

# CS 731: Blockchain Technology And Applications

Sandeep K. Shukla  
IIT Kanpur

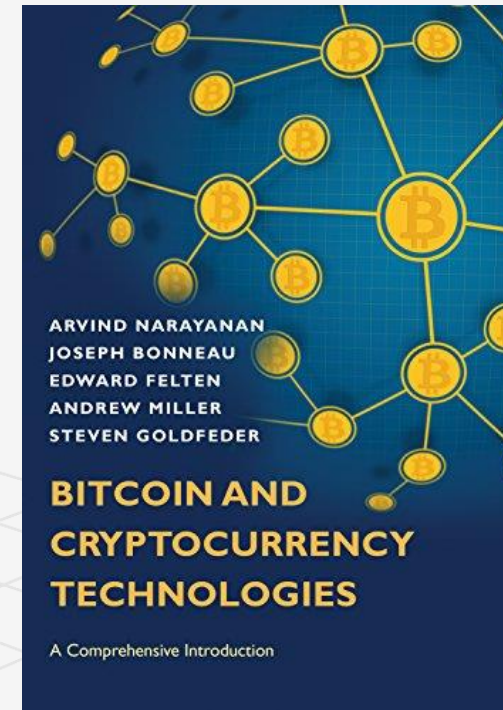
C3I Center



# Acknowledgement

---

- The material of this lecture material is mostly due to Prof. Arvind Narayanan's Lecture at Princeton and his book on Bitcoin (Chapter 3)



# Mechanics of Bitcoin

# Recap: Bitcoin consensus

---

Bitcoin consensus gives us:

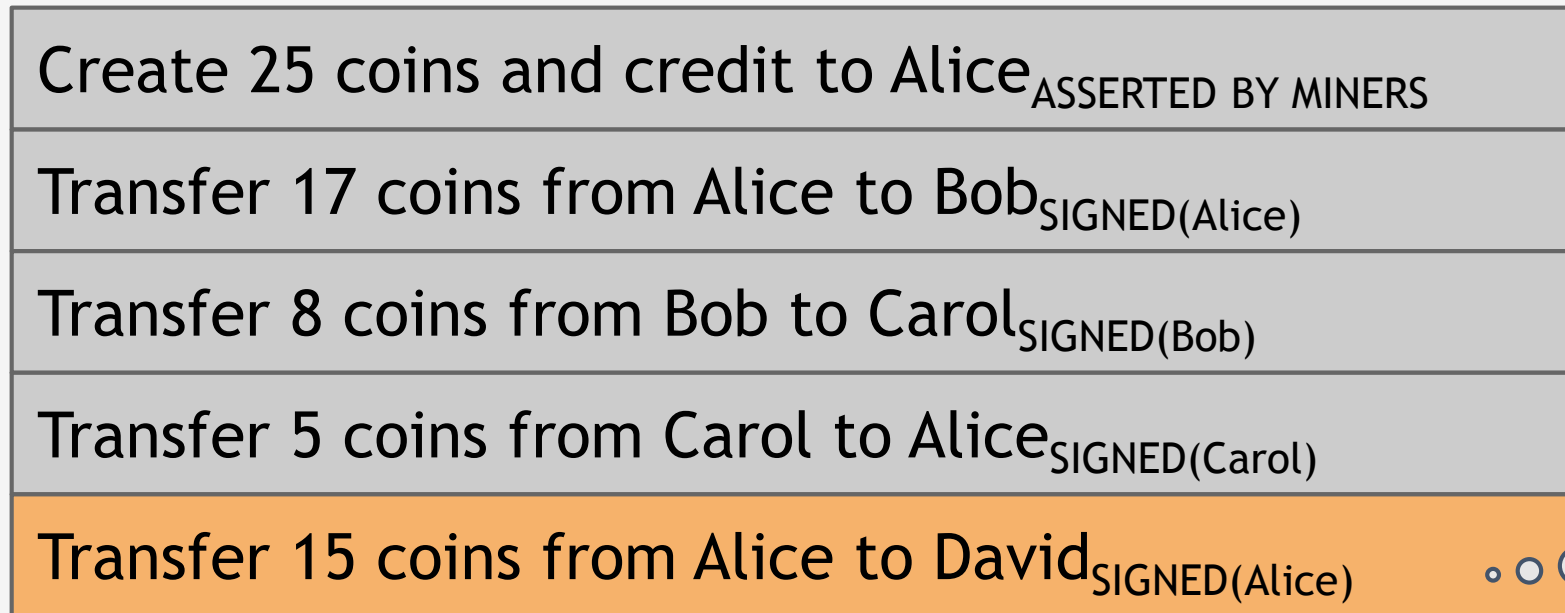
- Append-only ledger
- Decentralized consensus
- Miners to validate transactions

*assuming a currency exists to motivate miners!*

# Bitcoin transactions

# An account-based ledger (*not* Bitcoin)

time

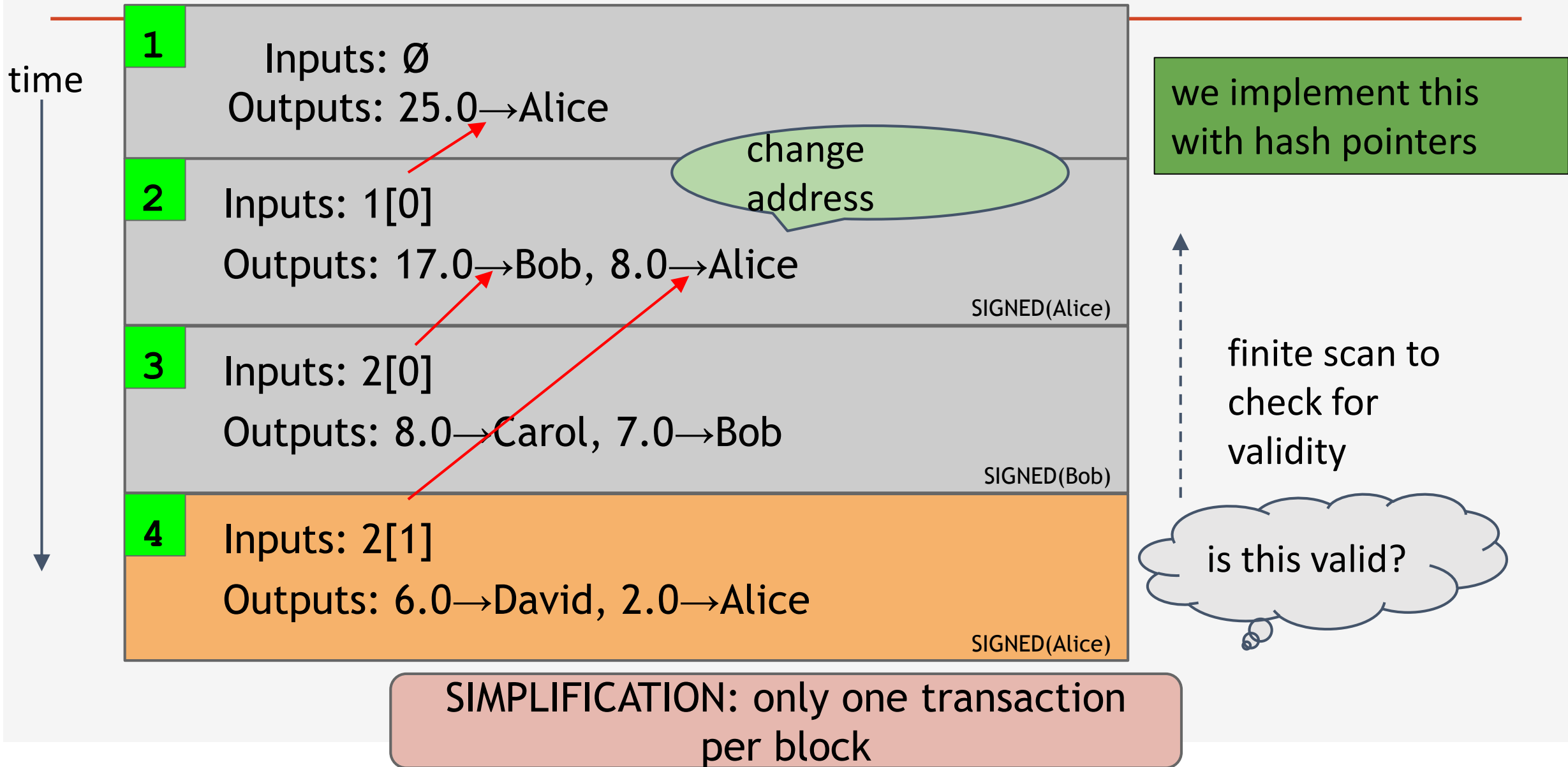


might need to  
scan  
backwards  
until genesis!

