

# Lecture 3

Mechanics of Bitcoin

# Recap: Bitcoin consensus

Bitcoin consensus gives us:

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

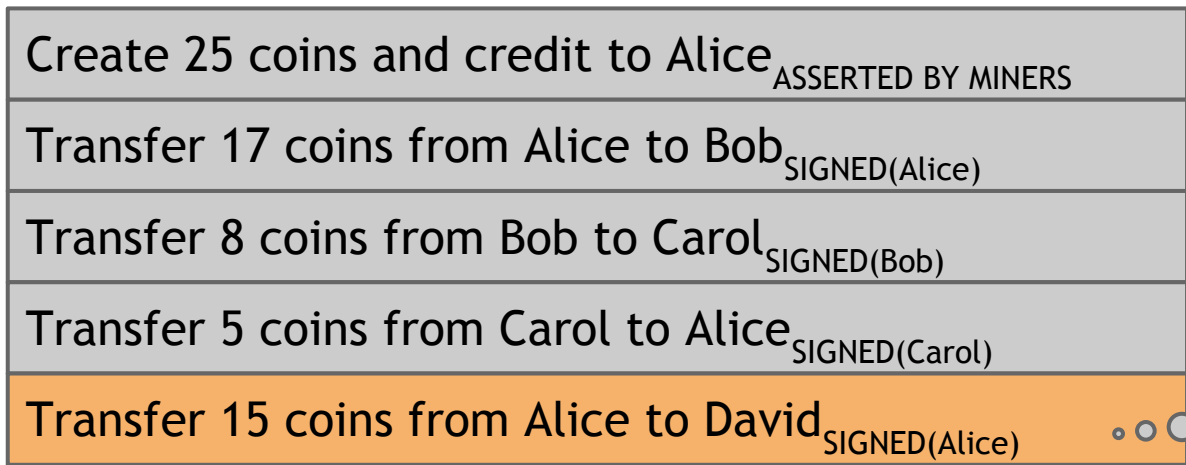
*assuming a currency exists to motivate miners!*

# Lecture 3.1:

## Bitcoin transactions

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

time

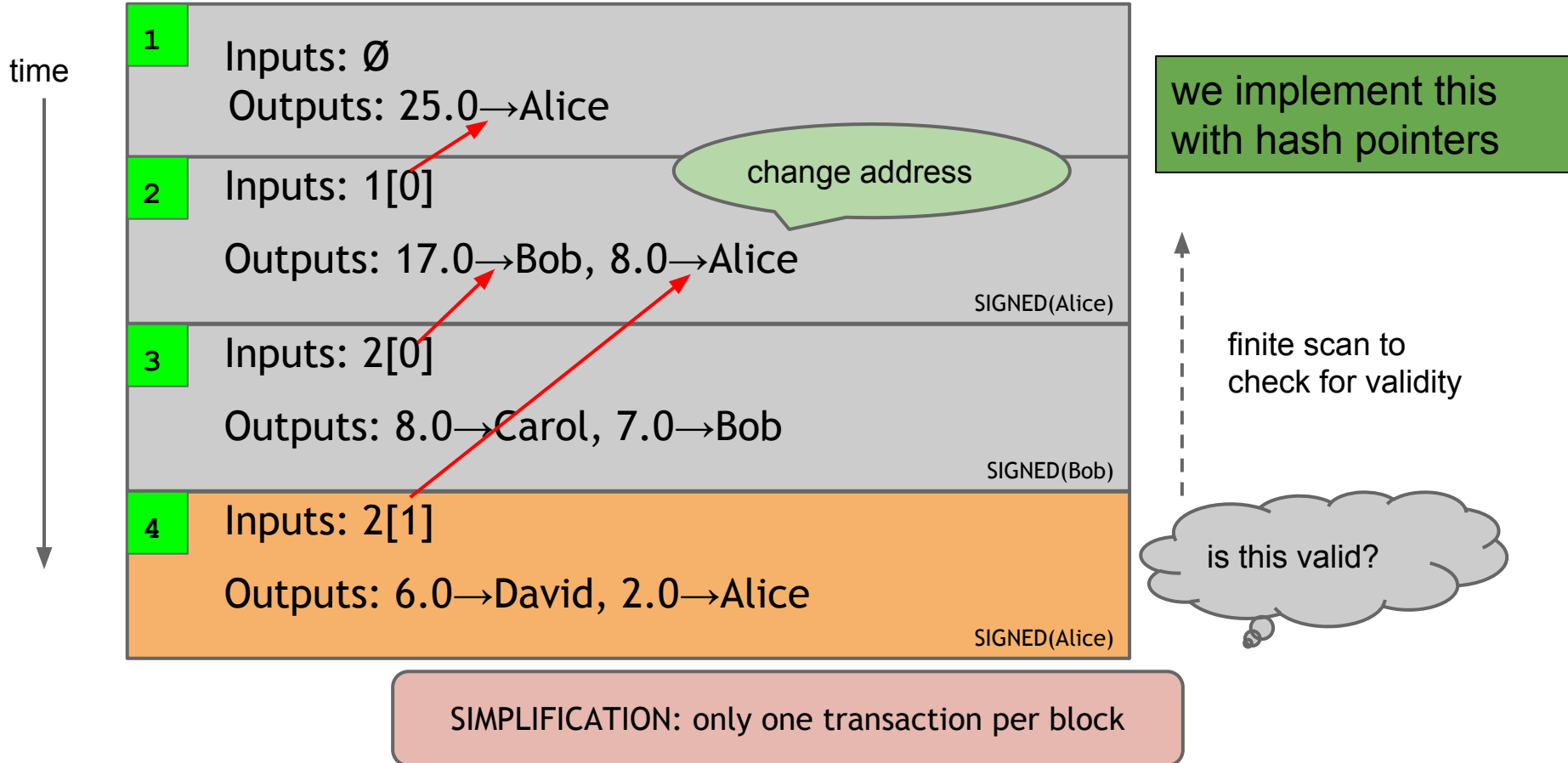


might need to  
scan backwards  
until genesis!

is this valid?

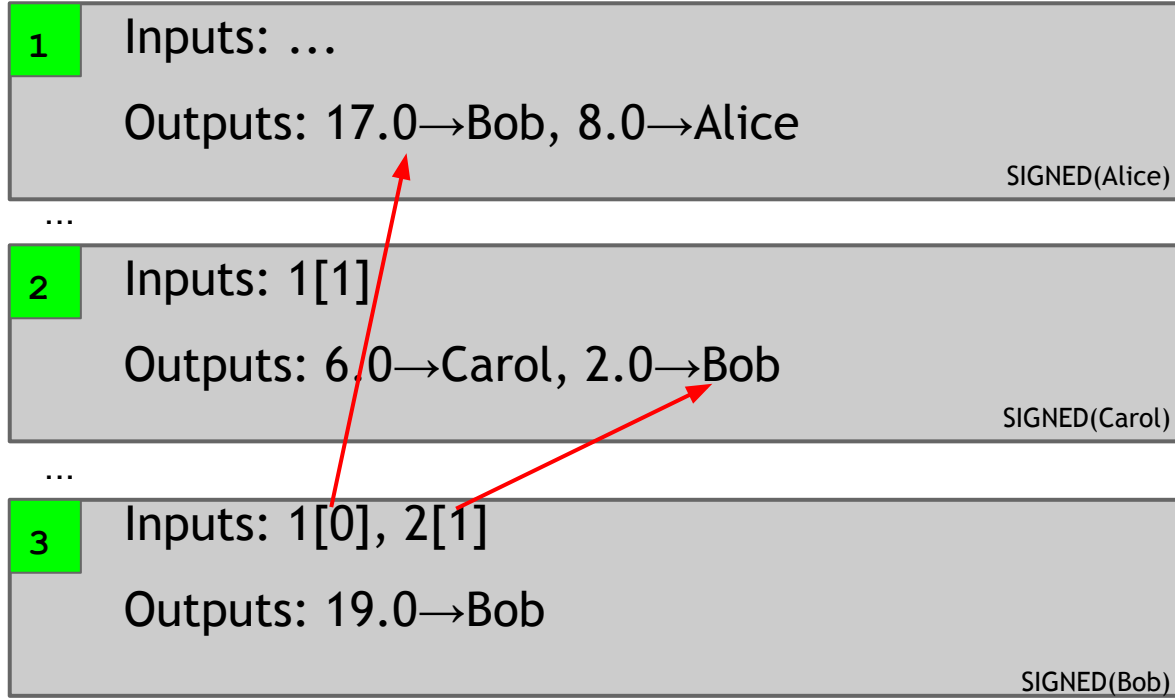
SIMPLIFICATION: only one transaction per block

