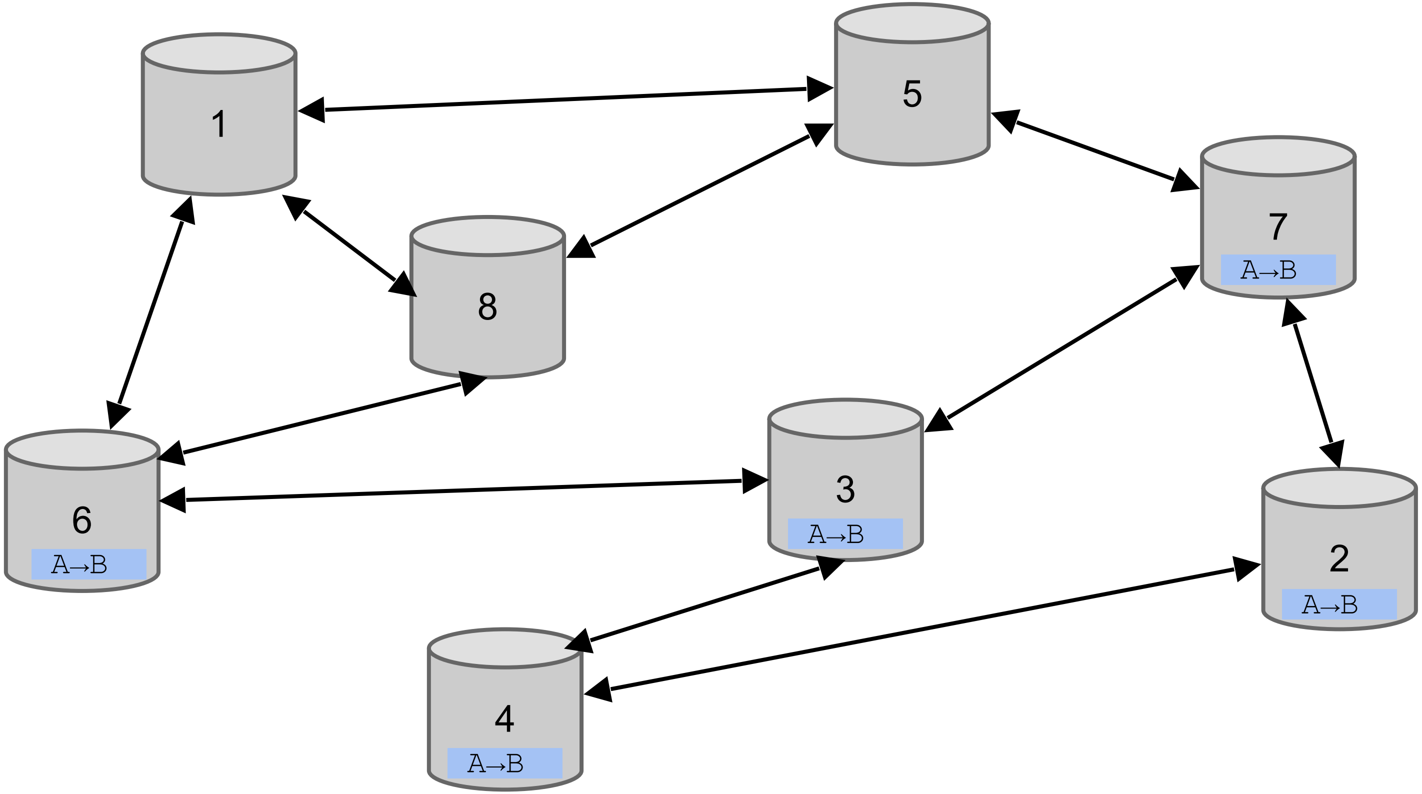
Transaction propagation (flooding)



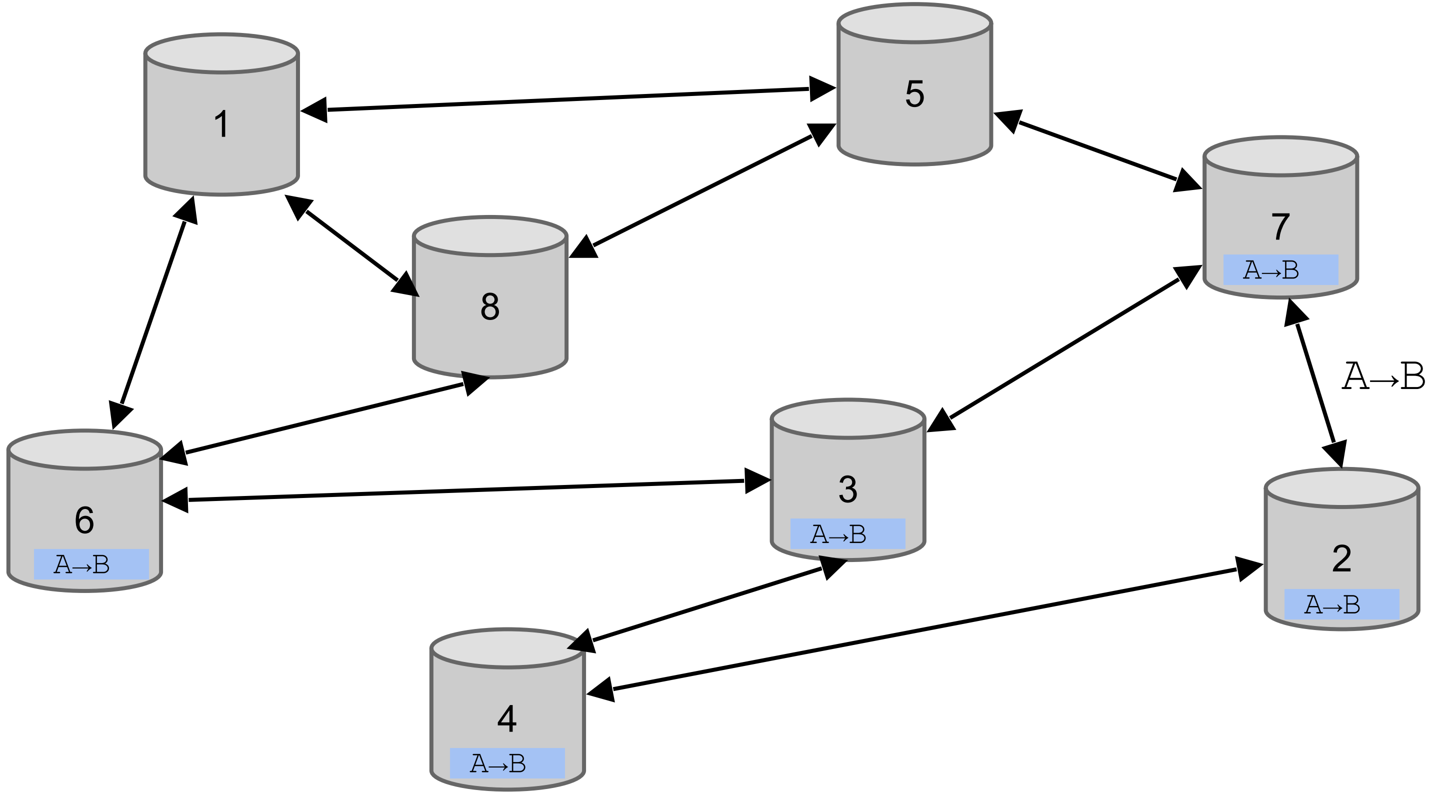
Transaction propagation (flooding)