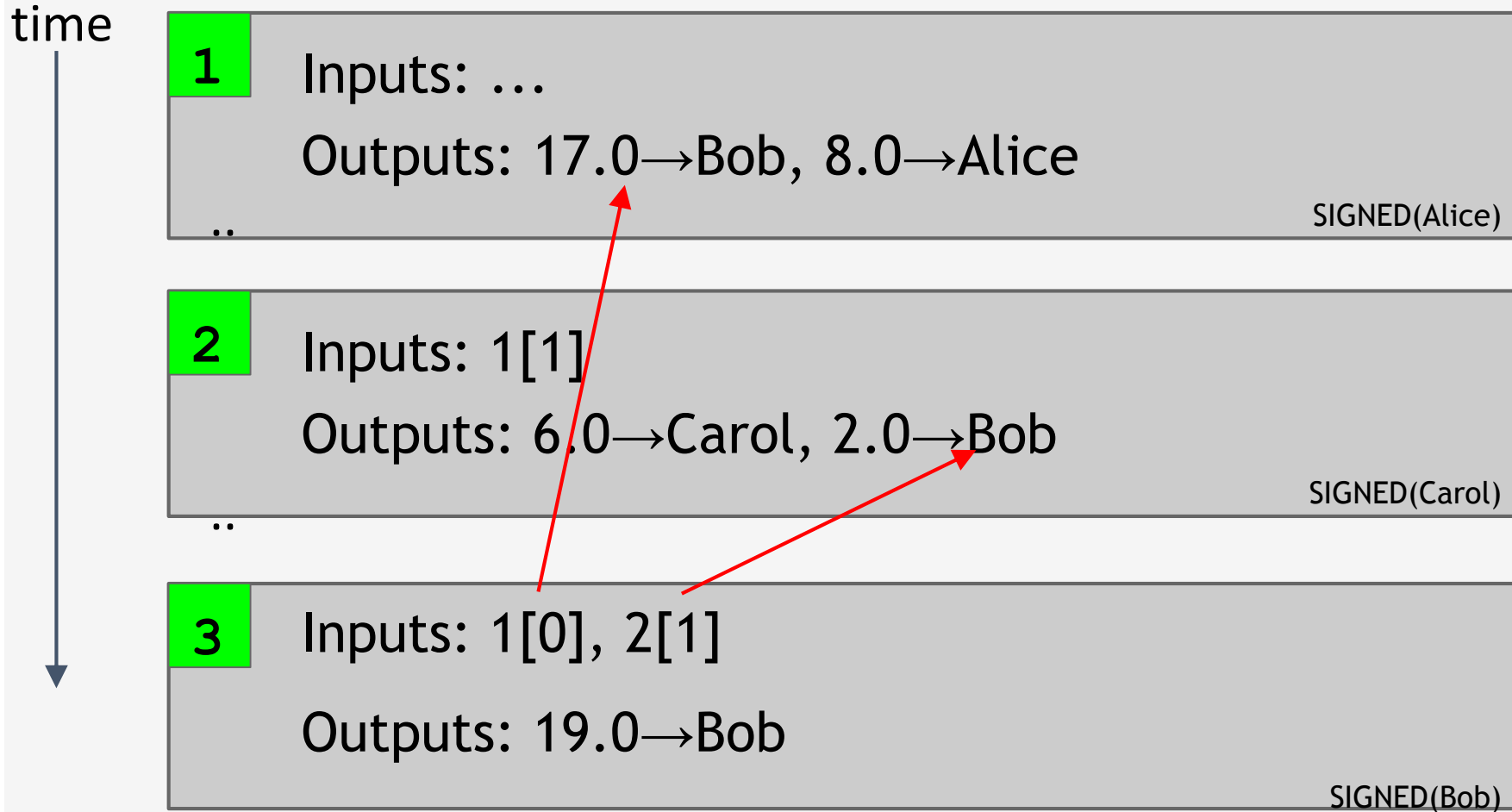
is this valid?

SIMPLIFICATION: only one transaction  
per block

# A transaction-based ledger (Bitcoin)



# Merging value

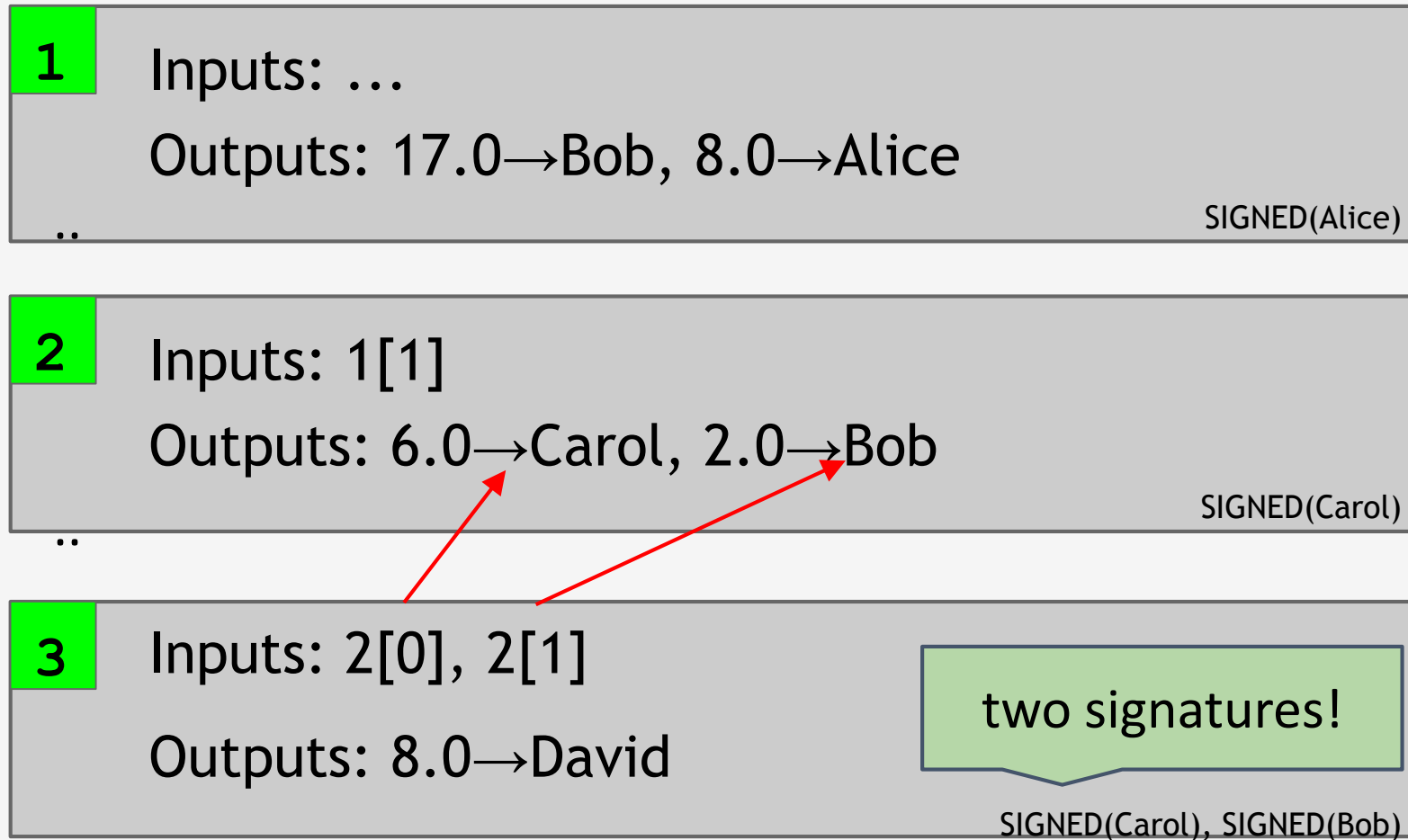


SIMPLIFICATION: only one transaction  
per block



# Joint payments

time  
↓



**SIMPLIFICATION: only one transaction  
per block**

# The real deal: a Bitcoin transaction



# The real deal: transaction metadata



# The real deal: transaction inputs

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

previous transaction {

signature {

(more inputs) {

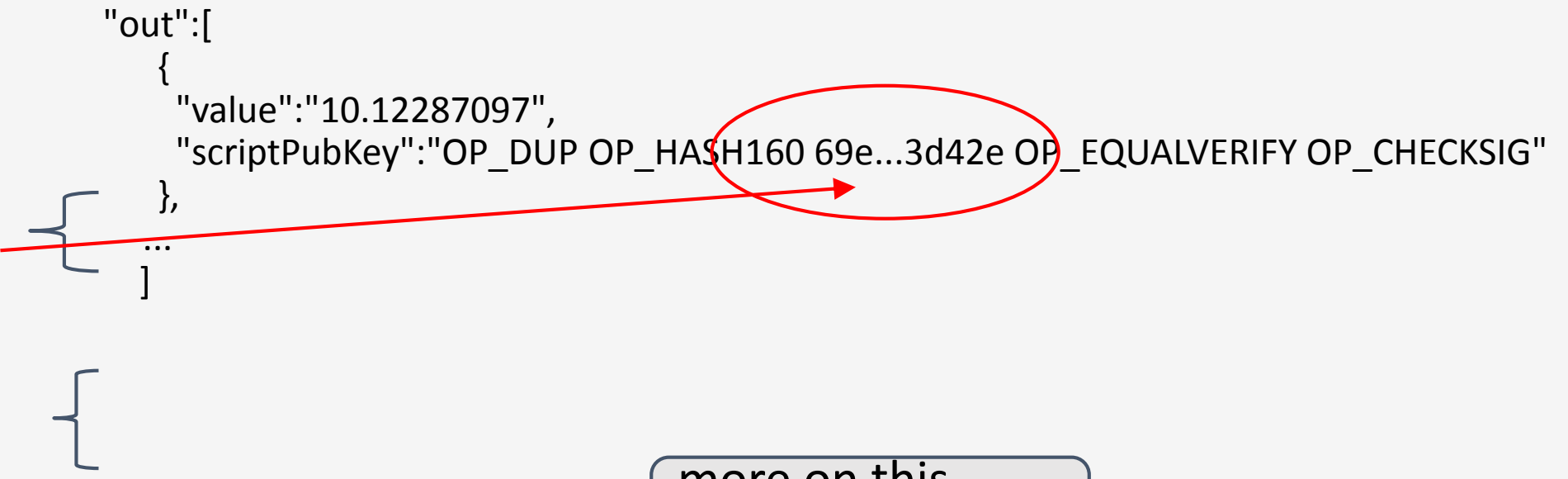
# The real deal: transaction outputs

output  
value

recipient  
address??