# A transaction-based ledger (Bitcoin)



# Merging value

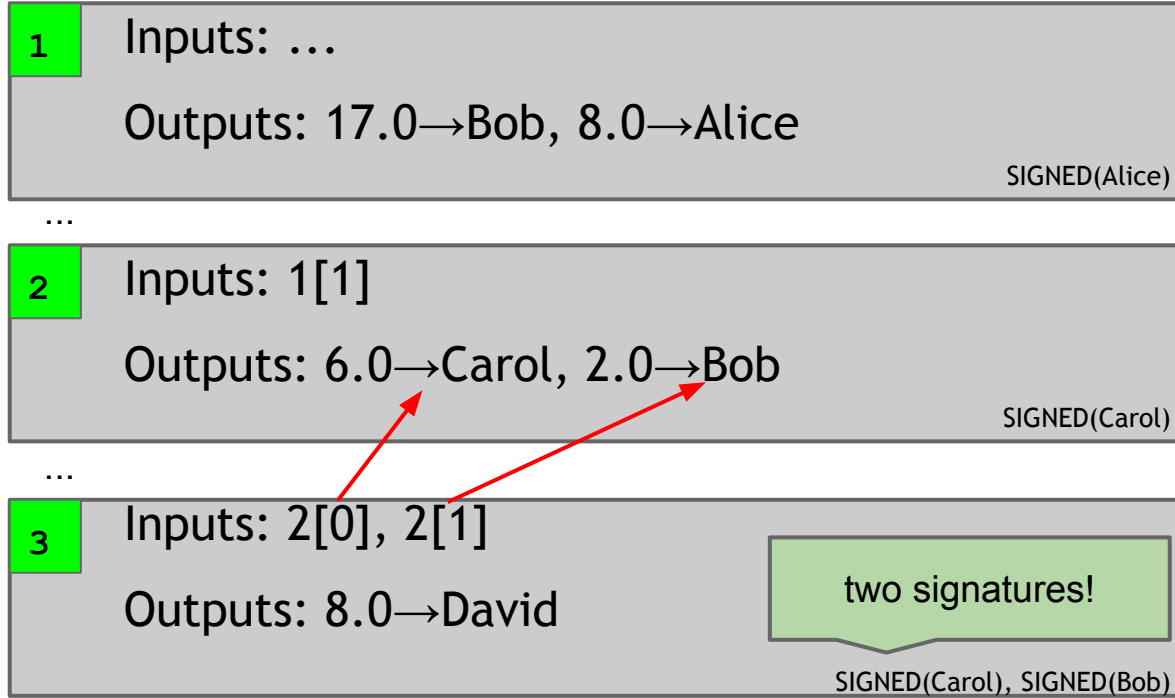
time



SIMPLIFICATION: only one transaction per block

# Joint payments

time



SIMPLIFICATION: only one transaction per block

# 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": "7508e6ab259b4dfd0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",  
        "n": 0  
      },  
      "scriptSig": "3f3a4ce81...."  
    }  
  ],  
  "out": [  
    {  
      "value": "10.12287097",  
      "scriptPubKey": "OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"  
    }  
  ]  
}
```

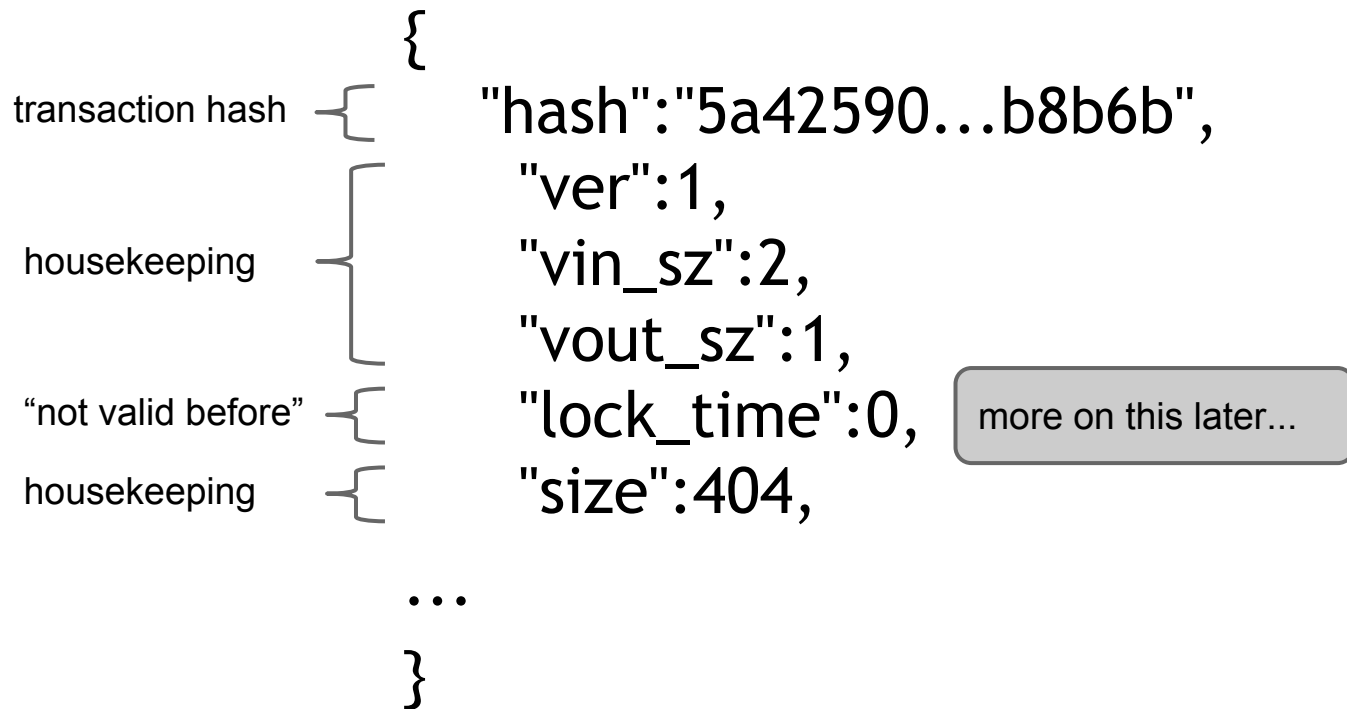
metadata

input(s)

output(s)



# The real deal: transaction metadata



# The real deal: transaction inputs

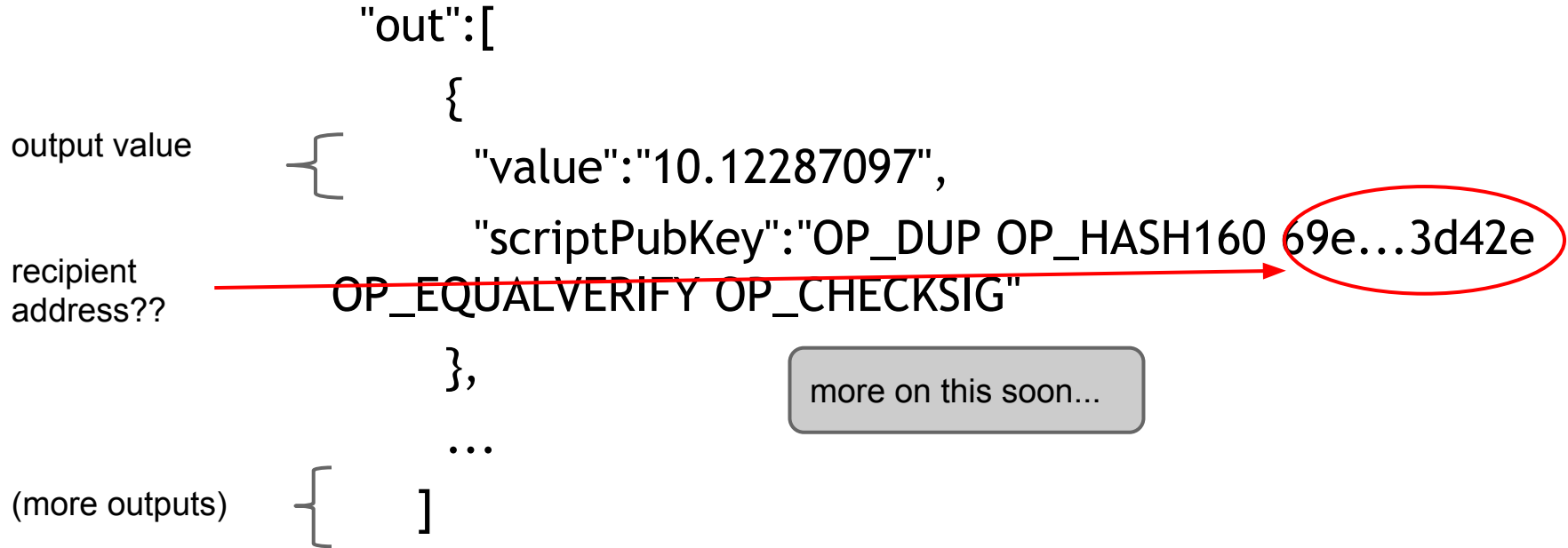
```
    "in":[  
      {  
        "prev_out":{  