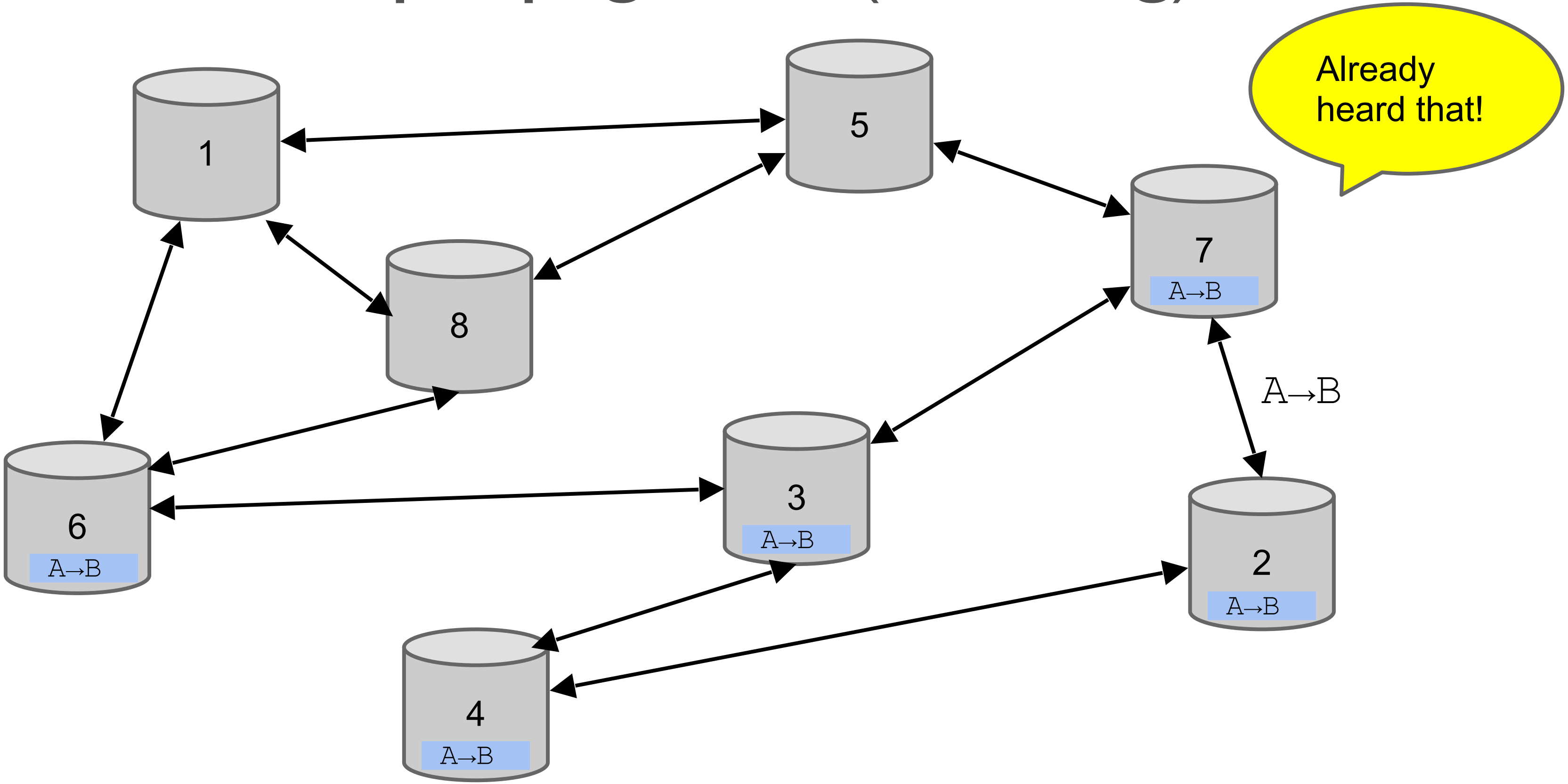
Transaction propagation (flooding)



Transaction propagation (flooding)



Transaction propagation (flooding)



Should I relay a proposed transaction?

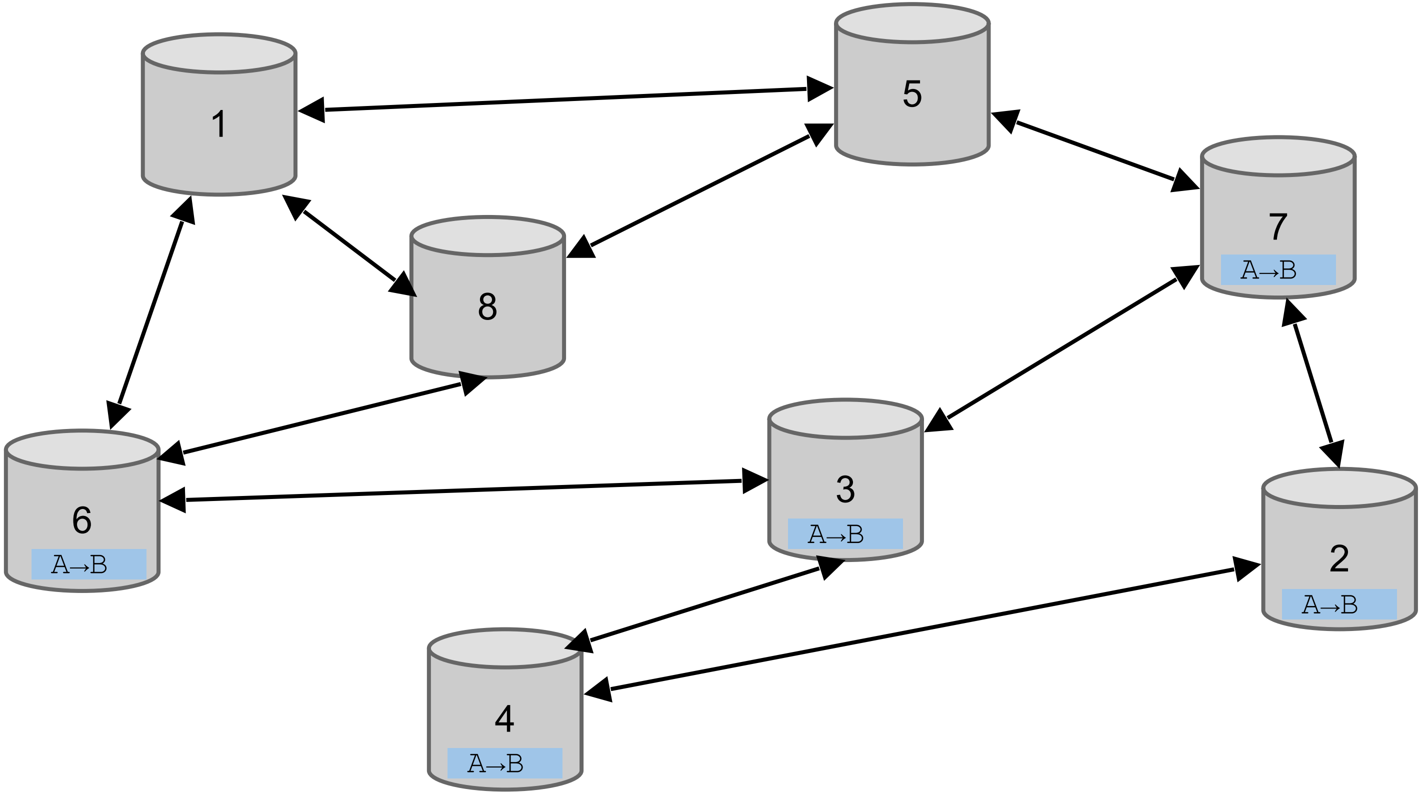
- Transaction valid with current block chain
- (default) script matches a whitelist
  - Avoid unusual scripts
- Haven't seen before
  - Avoid infinite loops
- Doesn't conflict with others I've relayed
  - Avoid double-spends

Should I relay a proposed transaction?