(more  
outputs)

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



more on this  
soon...

# Bitcoin scripts

# Output “addresses” are really *scripts*

OP\_DUP

OP\_HASH160

69e02e18...

OP\_EQUALVERIFY OP\_CHECKSIG

# Input “addresses” are *also* scripts

scriptSig

30440220...  
0467d2c9...

scriptPubKey

OP\_DUP  
OP\_HASH160  
69e02e18...  
OP\_EQUALVERIFY OP\_CHECKSIG

**TO VERIFY:** Concatenated script must execute completely  
with no errors

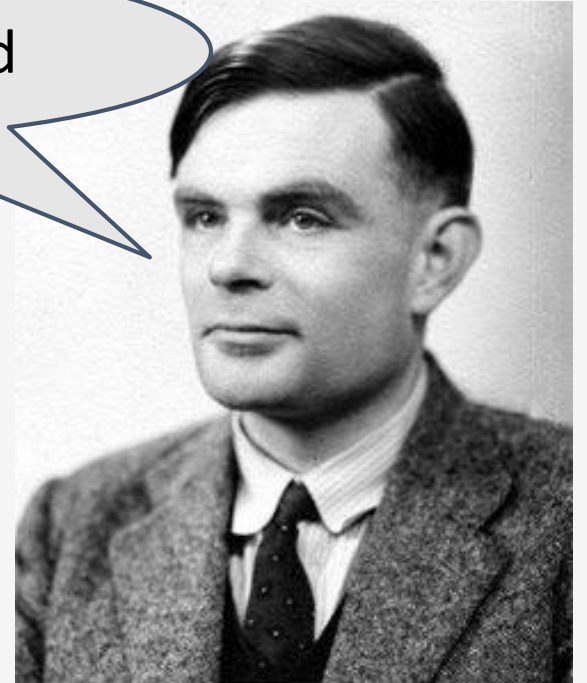


# Bitcoin scripting language (“Script”)

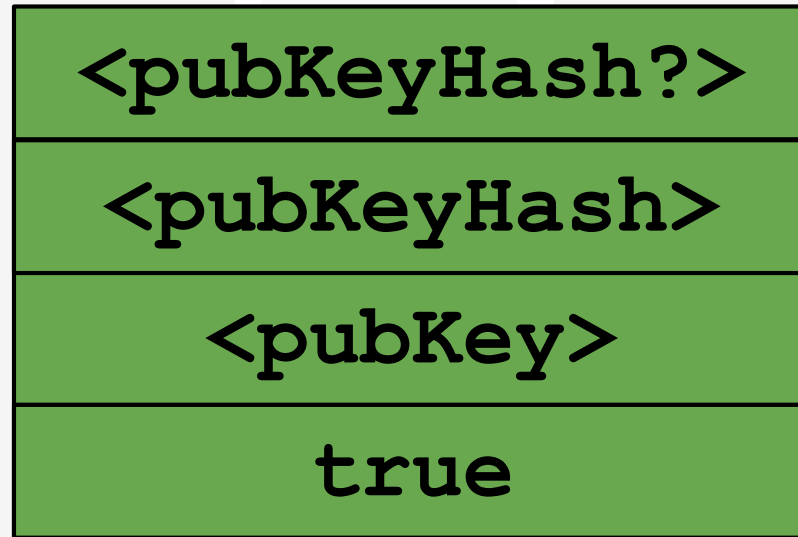
## Design goals

- Built for Bitcoin (inspired by Forth)
- Simple, compact
- Support for cryptography
- Stack-based
- Limits on time/memory
- No looping

I am not impressed



# Bitcoin script execution example



`<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



**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

More on this soon

# Proof-of-burn

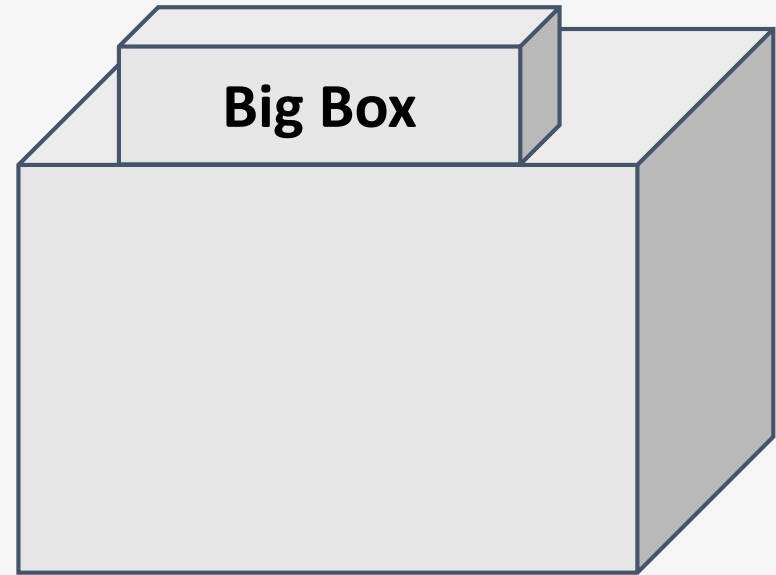
nothing's going to redeem that 😞

OP\_RETURN  
<arbitrary data>

# Should senders specify scripts?

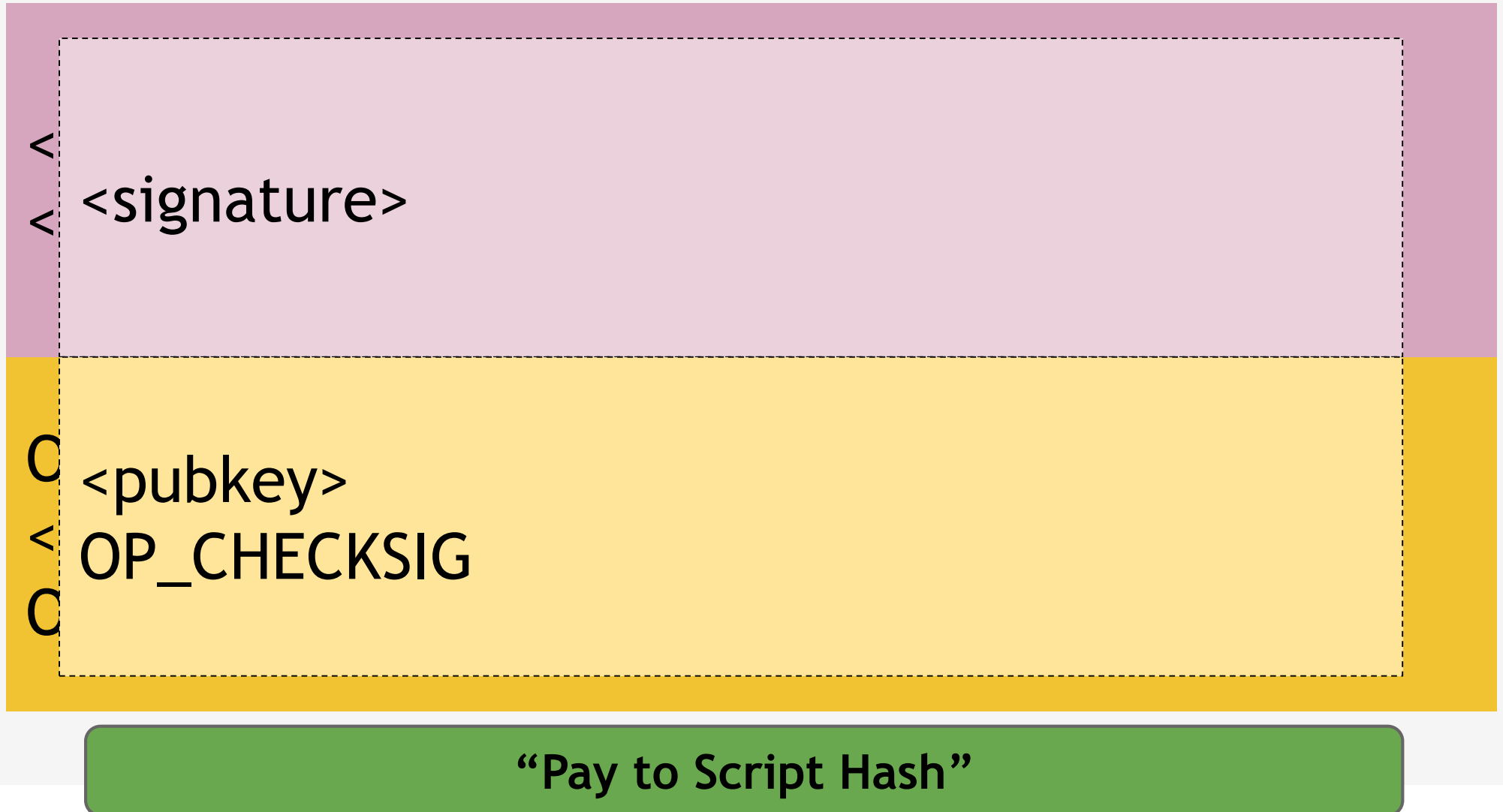


I'm ready to pay for my purchases!



