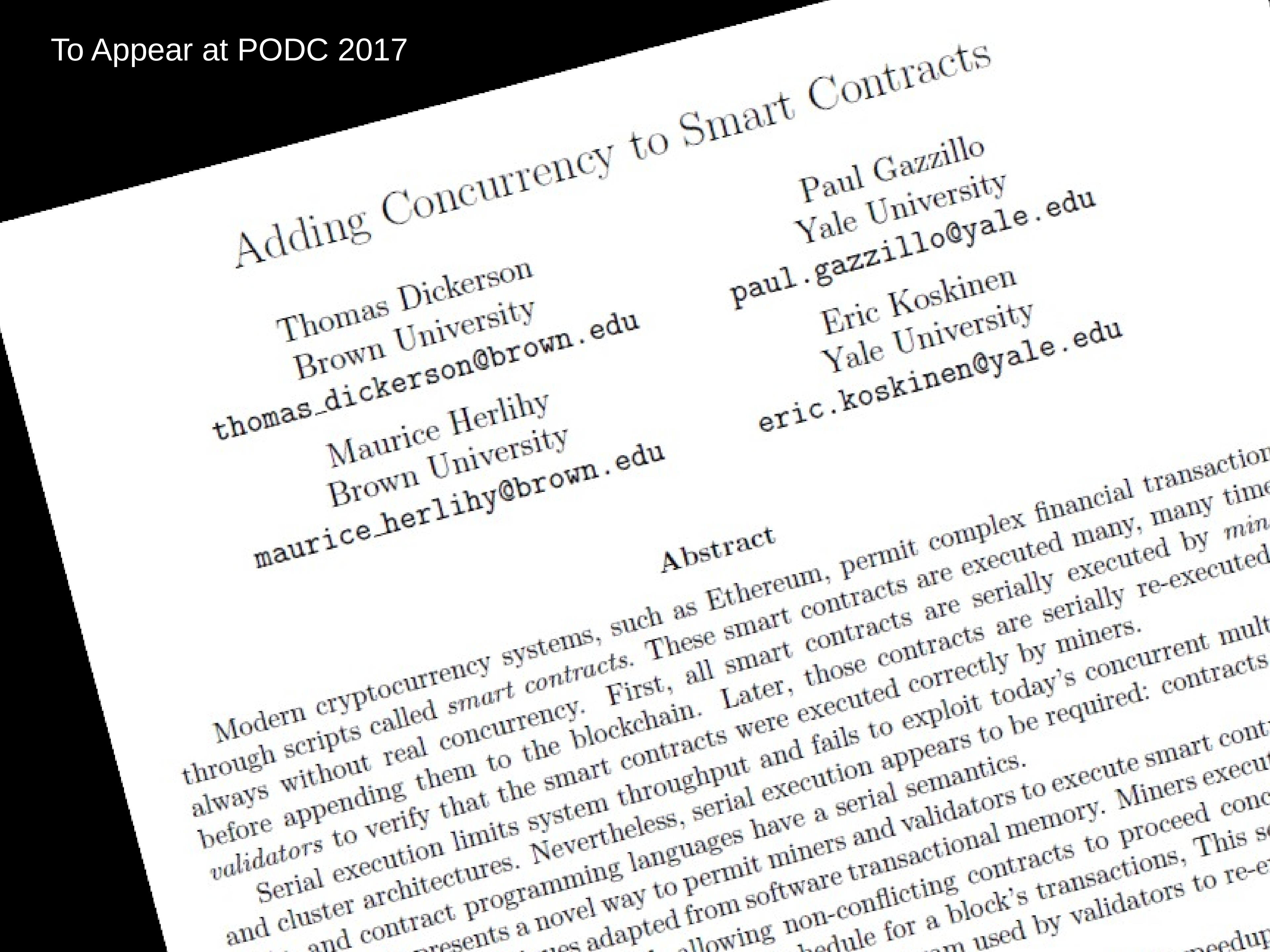
To Appear at PODC 2017



Adding Concurrency to Smart Contracts

Thomas Dickerson
Brown University
thomas_dickerson@brown.edu

Paul Gazzillo
Yale University
paul.gazzillo@yale.edu

Maurice Herlihy
Brown University
maurice_herlihy@brown.edu

Eric Koskinen
Yale University
eric.koskinen@yale.edu