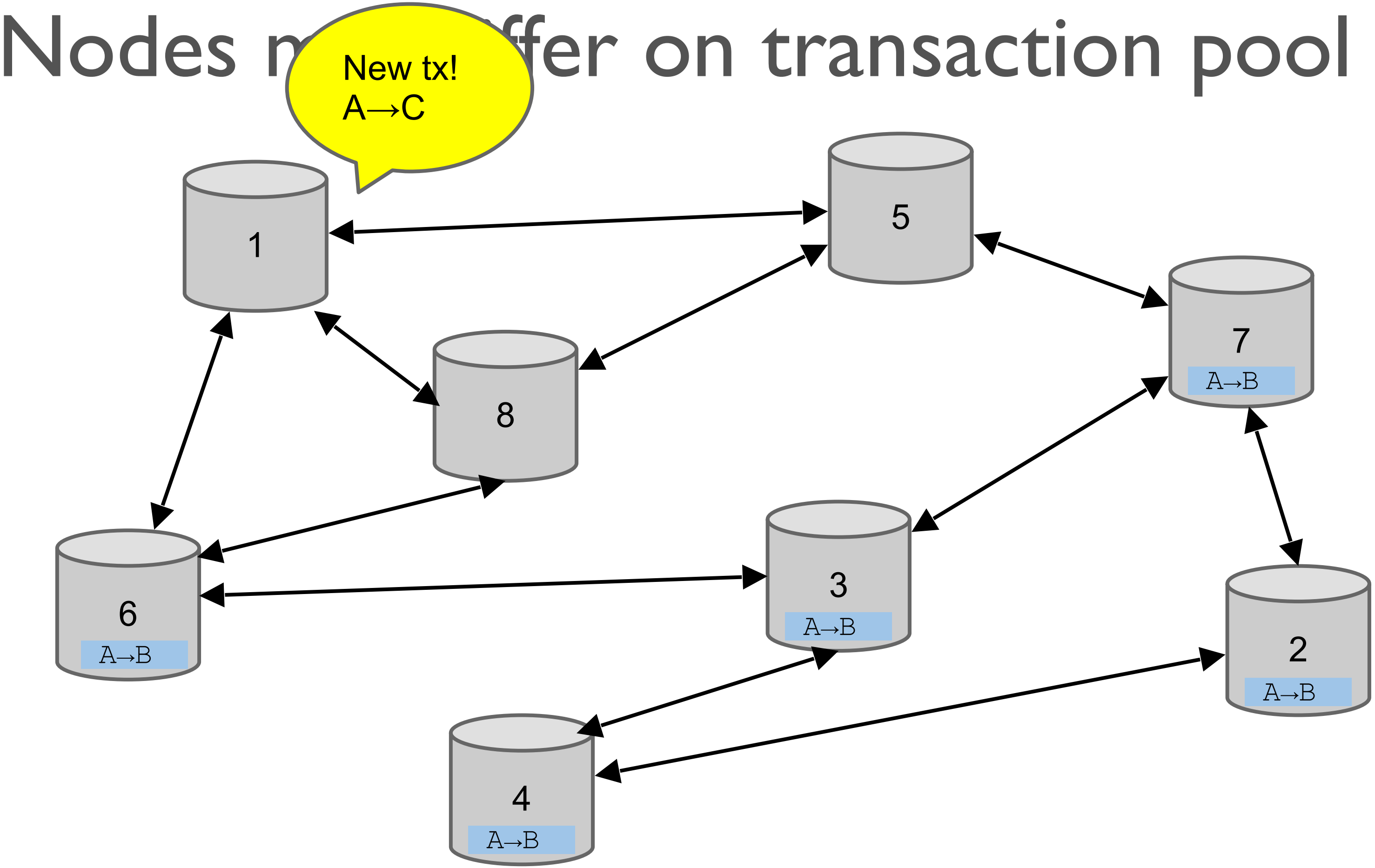
- Transaction valid with current block chain
- (default) script matches a whitelist
  - Avoid unusual scripts
- Haven’t seen before
  - Avoid infinite loops
- Doesn’t conflict with others I’ve relayed
  - Avoid double-spends

Sanity checks only...
Some nodes may ignore them!