          "hash":"3be4...80260",  
          "n":0  
        },  
        "scriptSig":"30440....3f3a4ce81"  
      },  
      ...  
    ],
```

previous transaction {

signature {

(more inputs) {

# The real deal: transaction outputs



# Lecture 3.2:

## Bitcoin scripts

# Output “addresses” are really *scripts*

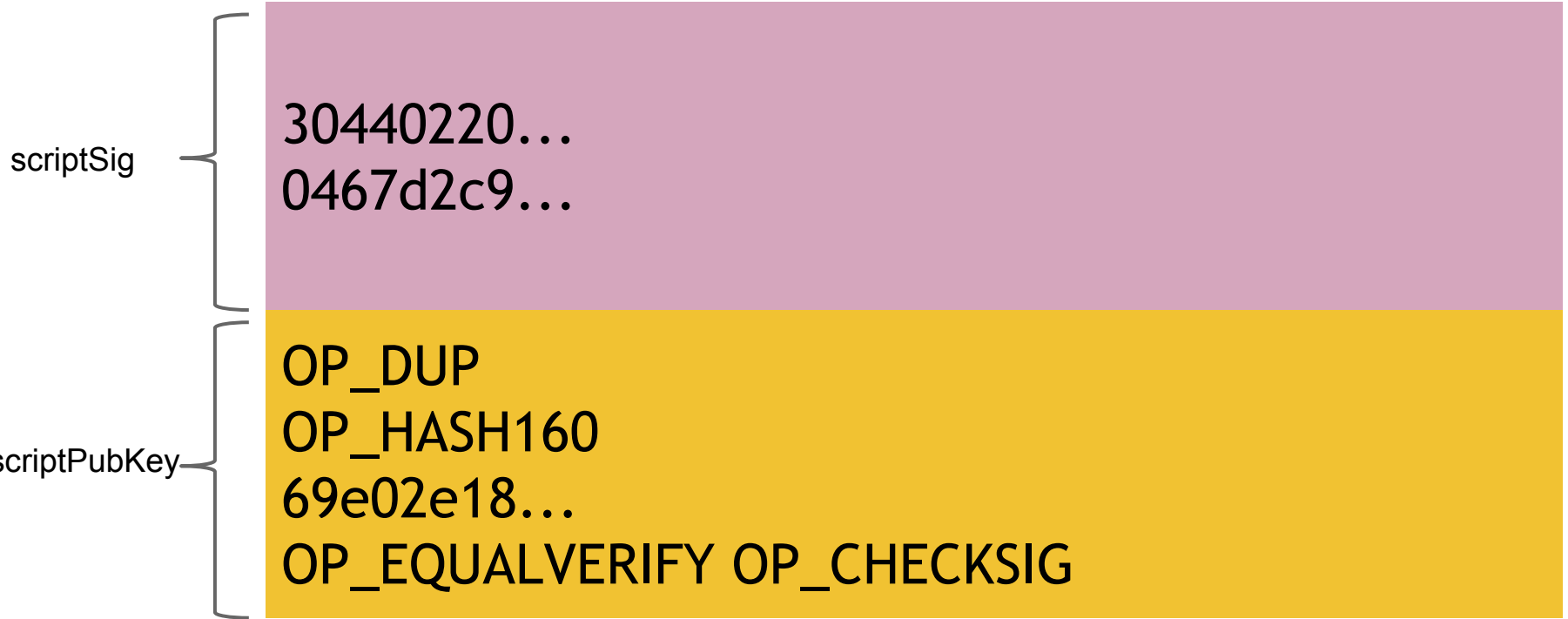
OP\_DUP

OP\_HASH160

69e02e18...

OP\_EQUALVERIFY OP\_CHECKSIG

# Input “addresses” are *also* scripts



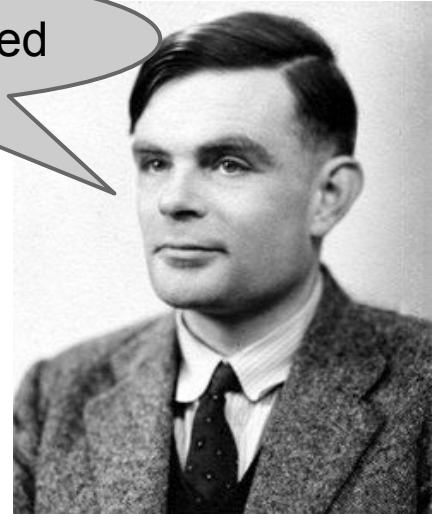
TO VERIFY: Concatenated script must execute completely with no errors

# Bitcoin scripting language (“Script”)

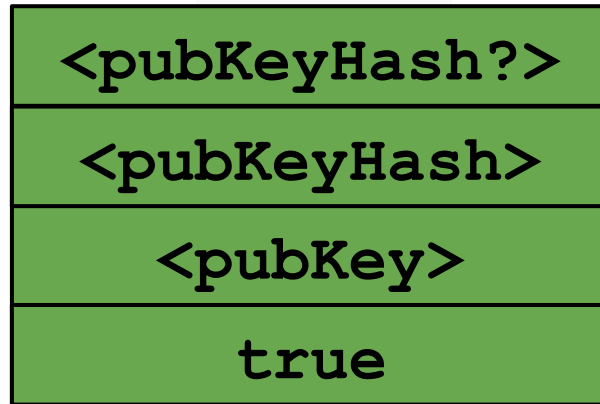
## Design goals

- 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



|               |
|---------------|
| <pubKeyHash?> |
| <pubKeyHash>  |
| <pubKey>      |
| 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