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





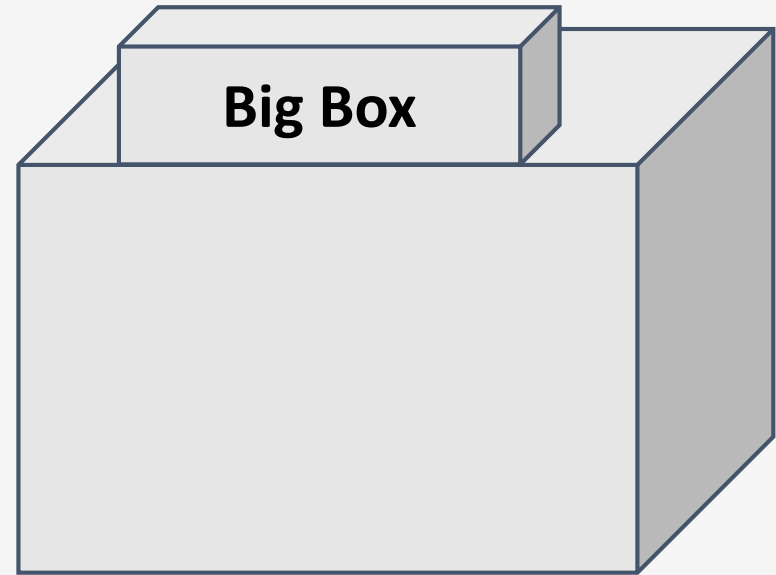
# Pay to script hash



I'm ready to pay for my  
purchases!

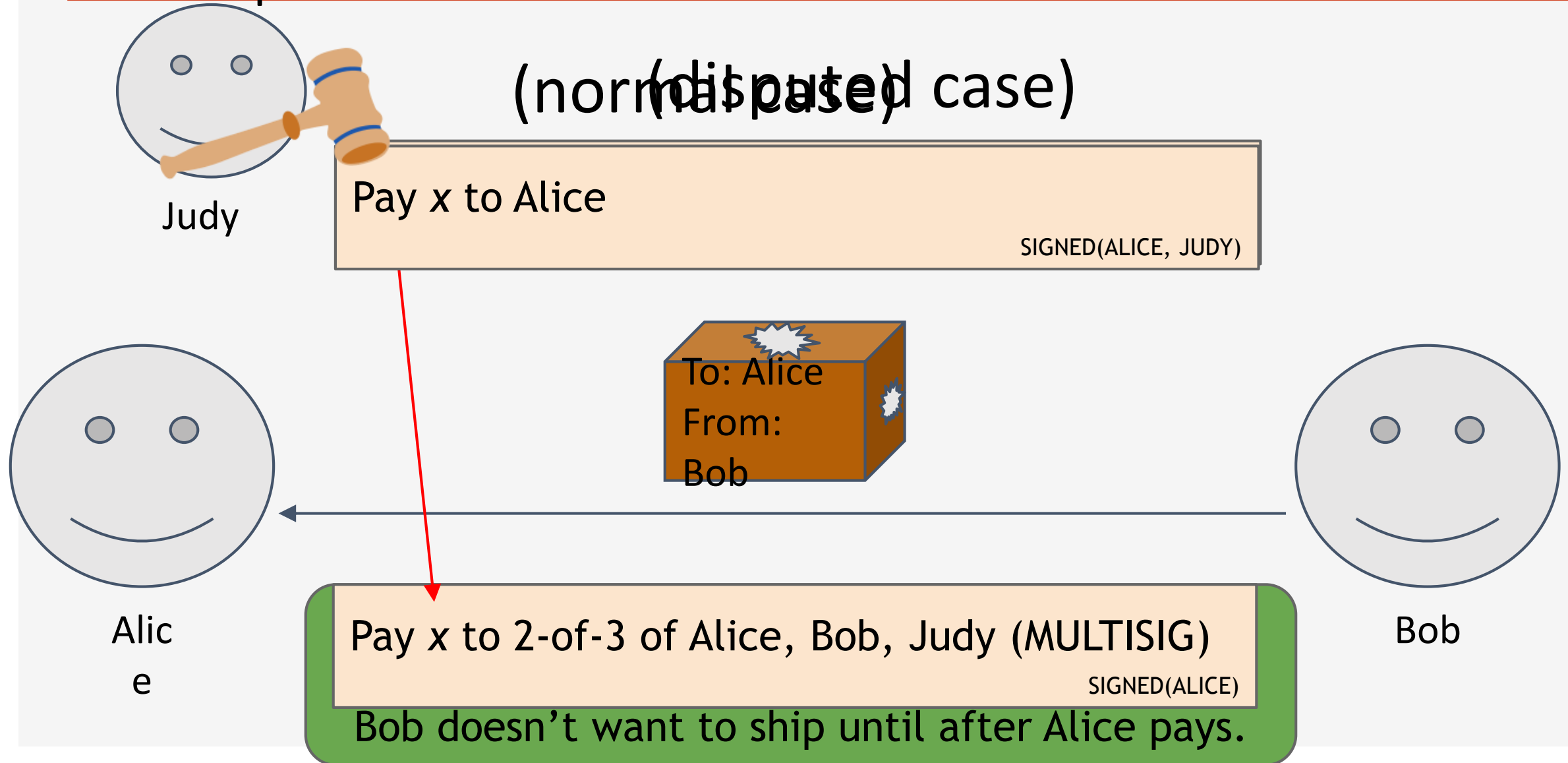


Great! Here's our address:  
0x3454

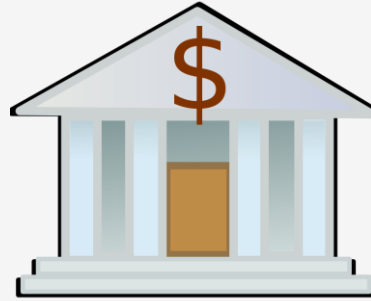
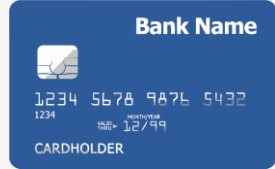


# Applications of Bitcoin scripts

# Example 1: Escrow transactions



# Example 2: Green addresses



Bank

004 days since last double spend!

It's me, Alice! Could you make out a green payment to Bob?



Alice

Pay  $x$  to Bob,  $y$  to Bank

No double spend

SIGNED(BANK)

Faraday cage



Bob

**PROBLEM:** Alice wants to pay Bob. Bob can't wait 6 verifications to guard against double-spends, or is offline completely.

# Example 3: Efficient micro-payments

What if Bob never signs??

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

SIGNED(ALICE) SIGNED(BOB)

⋮  
Alice demands a timed refund transaction before starting

Input:  $x$ ; Pay 100 to Alice, LOCK until time  $t$

SIGNED(ALICE) SIGNED(BOB)

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) \_\_\_\_\_

**PROBLEM:** Alice wants to pay Bob for each

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

SIGNED(ALICE)

all of these  
could be  
double-  
spends!

I'm  
done!

publish!

Alice

e

Bob

# 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

# More advanced scripts

---

- Multiplayer lotteries
- Hash pre-image challenges
- Coin-swapping protocols
  - Don't miss the lecture on anonymity!

**“Smart contracts”**

# Bitcoin blocks





# Bitcoin blocks

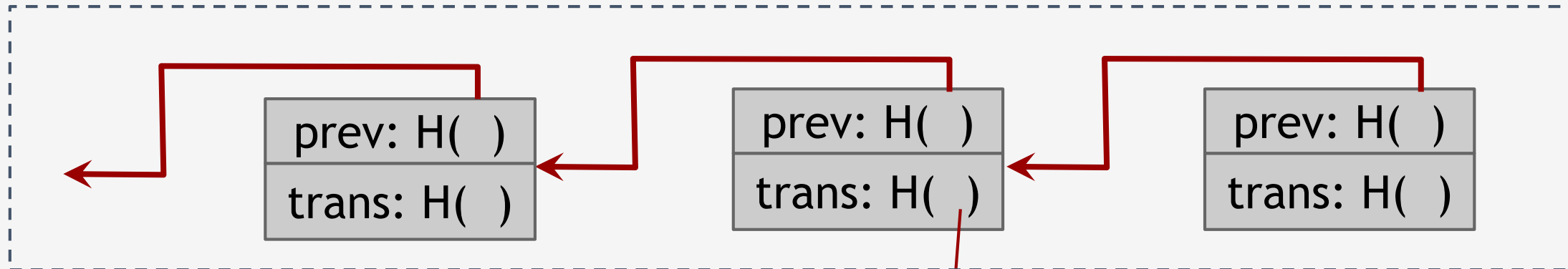
---

Why bundle transactions together?