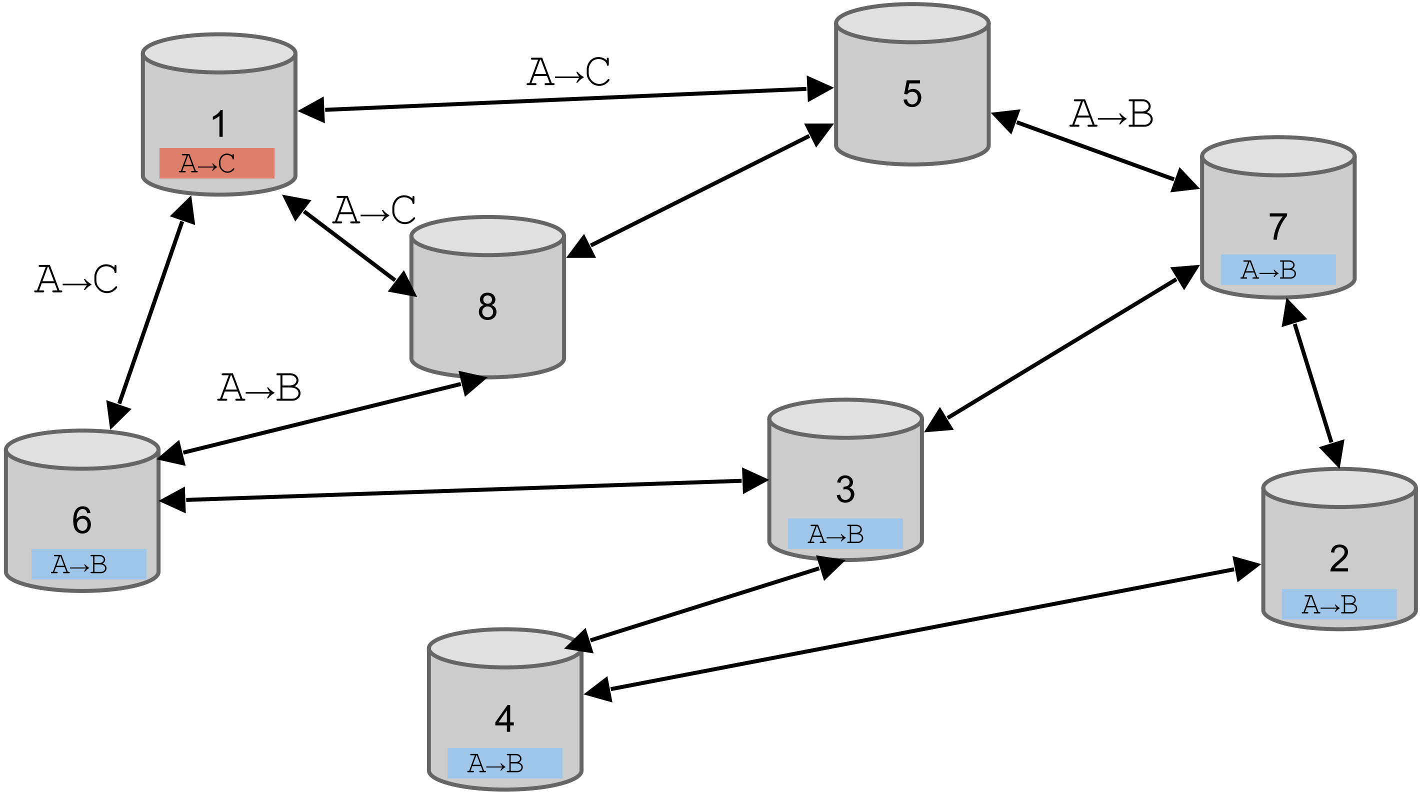
Nodes may differ on transaction pool



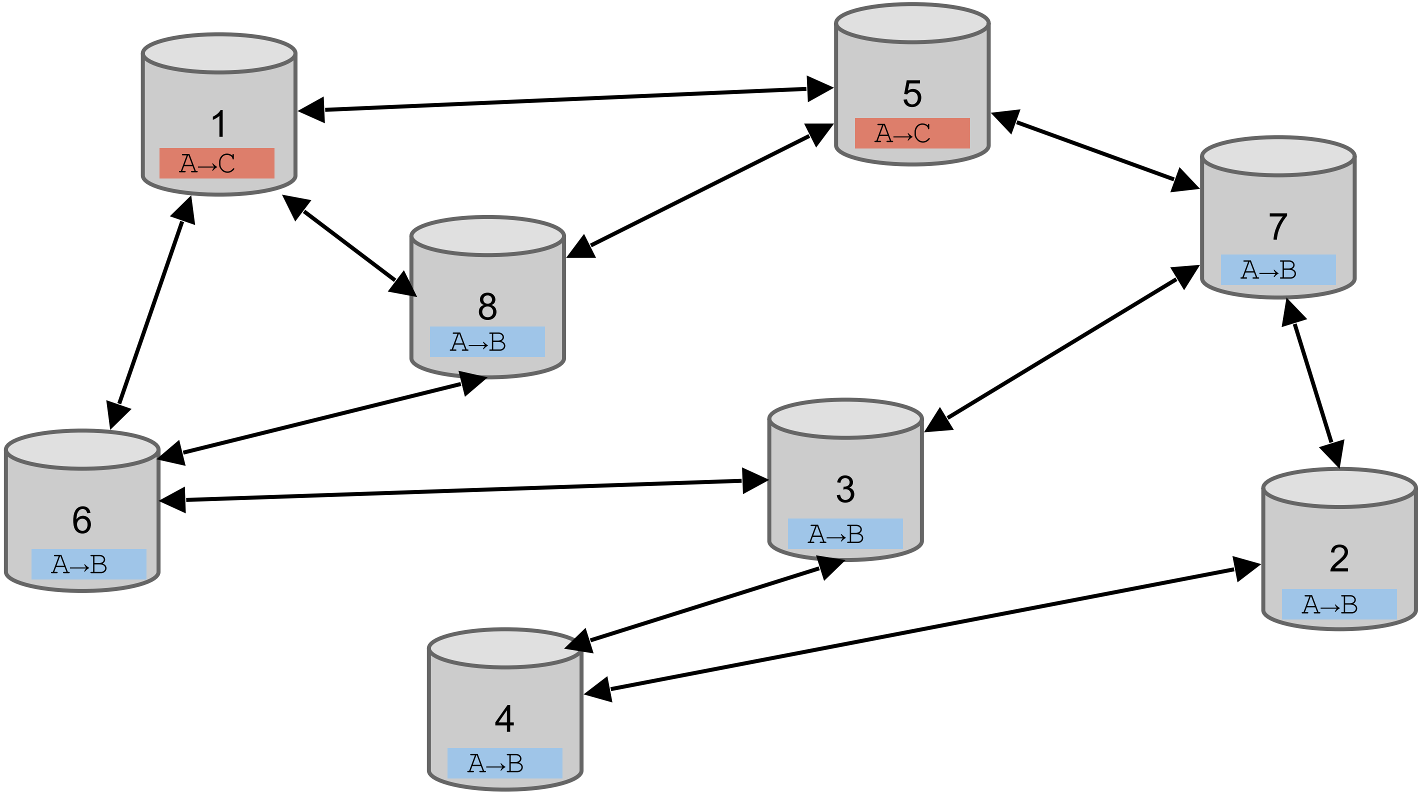
Nodes m ffer on transaction pool



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:

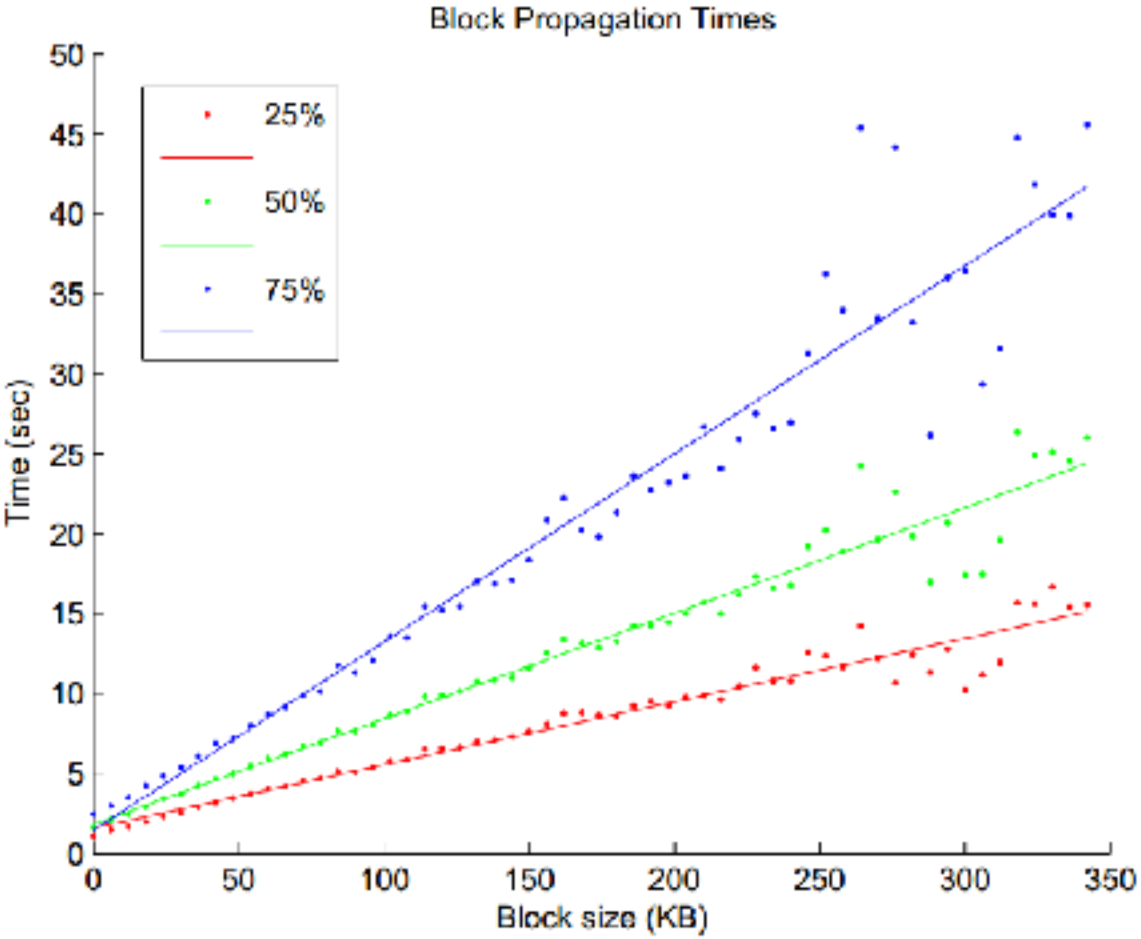
- Block meets the hash target
- Block has all valid transactions
  - Run ***all*** scripts, even if you wouldn't relay
- Block builds on current longest chain
  - Avoid forks