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

# Bitcoin scripts in practice (as of 2014)

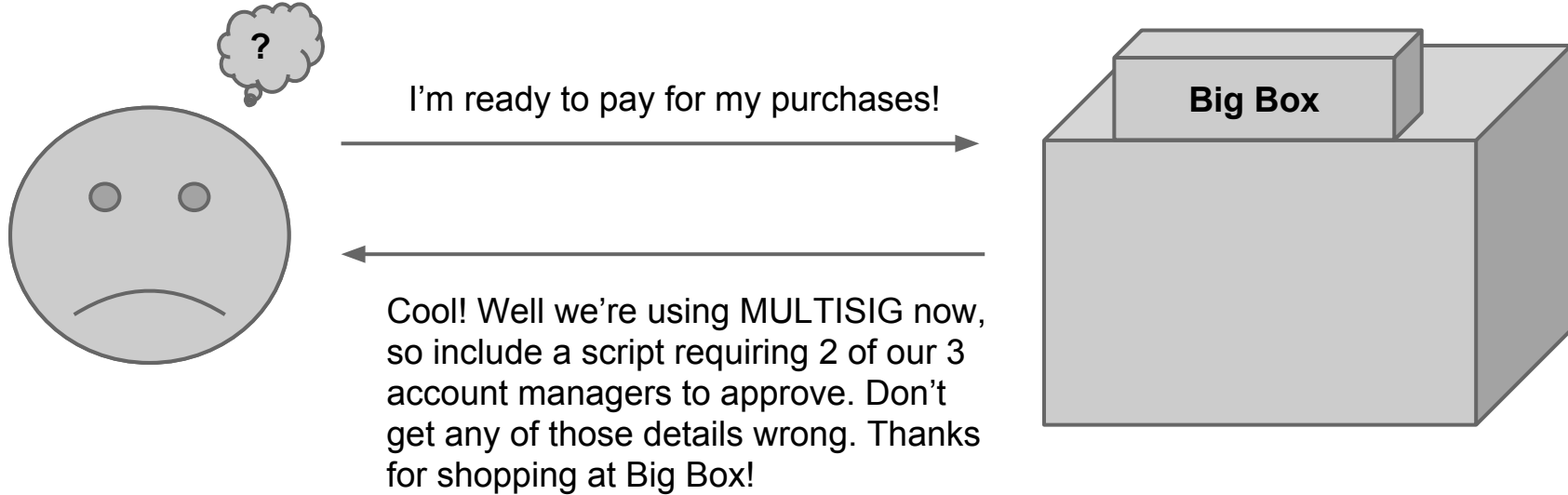
- Most nodes whitelist known scripts
- 99.9% are simple signature checks
- ~0.01% are MULTISIG 
- ~0.01% are **Pay-to-Script-Hash**
- Remainder are errors, proof-of-burn

# Proof-of-burn

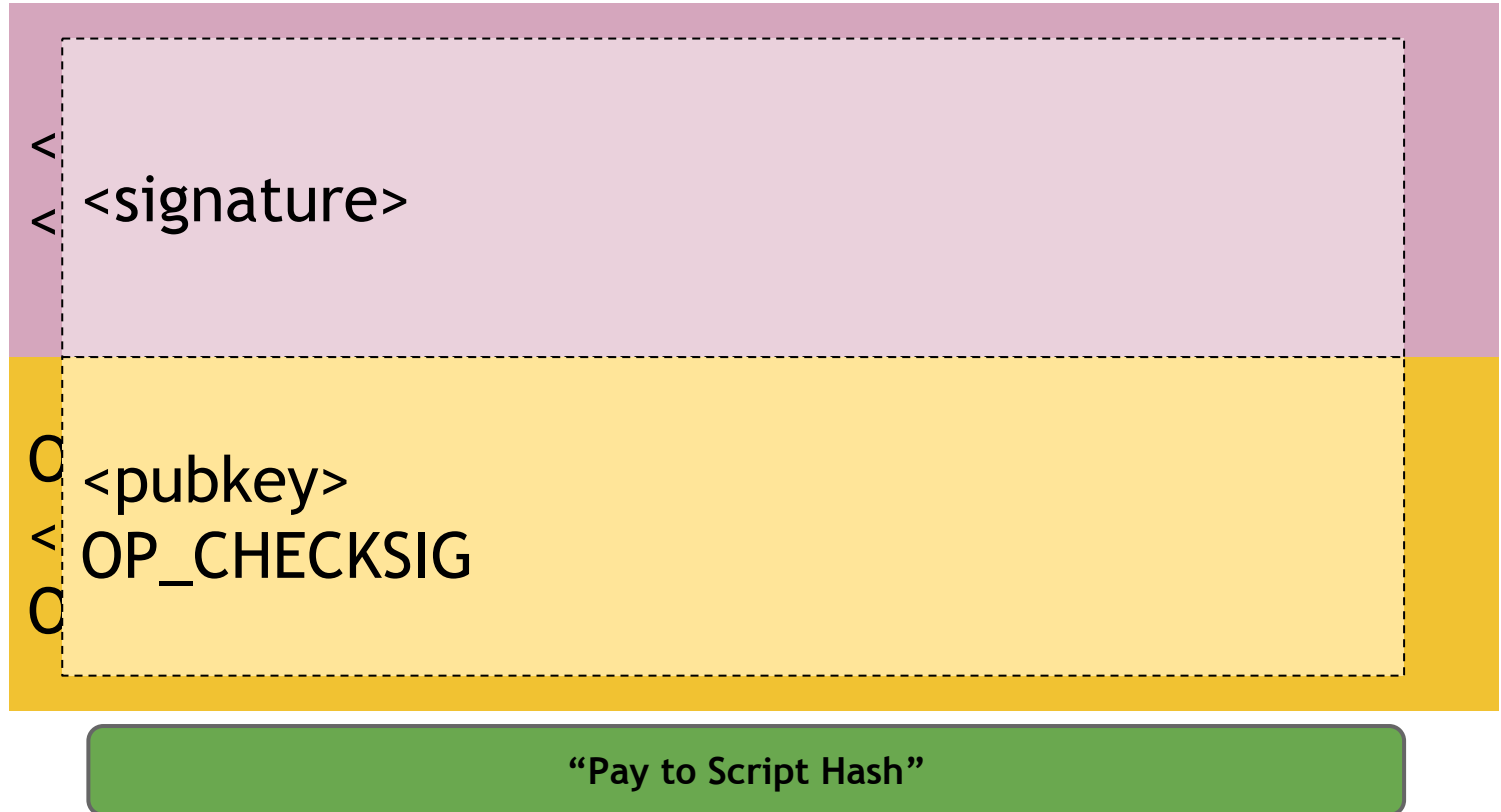
nothing's going to redeem that 😞

OP\_RETURN  
<arbitrary data>

# Should senders specify scripts?



# Idea: use the hash of redemption script



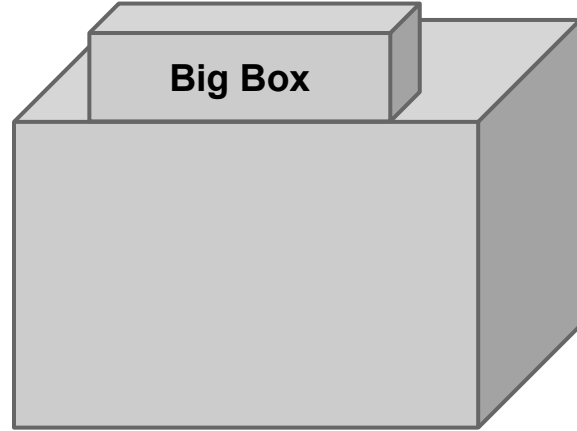
# Pay to script hash



I'm ready to pay for my purchases!



Great! Here's our address: 0x3454

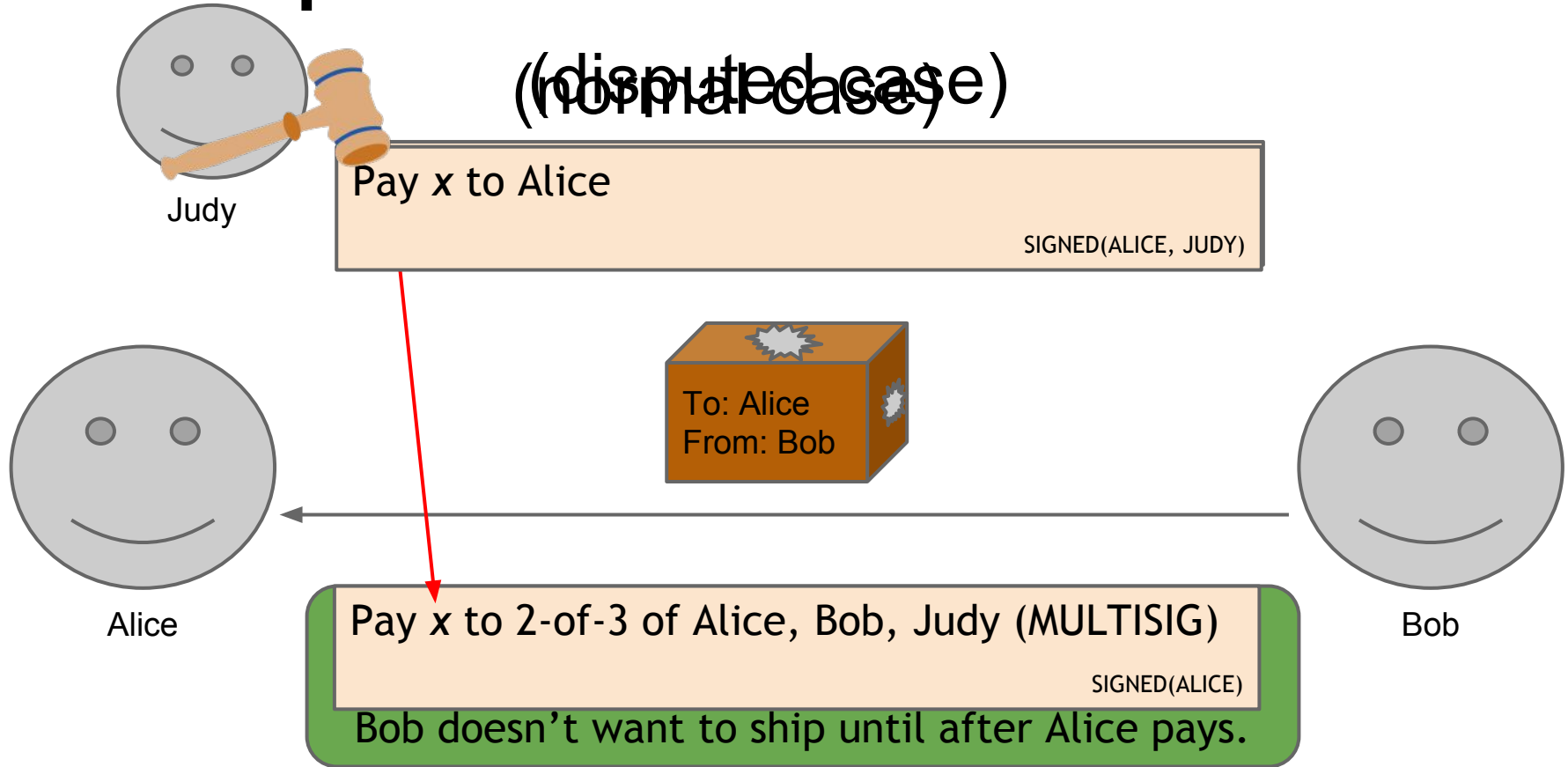


## Lecture 3.3:

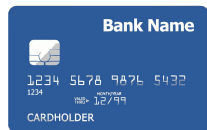
# Applications of Bitcoin scripts



# 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??

all of these could  