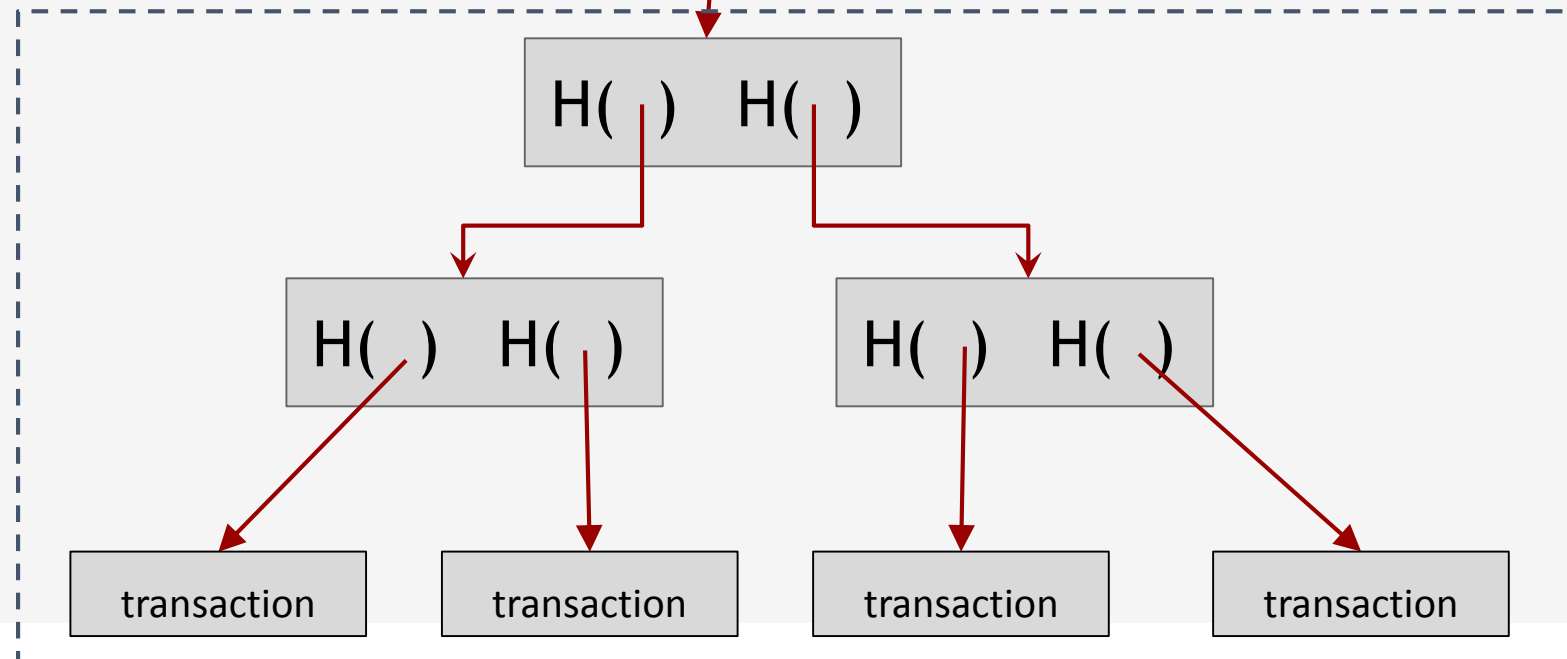
- Single unit of work for miners
- Limit length of hash-chain of blocks
  - Faster to verify history

# Bitcoin block structure

Hash chain of blocks



Hash tree (Merkle tree) of transactions in each block



# The real deal: a Bitcoin block

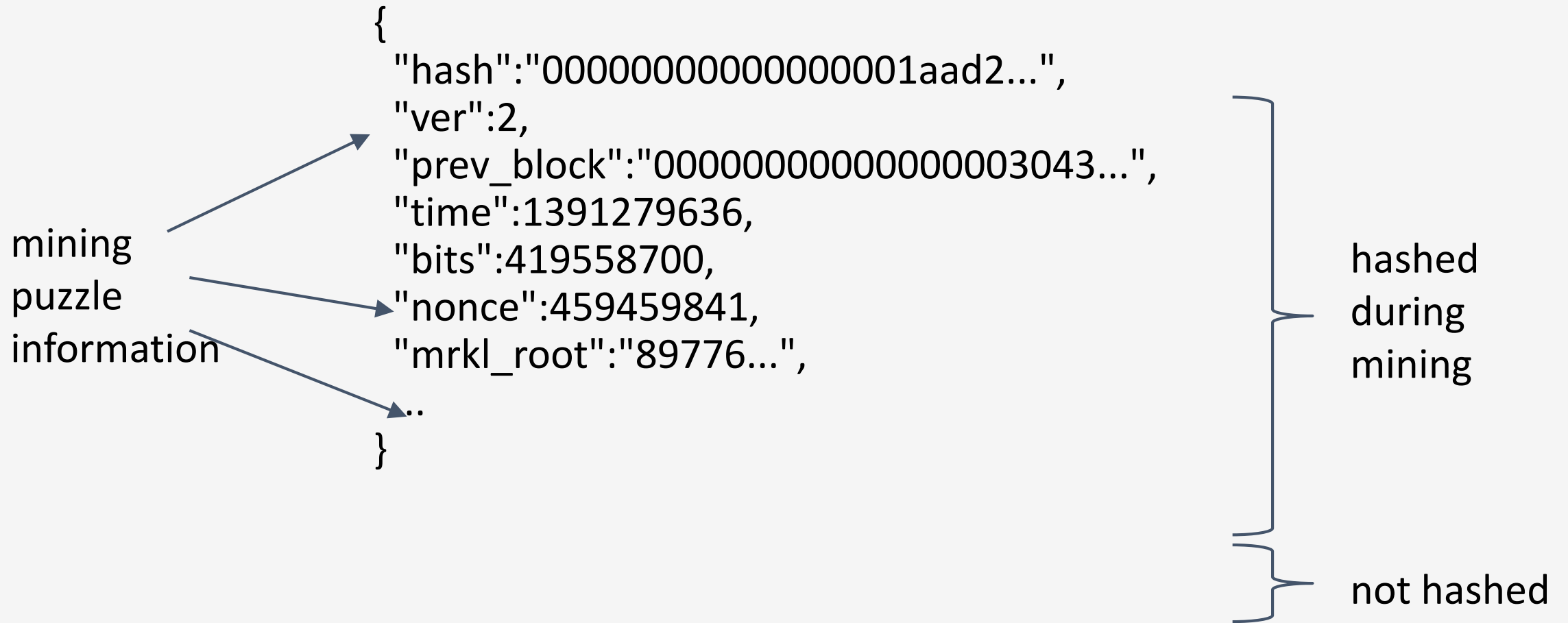
---

block  
header

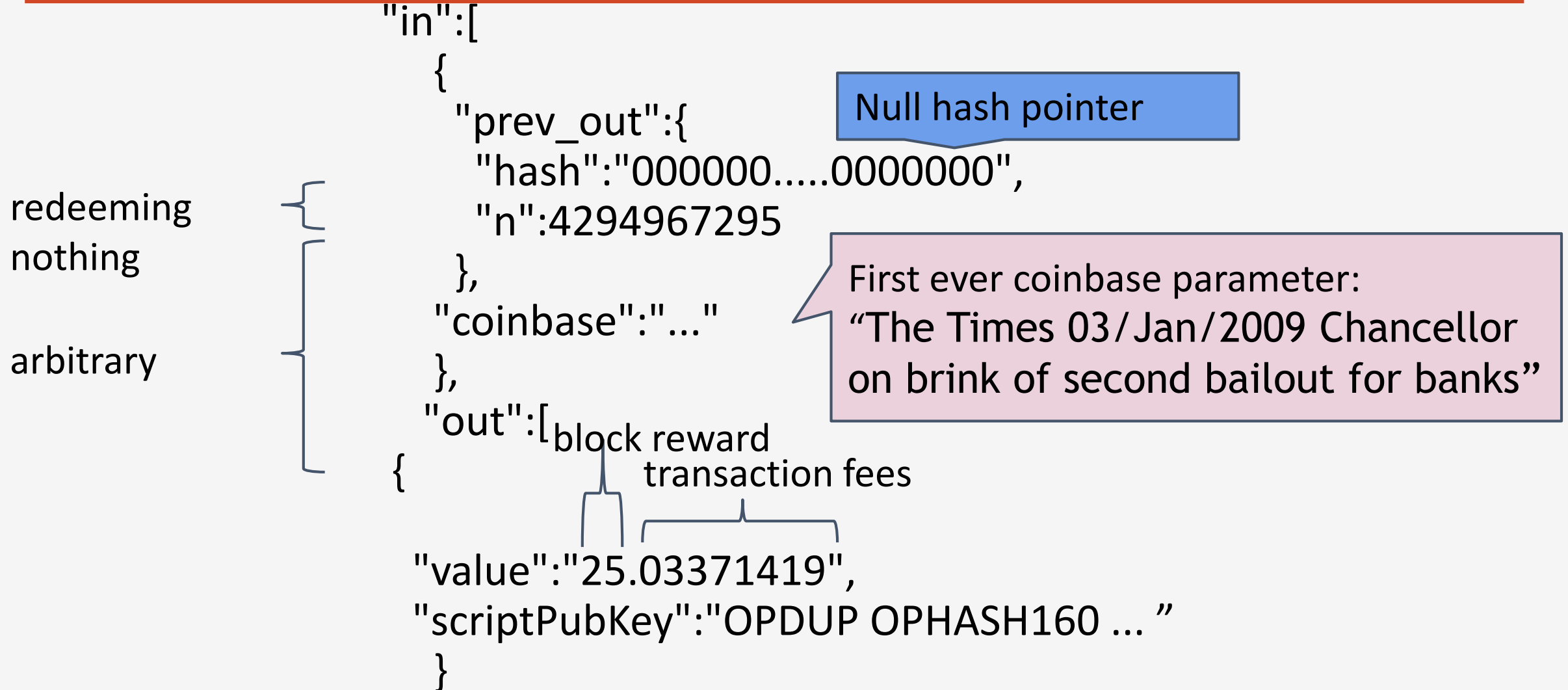
```
{  
  "hash": "0000000000000001aad2...",  
  "ver": 2,  
  "prev_block": "0000000000000003043...",  
  "time": 1391279636,  
  "bits": 419558700,  
  "nonce": 459459841,  
  "mrkl_root": "89776...",  
  "n_tx": 354,  
  "size": 181520,  
  "tx": [  
    ...  
  ],  
  "mrkl_tree": [  
    "6bd5eb25...",  
    ...  
    "89776cdb..."  
  ]  
}
```