Block propagation nearly identical

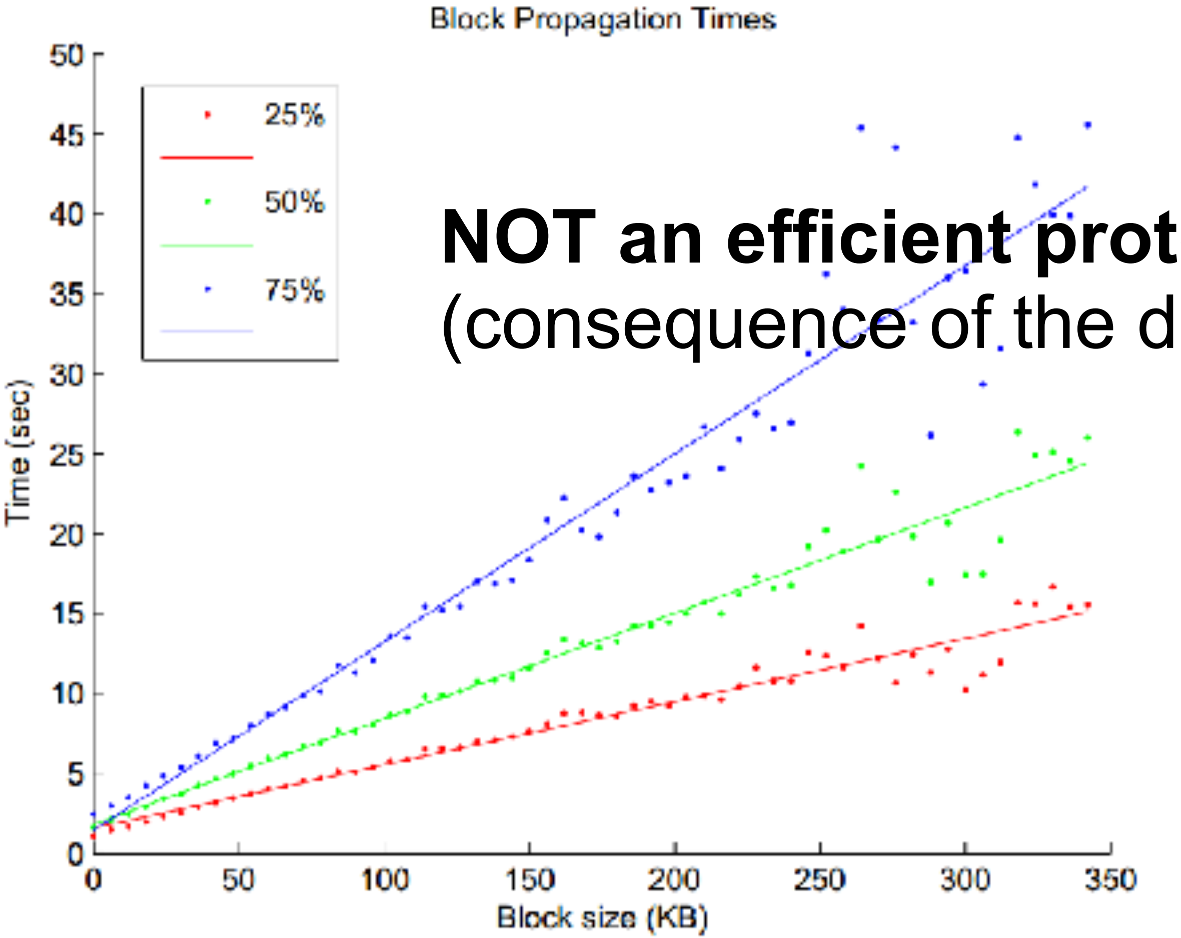
Relay a new block when you hear it if:

- 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

Sanity check
Also may be ignored...



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!

*(old numbers, might be outdated)

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

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
- 50,25,12.5,6.25… 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
- 50,25,12.5,6.25… 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 tran

Compare

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

Improving throughput:
strong motivation for Altcoins

Cryptographic limits in Bitcoin

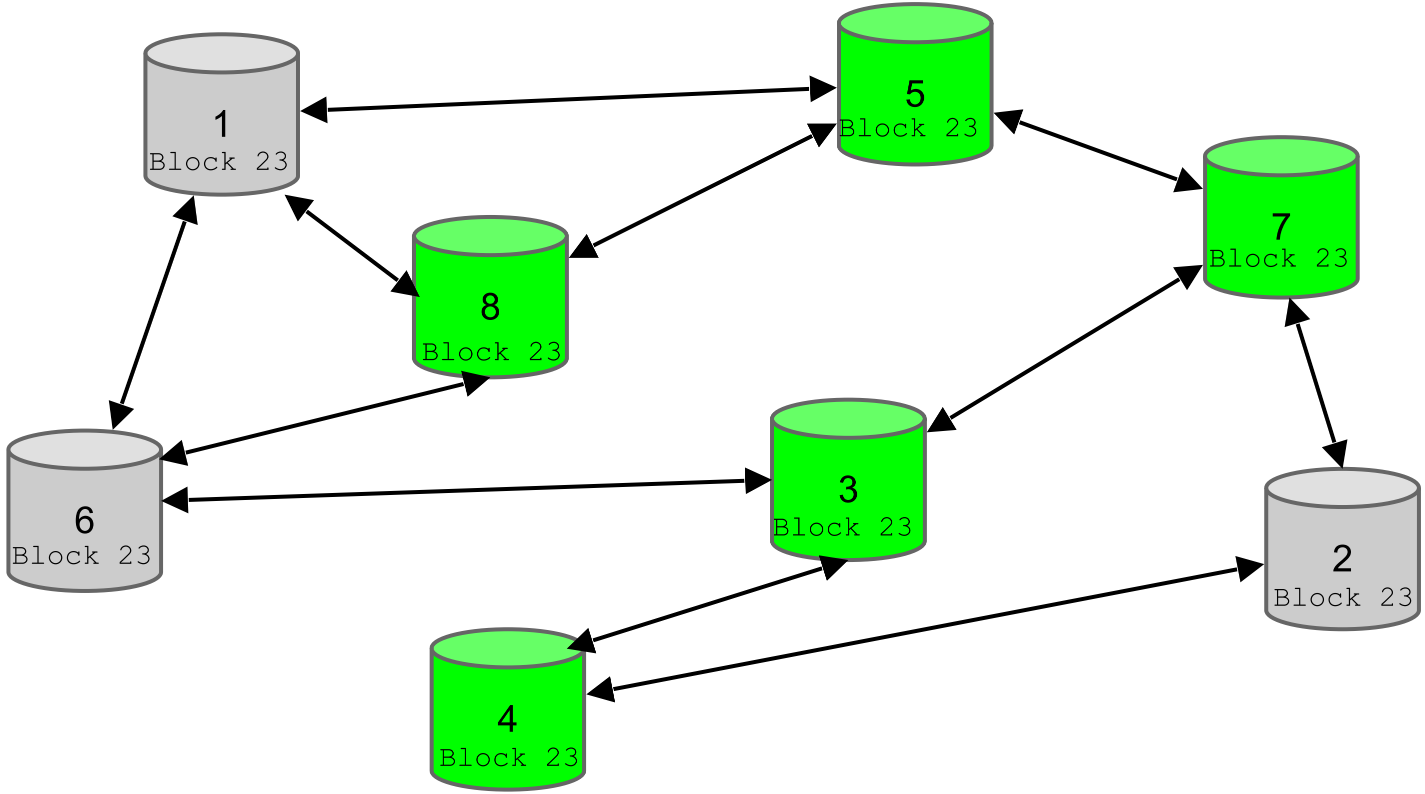
- Only 1 signature algorithm (ECDSA/P256)
- Hard-coded hash functions