be  
double-spends!

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)

I'm done!

Input:  $x$ ; Pay 03 to Bob, 97 to Alice

SIGNED(ALICE) \_\_\_\_\_

I'll publish!

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)

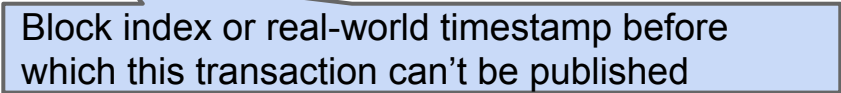
SIGNED(ALICE)

Alice

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”**

# Lecture 3.4:

## Bitcoin blocks

# Bitcoin blocks

Why bundle transactions together?

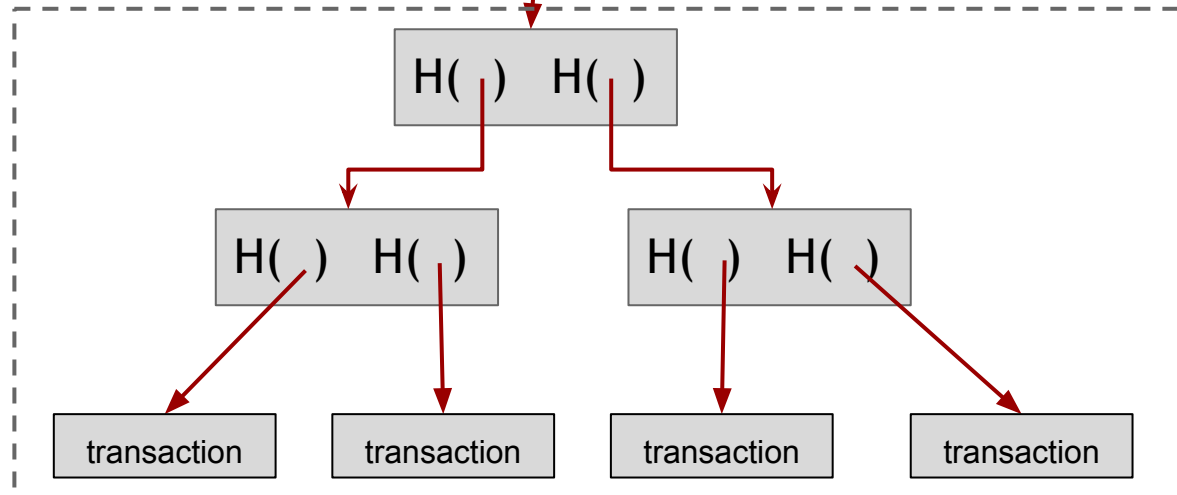
- Single unit of work for miners
- Limit length of hash-chain of blocks
  - Faster to verify history

# Bitcoin block structure

Hash chain of blocks



Hash tree (Merkle tree) of transactions in each block





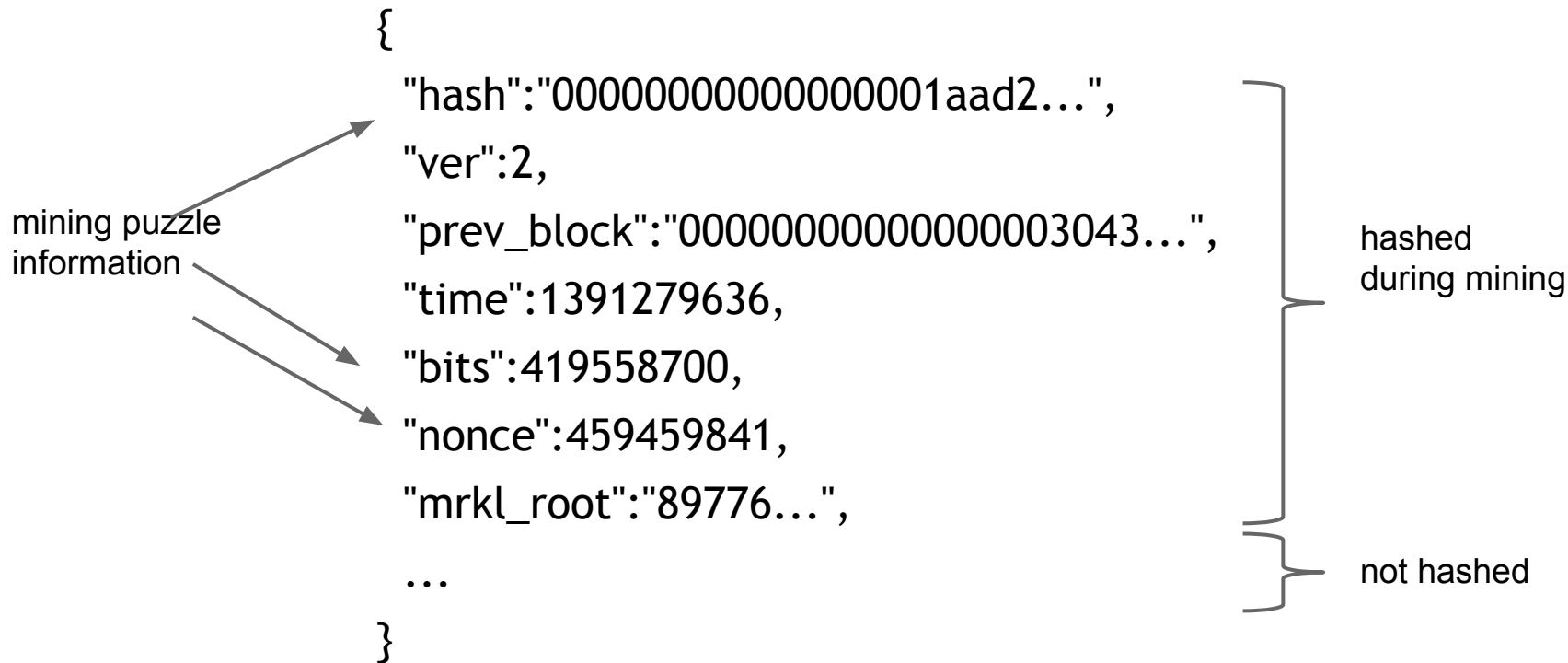
# The real deal: a Bitcoin block

block header

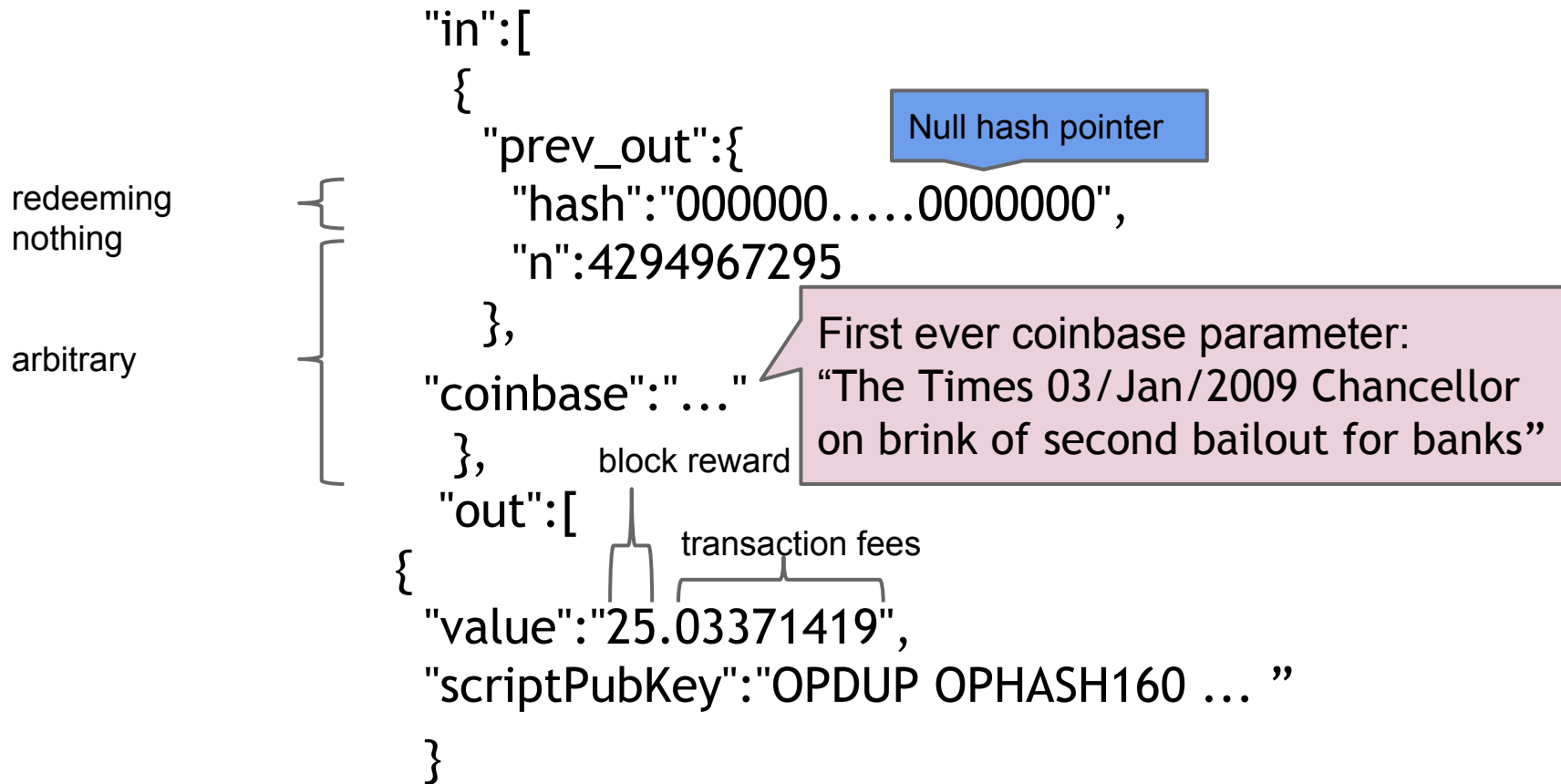
transaction data