transaction  
data

# The real deal: a Bitcoin block header



# The real deal: coinbase transaction



# See for yourself!

## Transaction View information about a bitcoin transaction

151b750d1f13e76d84e82b34b12688811b23a8e3119a1cba4b4810f9b0ef408d

1KryFUt9tXHvaoCYTNPbqpWPJKQ717YmL5



1KvrdrQ3oGqMAiDTMEYCcdDSnVaGNW2YZh  
1KryFUt9tXHvaoCYTNPbqpWPJKQ717YmL5

1.0194 BTC  
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

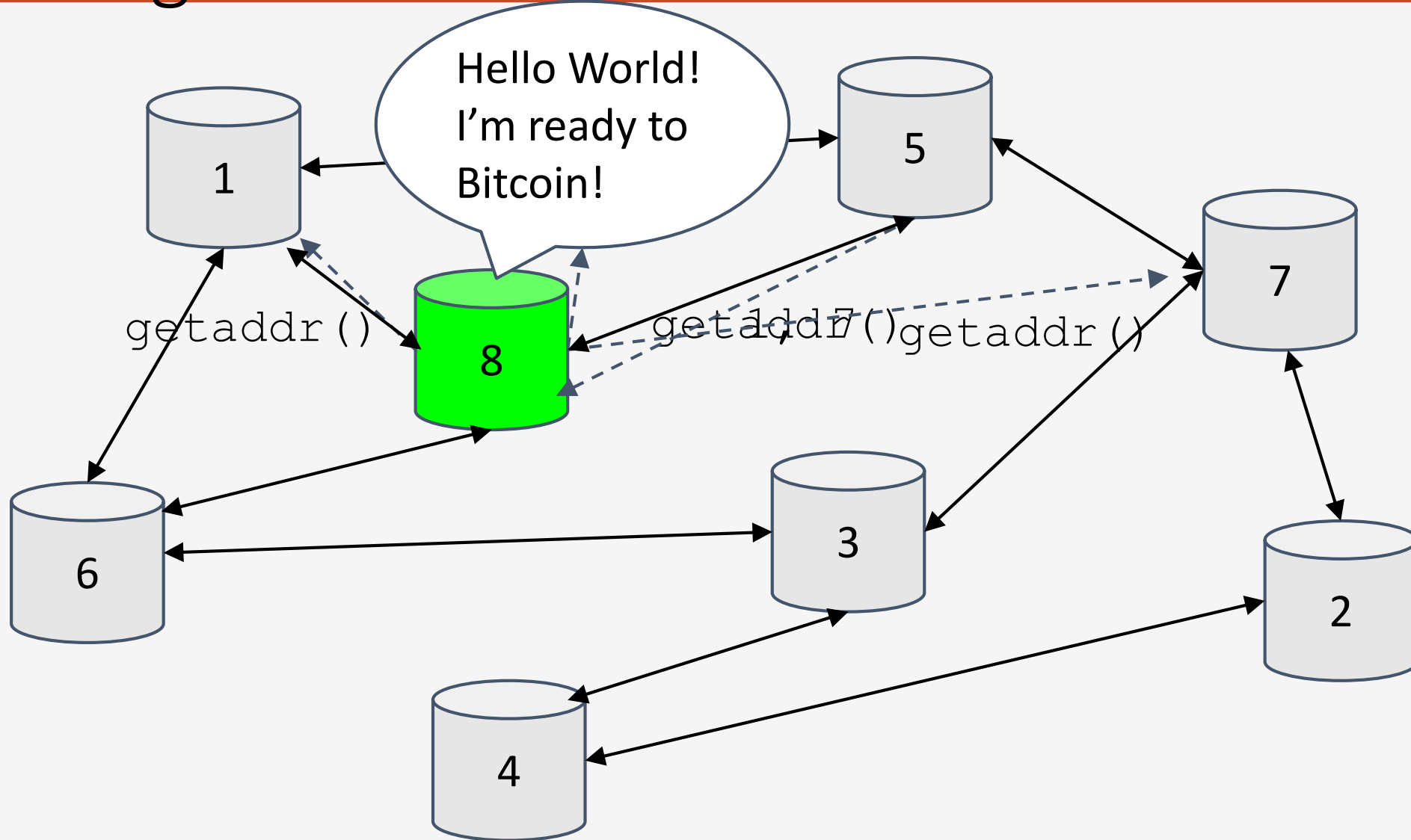
# Bitcoin P2P network

---

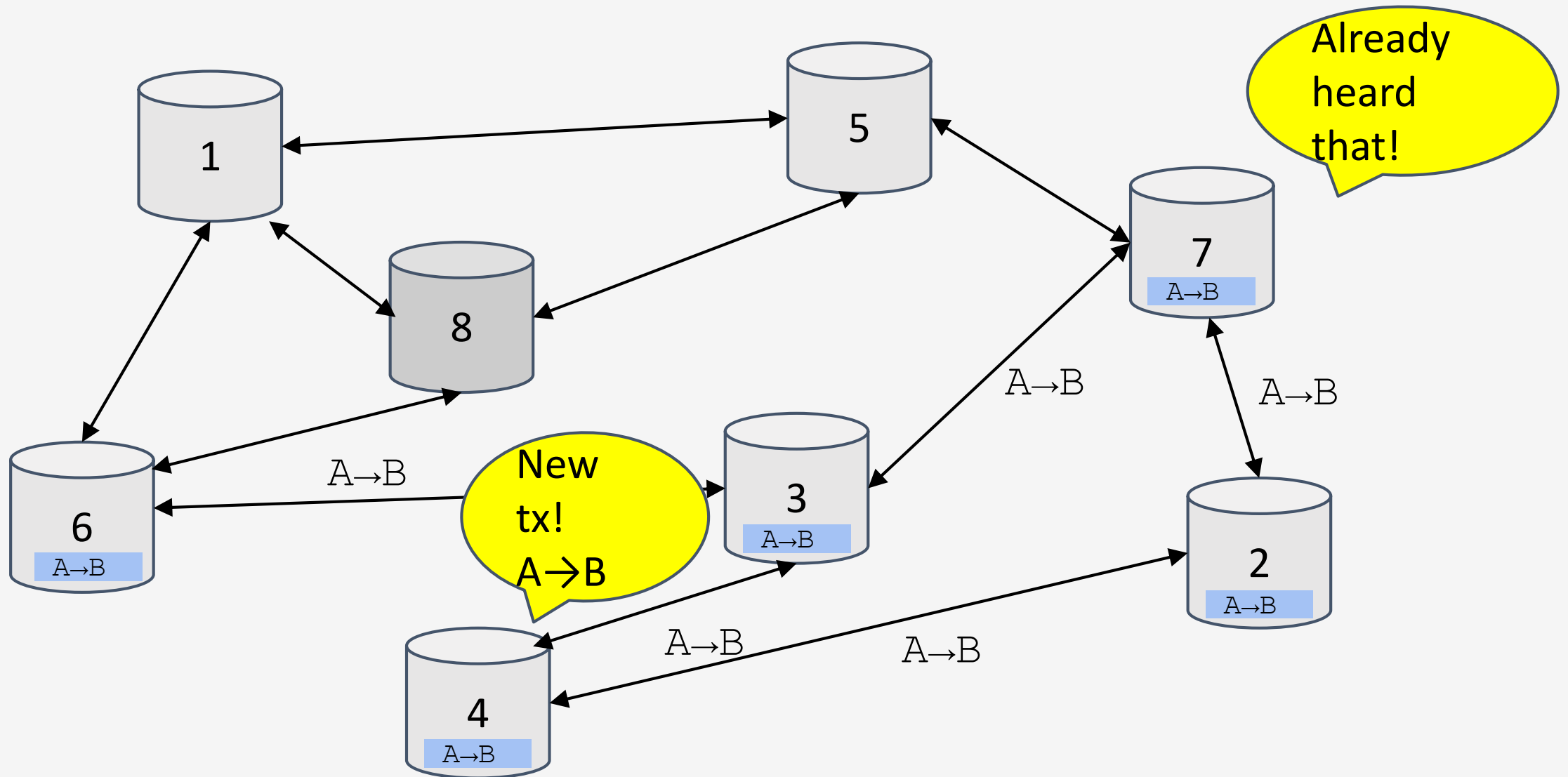
- Ad-hoc protocol (runs on TCP port 8333)
- Ad-hoc network with random topology
- All nodes are equal
- New nodes can join at any time
- Forget non-responding nodes after 3 hr