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

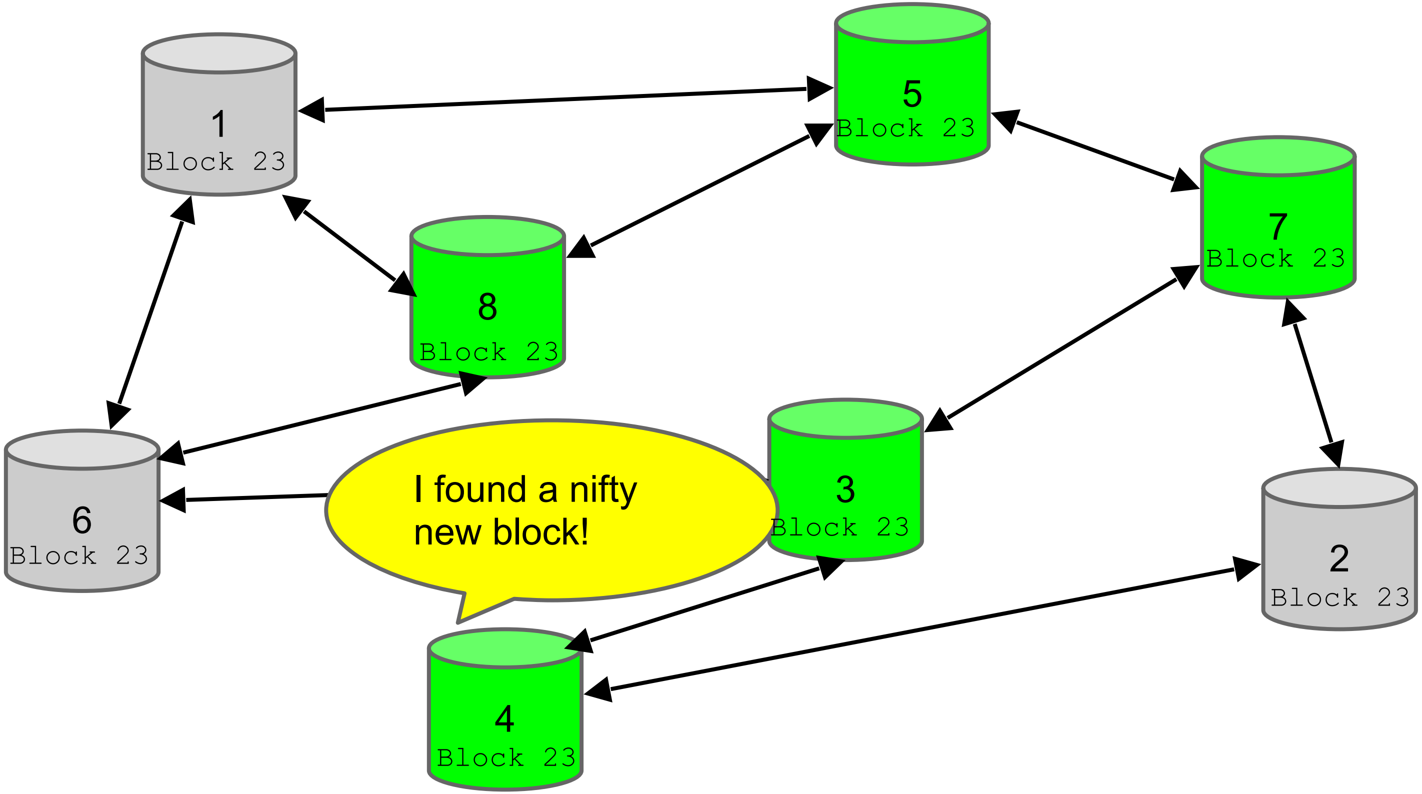
Why not update Bitcoin software to overcome these limitations?