```
{
  "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..."
  ]
}
```

# 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 | <a href="#">314018</a> (2014-08-05 02:00:40 +5 minutes) |
| Confirmations      | 9 Confirmations   |
| Relayed by IP ⓘ    | <a href="#">Blockchain.info</a>                         |
| Visualize          | <a href="#">View Tree Chart</a>                         |

| Inputs and Outputs       |   |
|--------------------------|---|
| Total Input              | 4.4775 BTC                                  |
| Total Output             | 4.4774 BTC                                  |
| Fees                     | 0.0001 BTC                                  |
| Estimated BTC Transacted | 1.0194 BTC                                  |
| Scripts                  | <a href="#">Show scripts &amp; coinbase</a> |

blockchain.info (and many other sites)

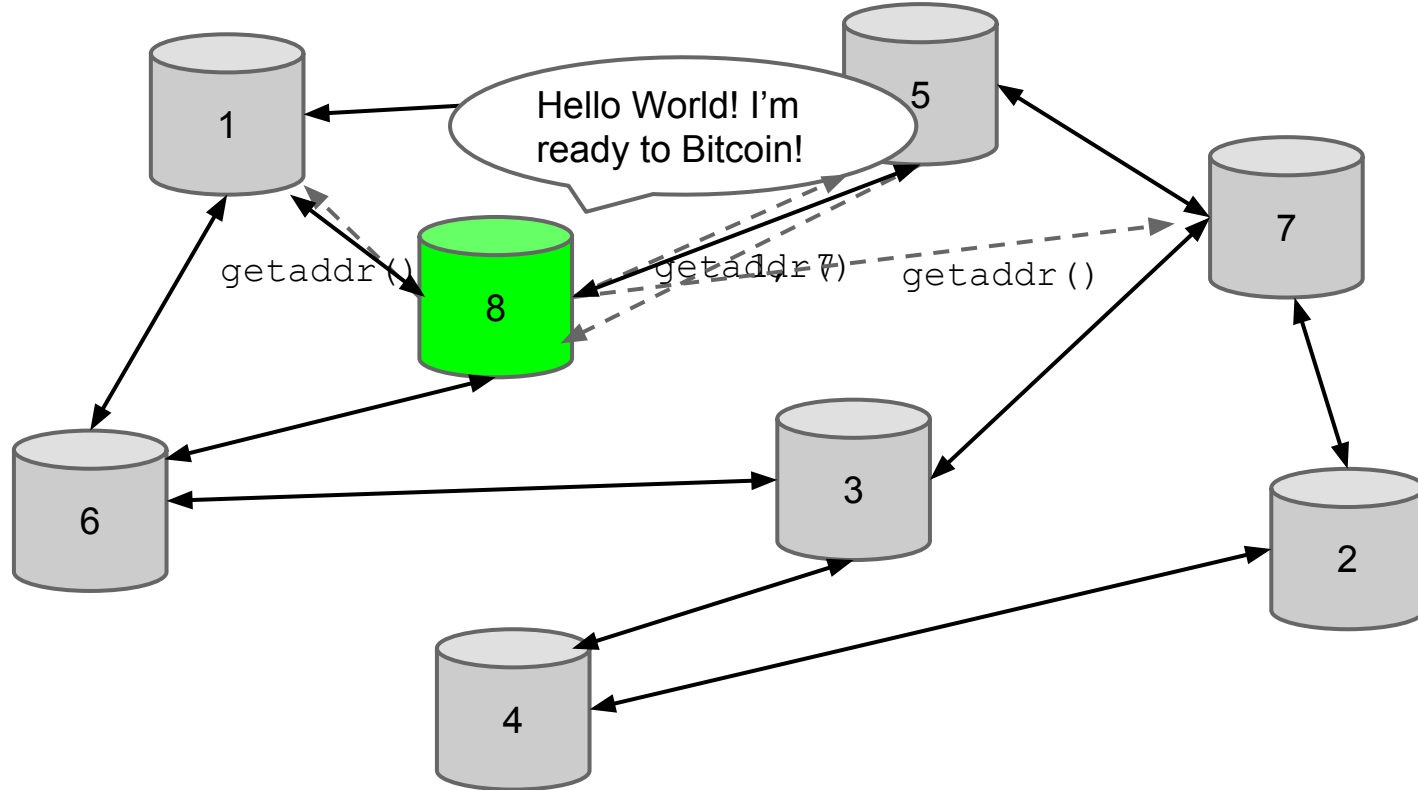
## Lecture 3.5:

# The Bitcoin network

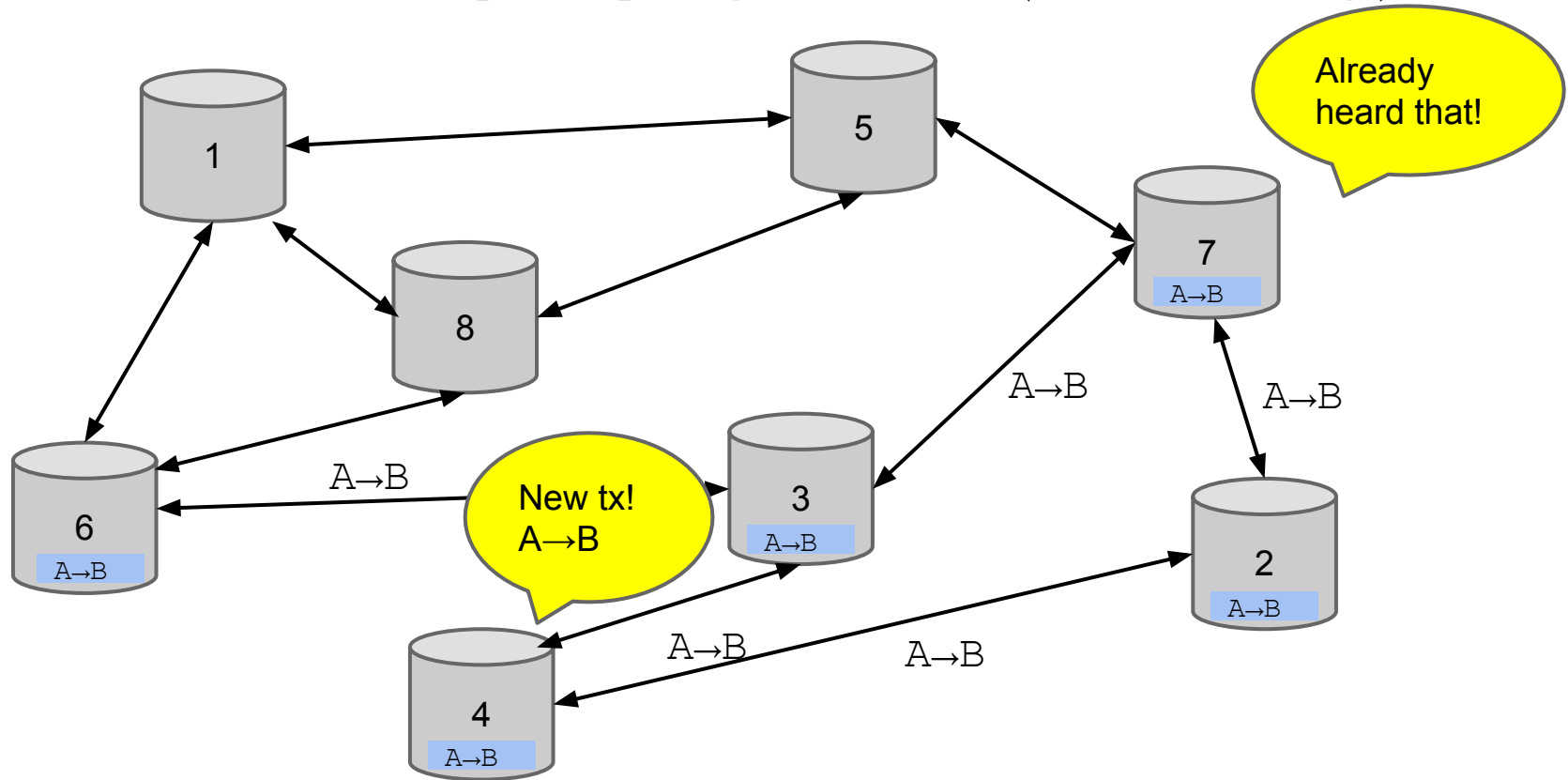