# Joining the Bitcoin P2P network

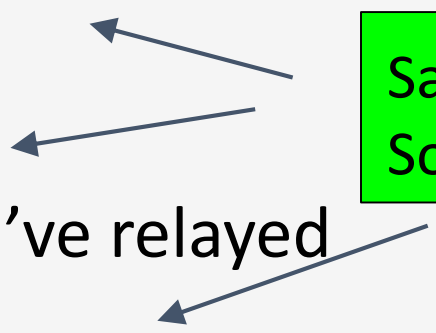


# Transaction propagation (flooding)



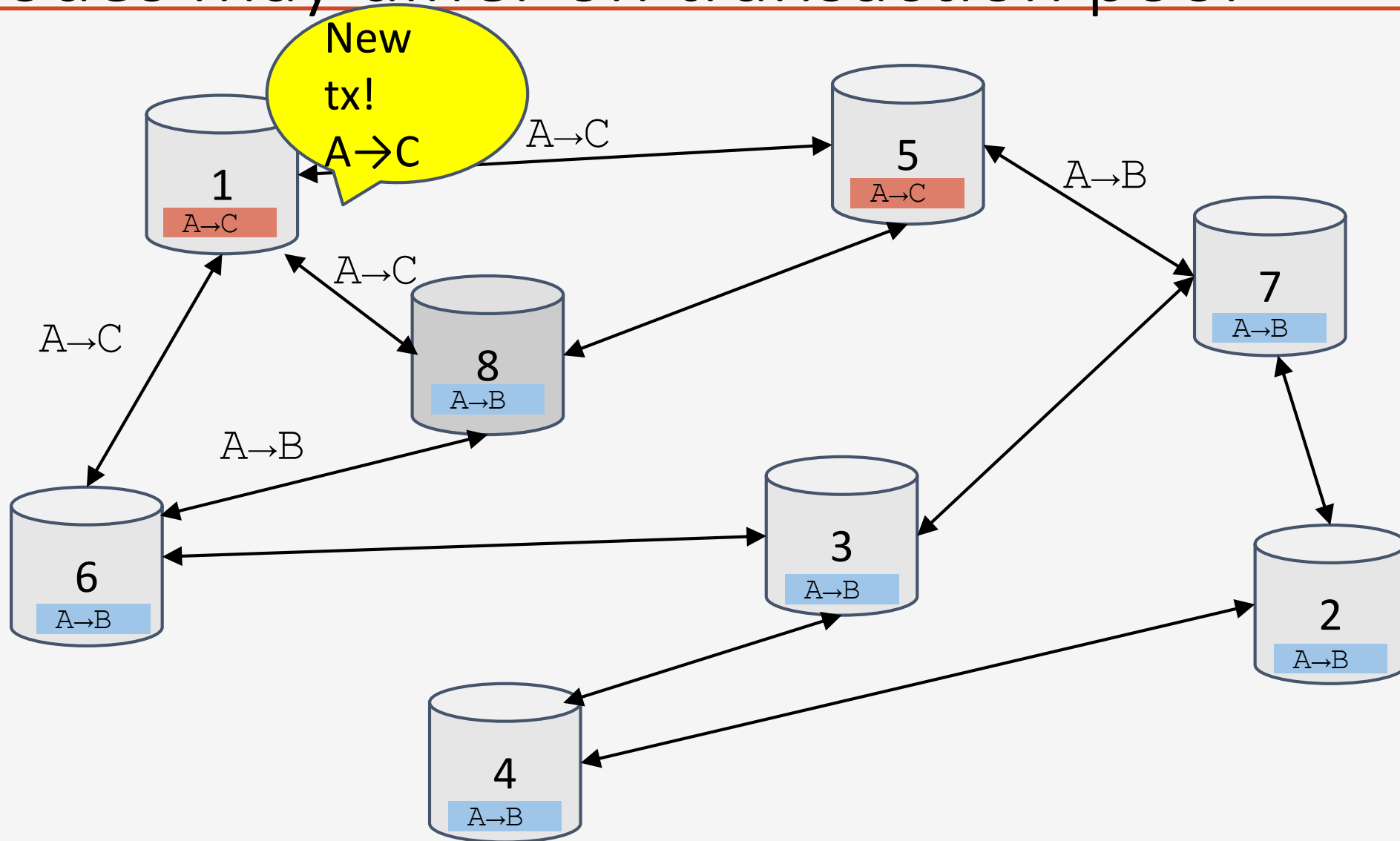
# Should I relay a proposed transaction?

- 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



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

# 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!

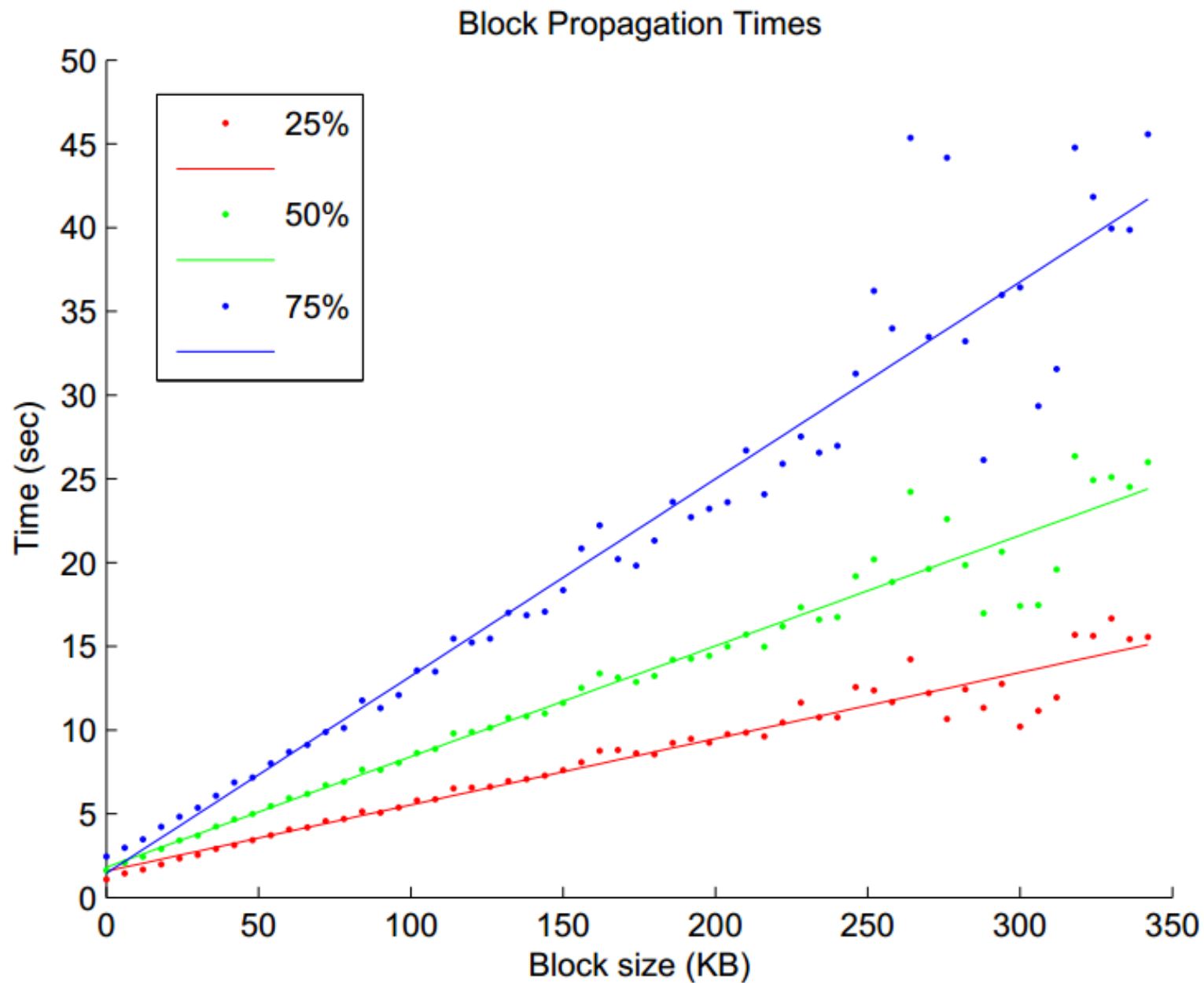
# Block propagation nearly identical

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



Sanity check  
Also may be ignored...



Source: Yonatan Sompolinsky and Aviv Zohar: "Accelerating Bitcoin's Transaction Processing" 2014

# How big is the network?

---

- Impossible 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!