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

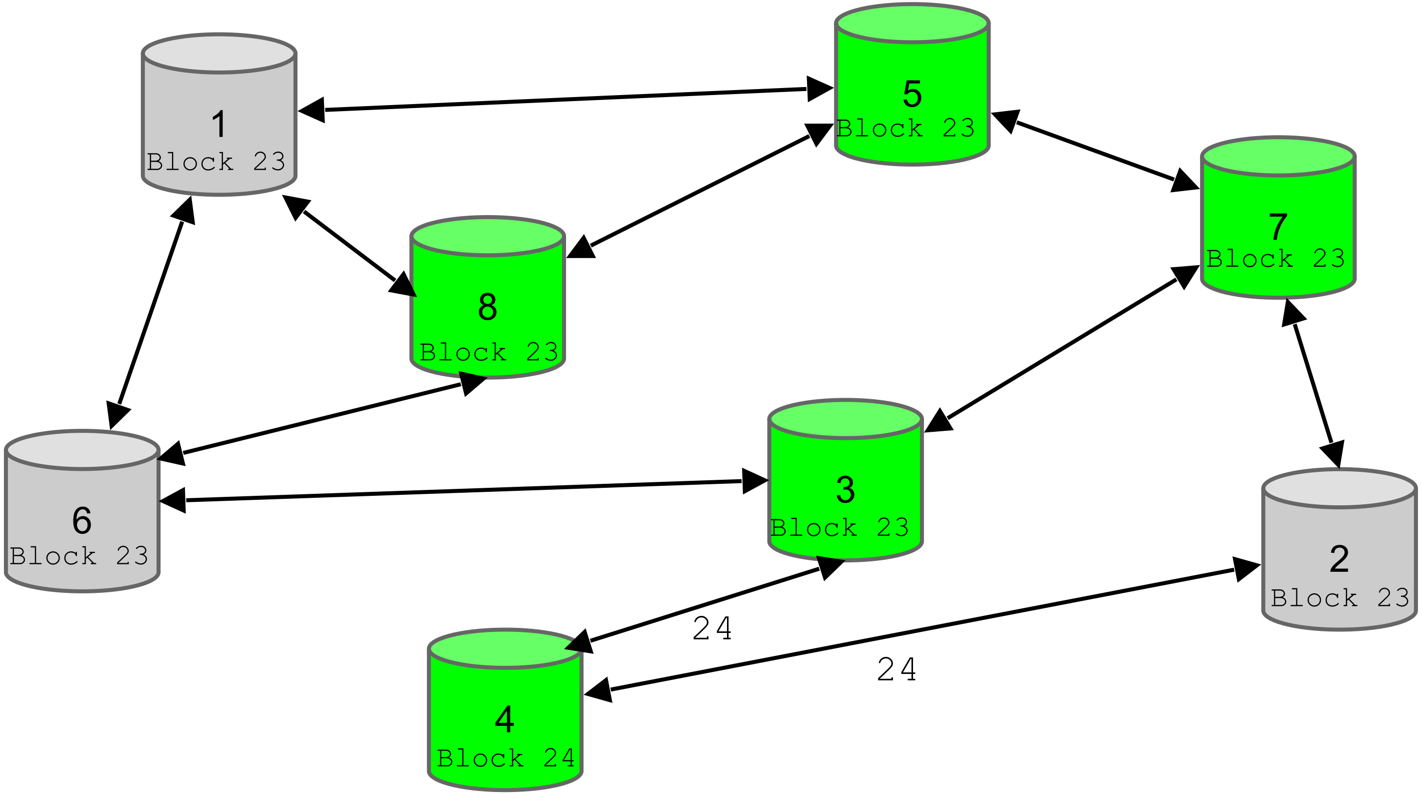
“Hard-forking” changes to Bitcoin



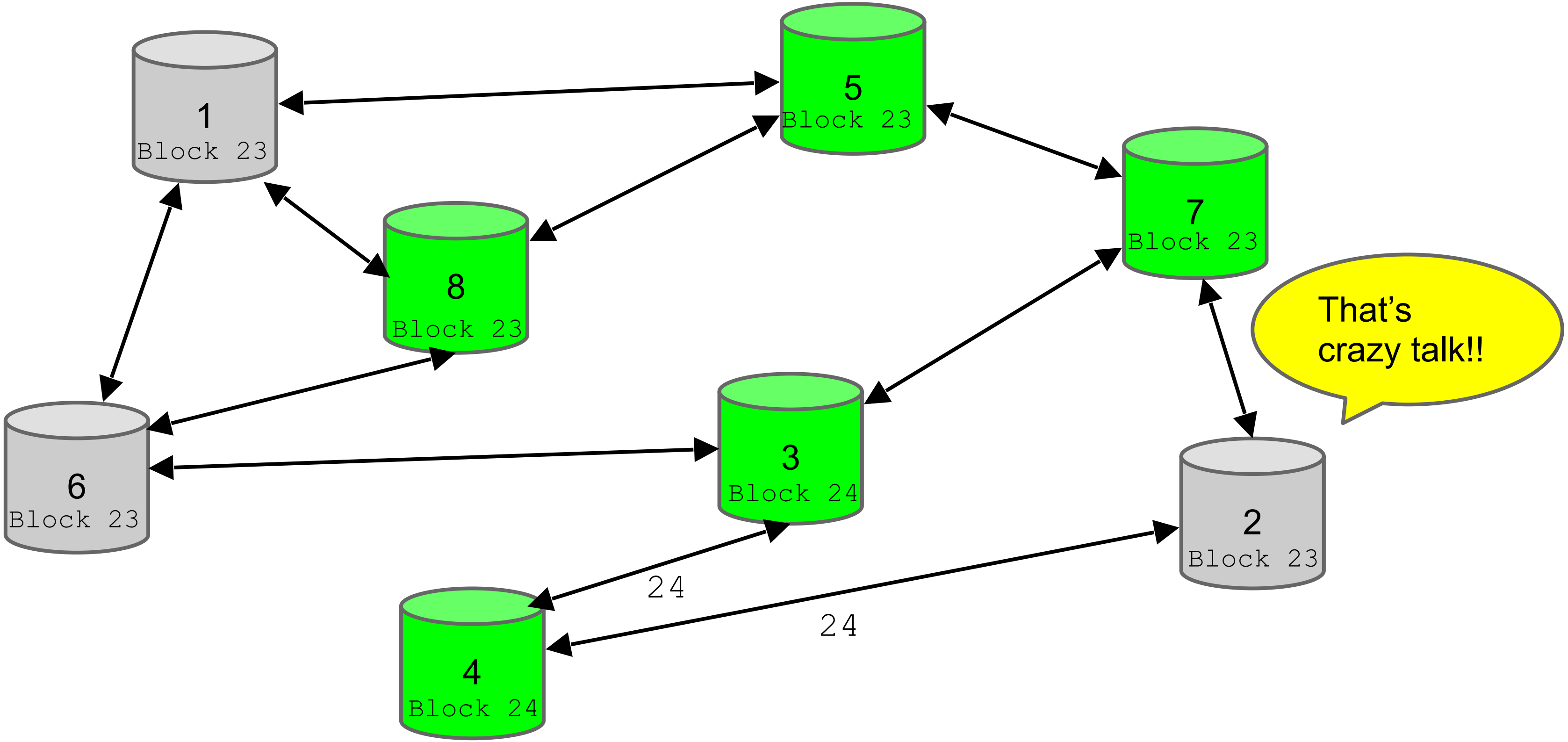
“Hard-forking” changes to Bitcoin



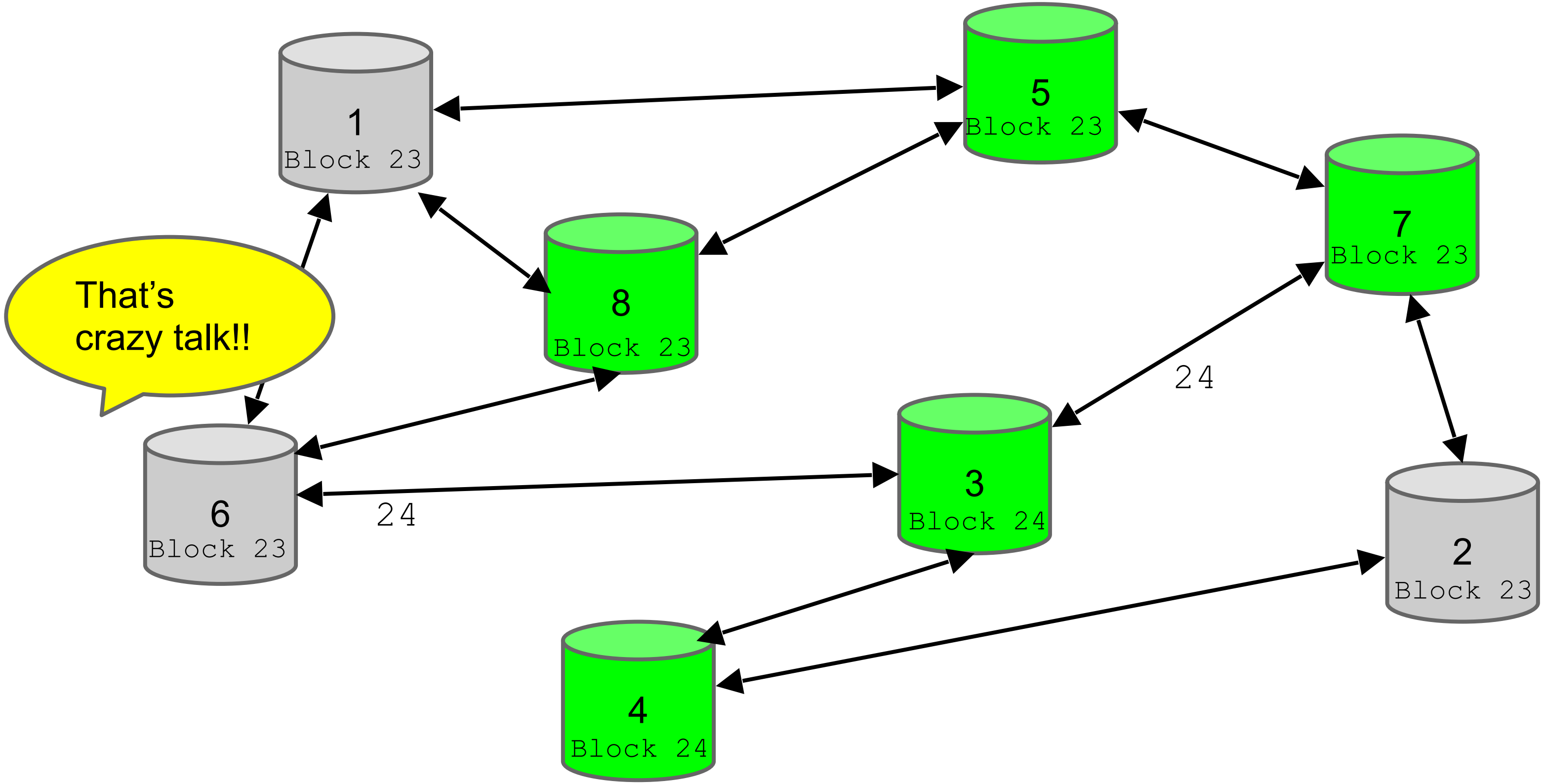