# Bitcoin P2P network

- 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)

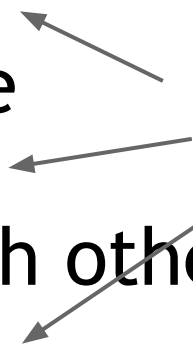




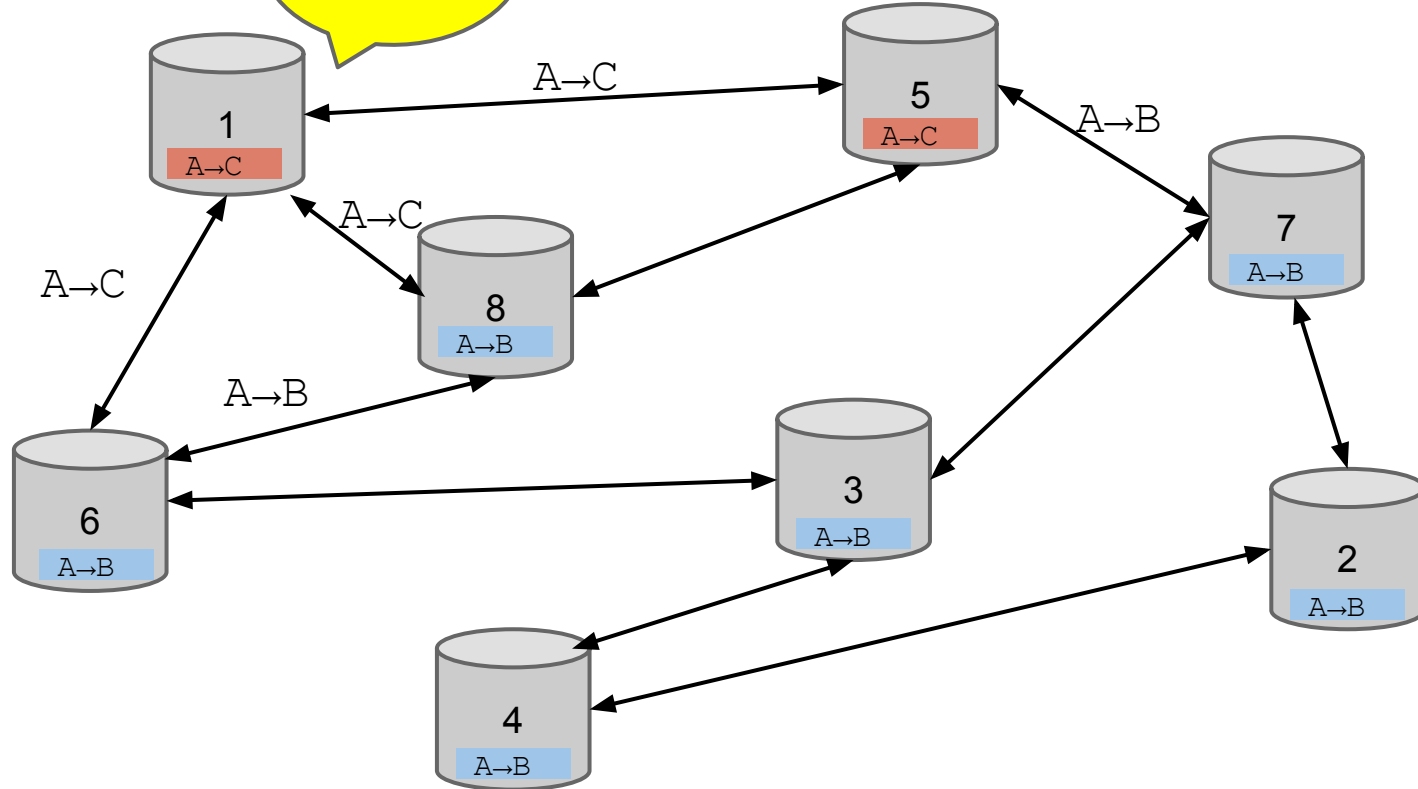
# Should I relay a proposed transaction?

- 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!

Three arrows originate from the green box and point to specific items in the list: one points to 'Avoid unusual scripts', one points to 'Avoid infinite loops', and one points to 'Avoid double-spends'.

# Nodes 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!

Stay tune for our lecture on mining!

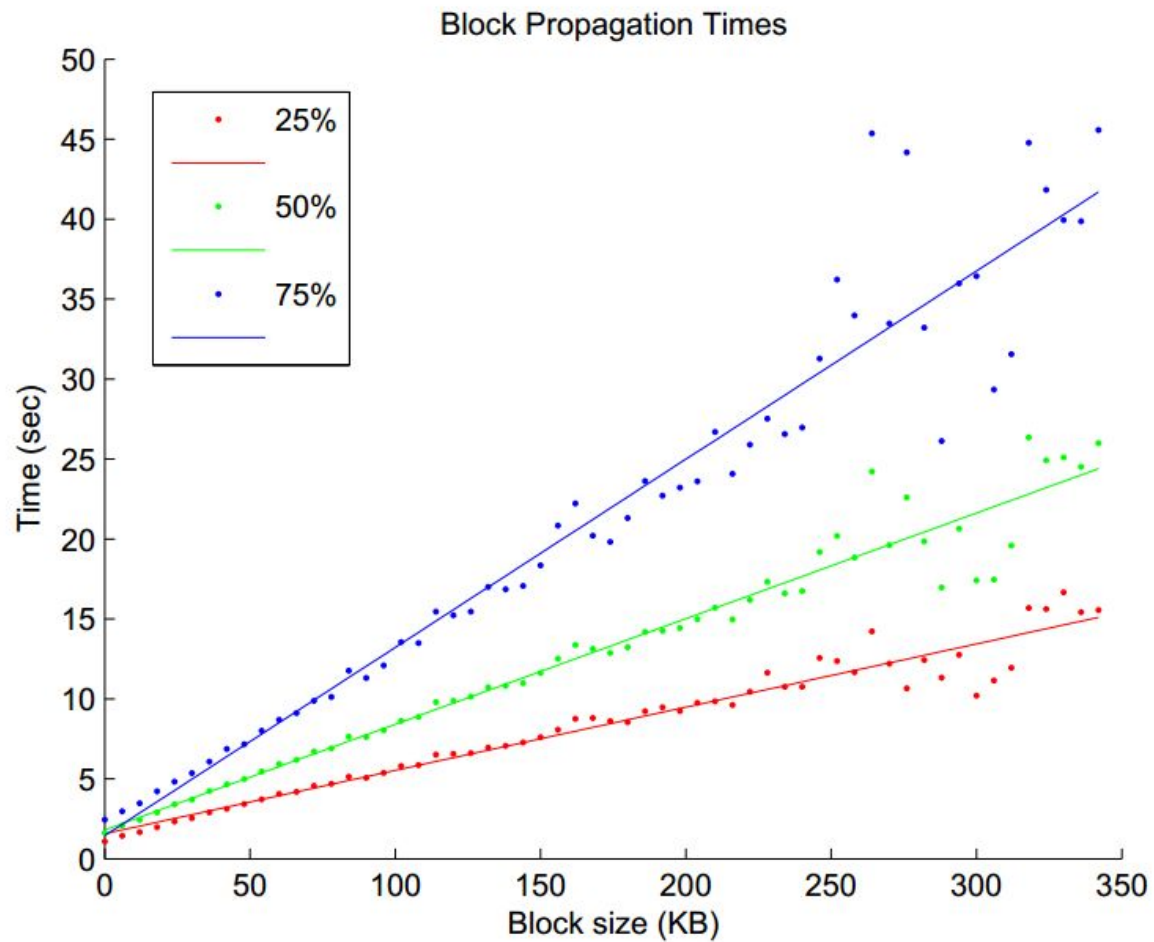
# Block propagation nearly identical

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...



# How big is the network?

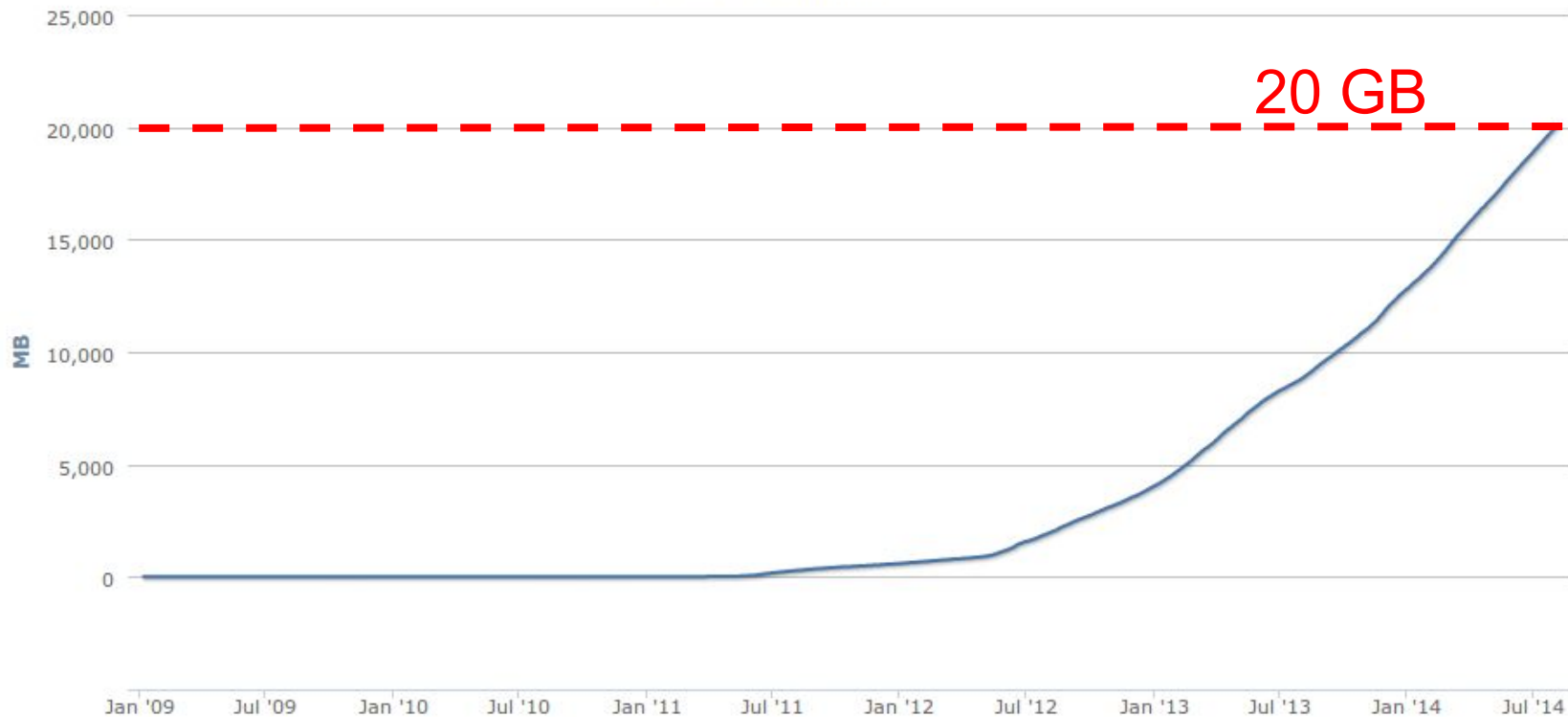
- Impossible to measure exactly
- Estimates-up to 1M IP addresses/month
- Only about 5-10k “full nodes”
  - Permanently connected
  - Fully-validate
- This number may be dropping!

# Fully-validating nodes

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