# Bitcoin full node distribution

---

RANK	COUNTRY	NODES
1	<a href="#">United States</a>	2466 (24.32%)
2	<a href="#">Germany</a>	1936 (19.09%)
3	<a href="#">France</a>	674 (6.65%)
4	<a href="#">Netherlands</a>	484 (4.77%)
5	<a href="#">China</a>	402 (3.96%)
6	<a href="#">Canada</a>	402 (3.96%)
7	<a href="#">United Kingdom</a>	351 (3.46%)
8	<a href="#">Singapore</a>	312 (3.08%)
9	<a href="#">Russian Federation</a>	270 (2.66%)
10	<a href="#">Japan</a>	248 (2.45%)
	<a href="#">More (102)</a>	

TOTAL 10140 NODES AT 5 PM ON JAN 23, 2019

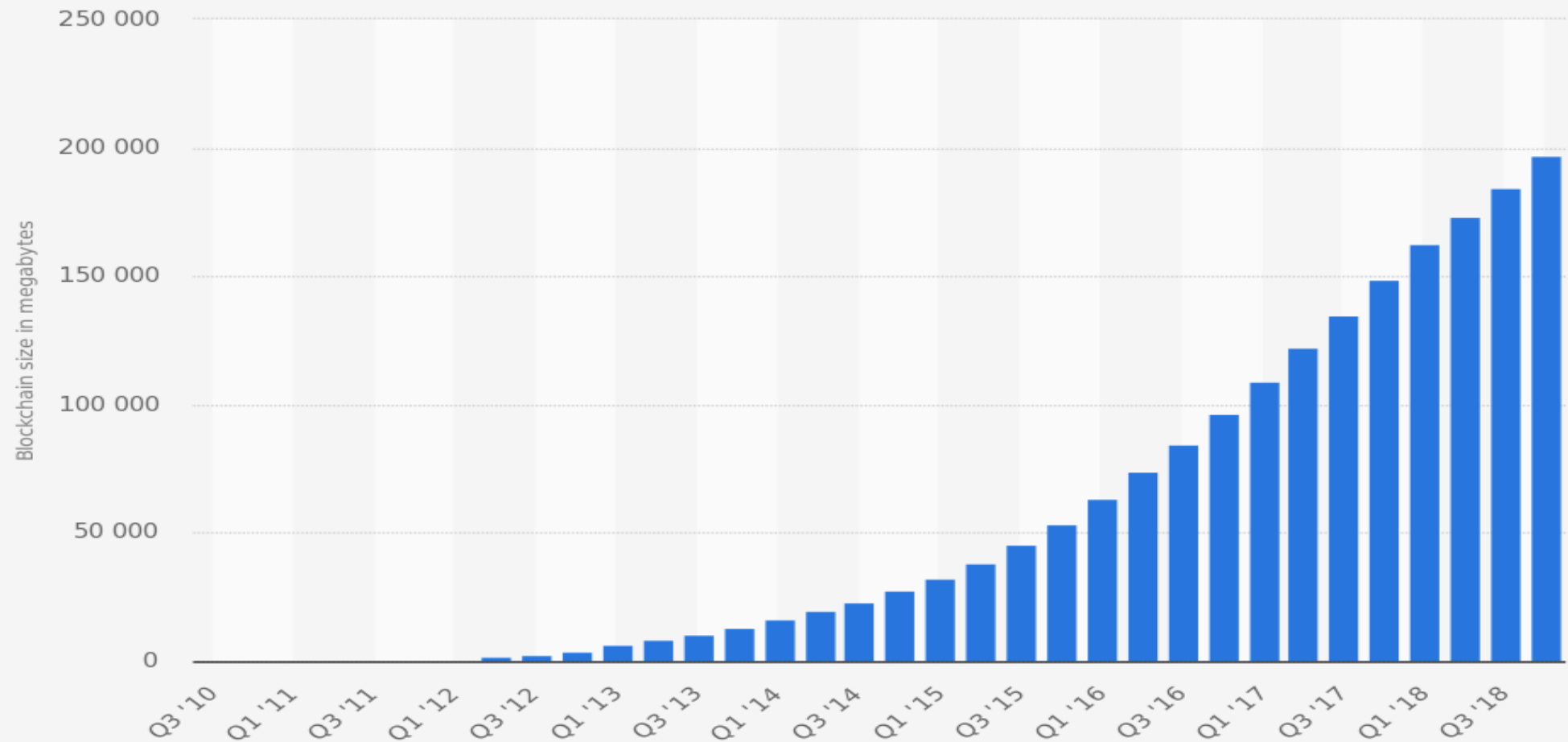
# Fully-validating nodes

---

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

# Storage costs

Size of the Bitcoin blockchain from 2010 to 2019, by quarter (in megabytes)



Source  
Blockchain  
© Statista 2019

Additional Information:  
Worldwide; Blockchain; 2010 to 2019

# Tracking the UTXO set

---

- Unspent Transaction Output
  - Everything else can be stored on disk
- Currently ~61.7 M UTXOs
  - Out of 375 M transactions (as of Jan 2019)

# 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

1000x cost savings! (200 GB->200MB)

# Software diversity

---

- About 90% of nodes run “Core Bitcoin” (C++)
  - Some are out of date versions
- Other implementations running successfully
  - BitcoinJ (Java)
  - Libbitcoin (C++)
  - btcd (Go)
- “Original Satoshi client”

# Limitations & improvements

# Hard-coded limits in Bitcoin

---

- 10 min. average creation time per block
- 1 M bytes in a block (post SegWit – it is more)
- 20,000 signature operations per block
- 100 M *satoshis* per bitcoin
- 21M 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) - post SegWit is slightly different
- >250 bytes/transaction
- 7 transactions/sec 😞

Compare to:

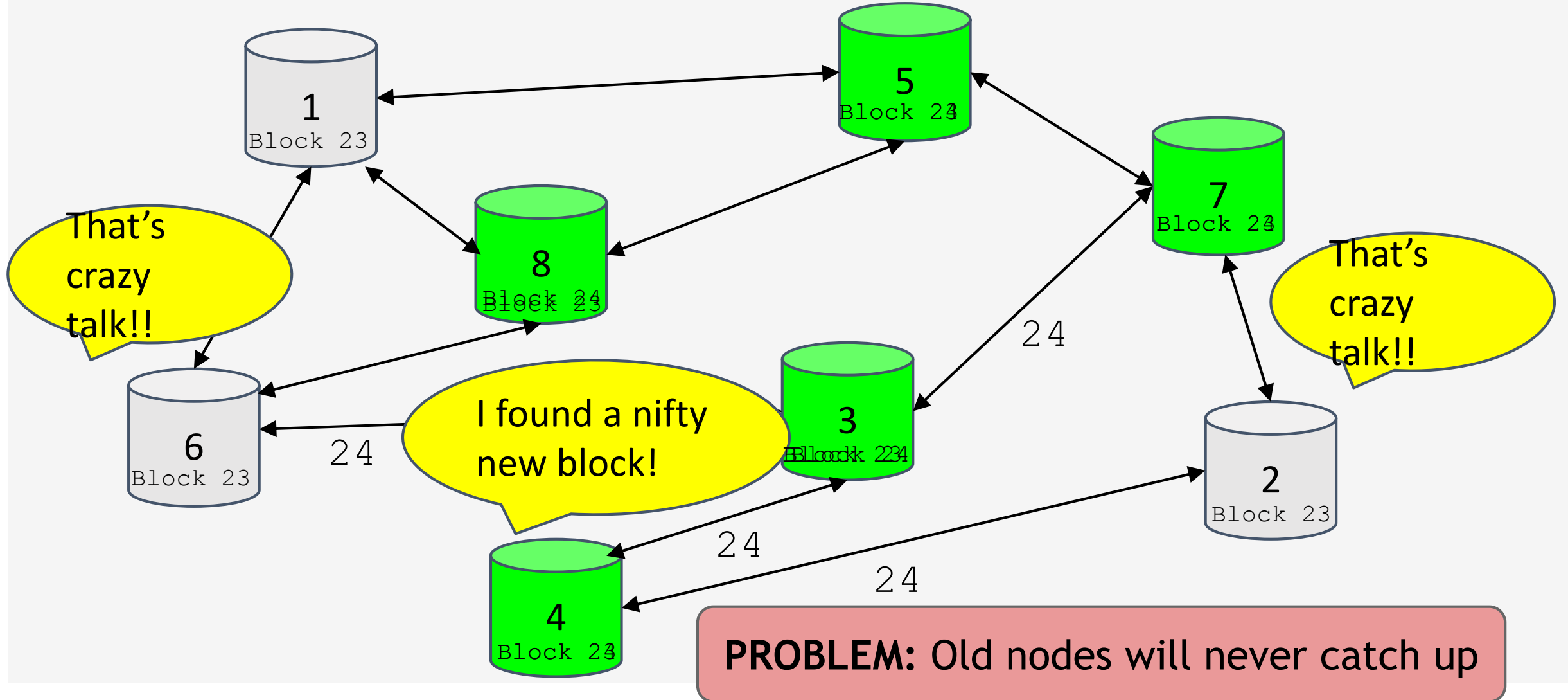
- VISA: 2,000-10,000 transactions/sec
- PayPal: 50-100 transaction/sec

# Cryptographic limits in Bitcoin

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

Crypto primitives might break by 2040...

# “Hard-forking” changes to Bitcoin



# 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 UTXO tree in each block

# Hard forks

---

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