“Hard-forking” changes to Bitcoin



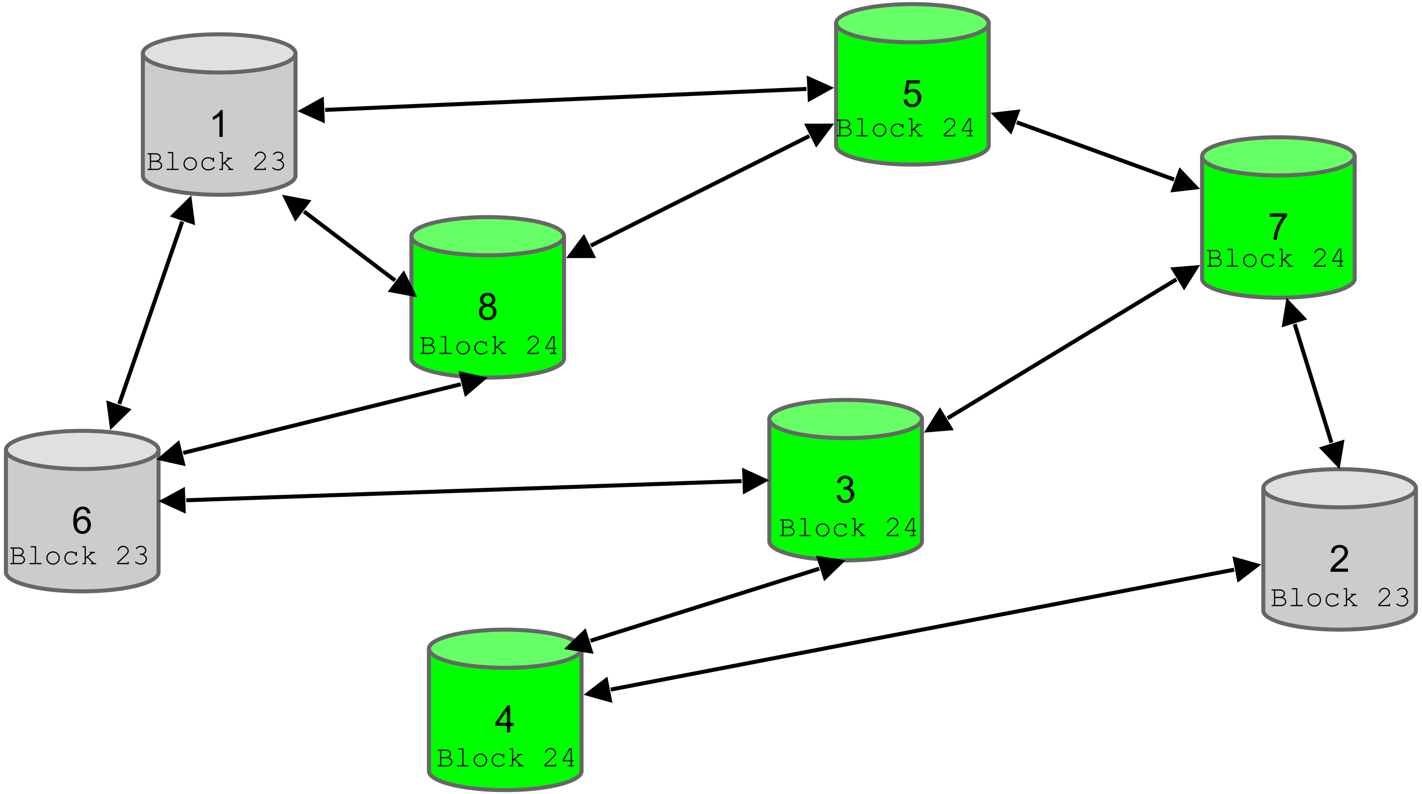
“Hard-forking” changes to Bitcoin



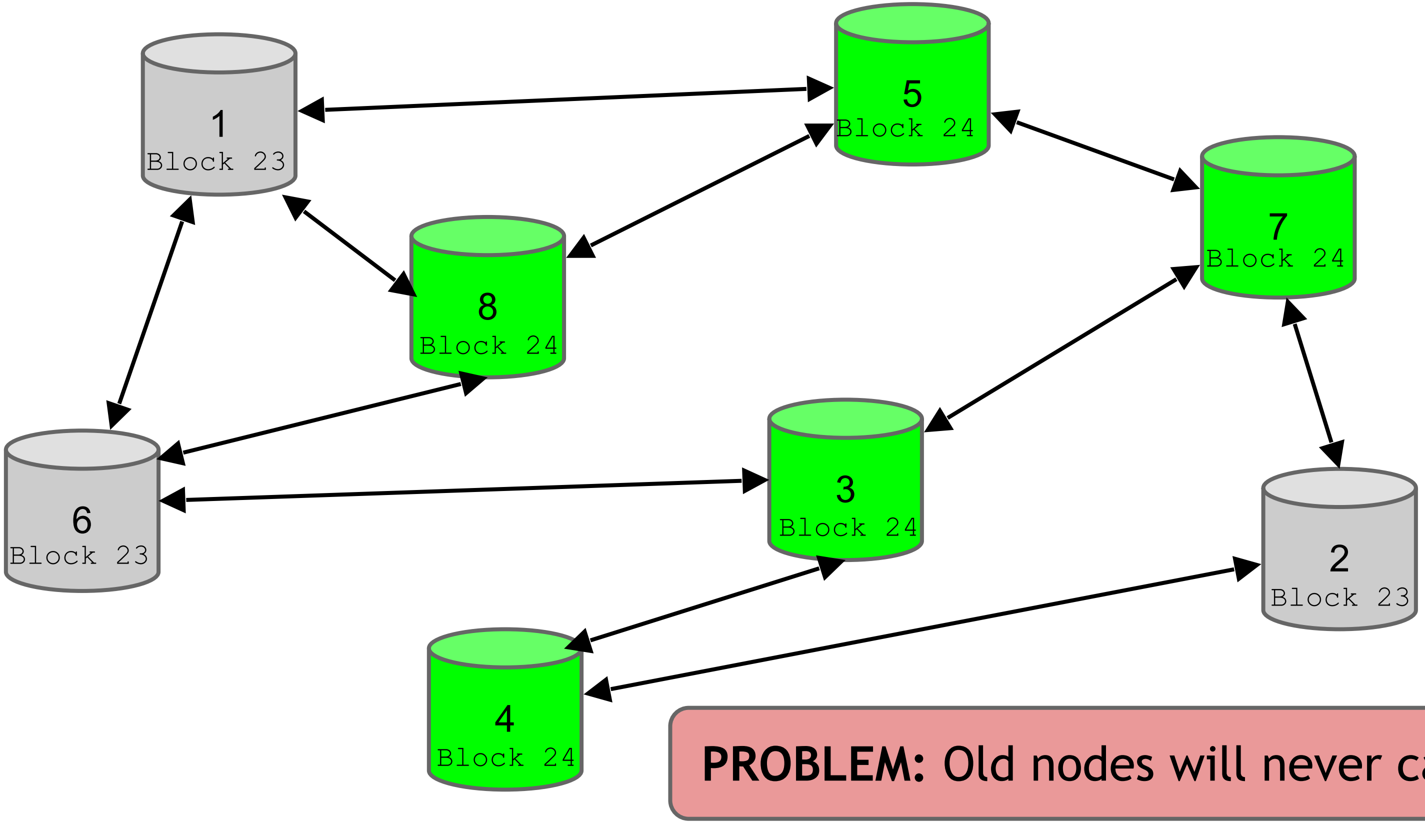
“Hard-forking” changes to Bitcoin



“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