# Storage costs

Blockchain Size  
Source: blockchain.info





# Tracking the UTXO set

- **U**nspent **T**ransaction **O**utput
  - Everything else can be stored on disk
- Currently ~12 M UTXOs
  - Out of 44 M transactions
- Can easily fit into RAM

# 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! (20 GB-23MB)

# 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”

## Lecture 3.6:

### 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
- 100 M *satoshis* per bitcoin
- 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:

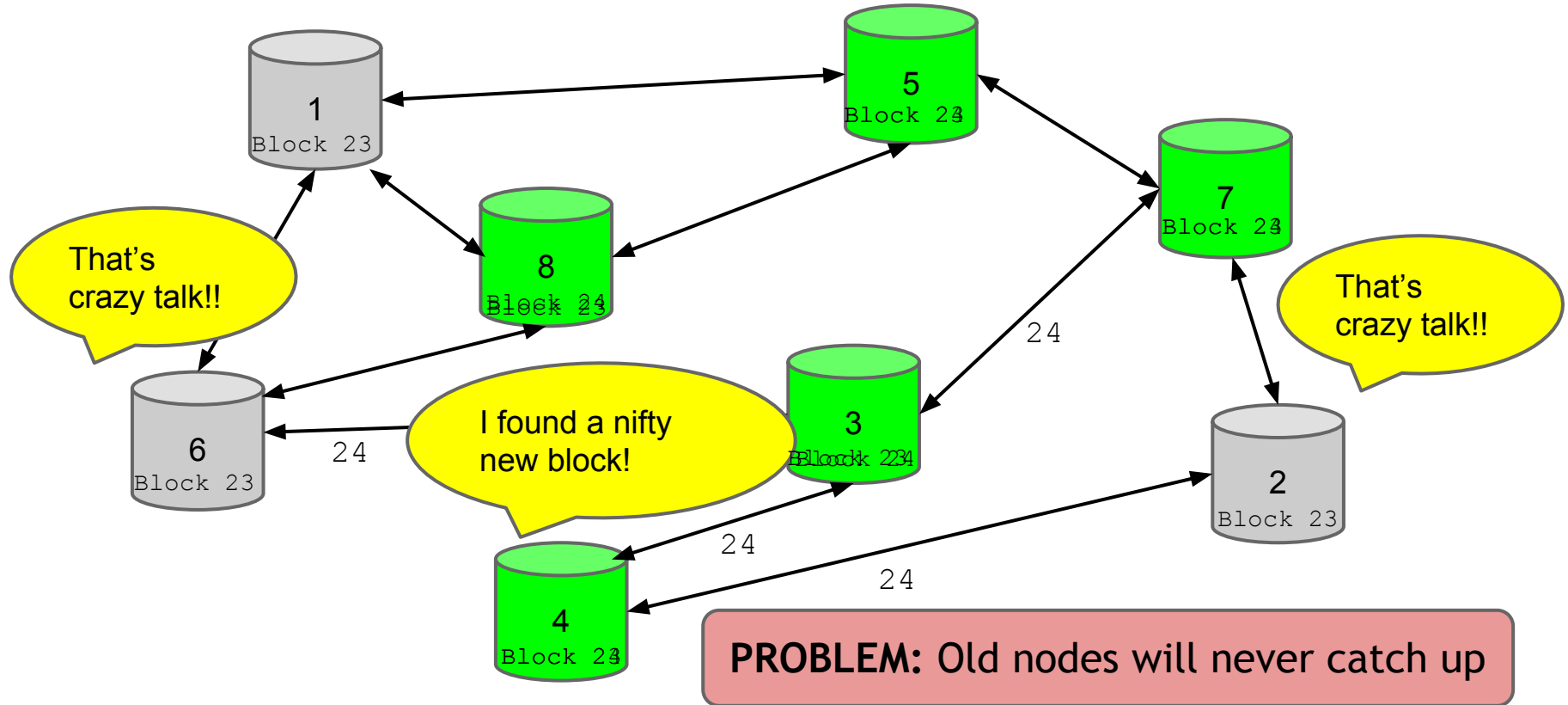
- VISA: 2,000-10,000 transactions/sec
- PayPal: 50-100 transaction/sec

# 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

Currently seem very unlikely to happen

Stay tuned for our lecture on altcoins!

In the next lecture...

# Human beings aren't Bitcoin nodes

- How do people interact with the network?
- How do people exchange bitcoins for cash?
- How do people securely store bitcoins?

Currency needs to work for people, not nodes