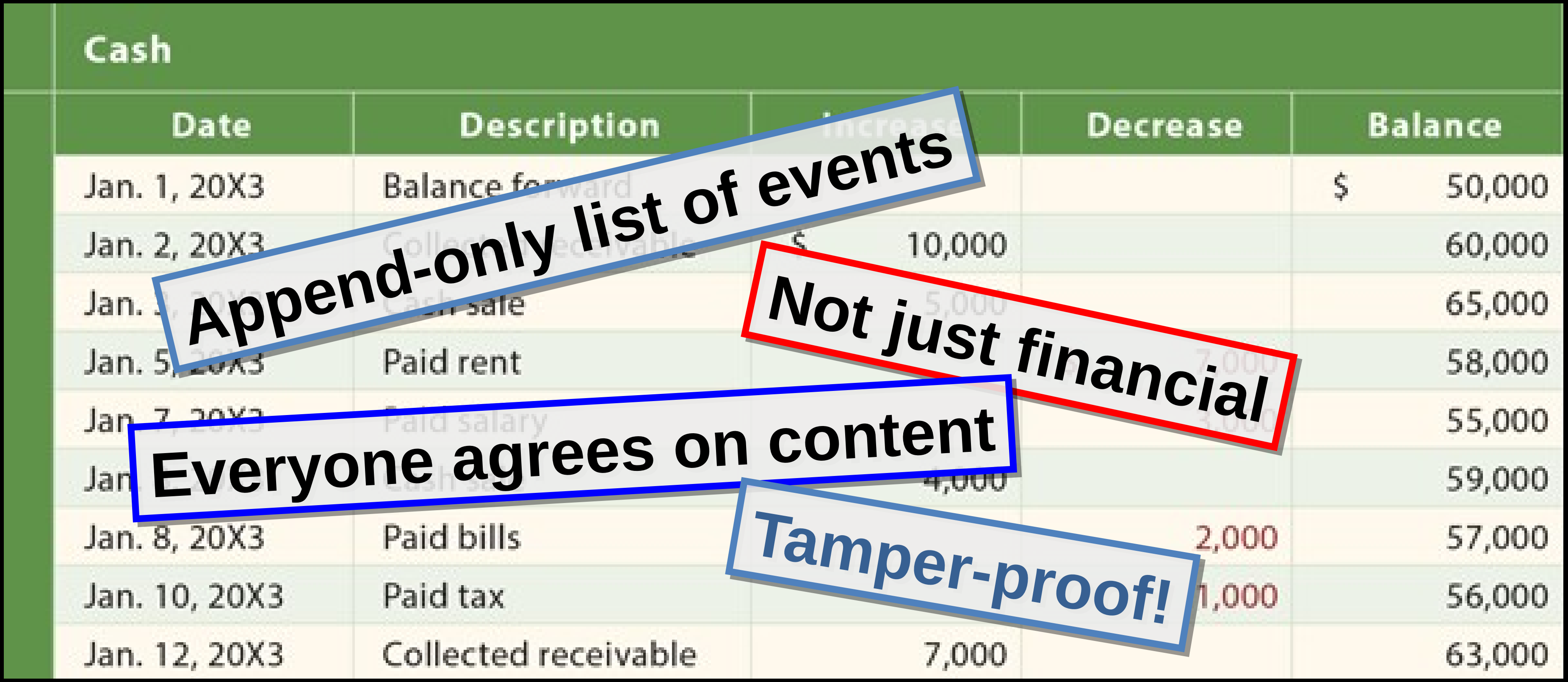
Abstract

Modern cryptocurrency systems, such as Ethereum, permit complex financial transaction
through scripts called *smart contracts*. These smart contracts are executed many, many time
always without real concurrency. First, all smart contracts are serially executed by *min*
before appending them to the blockchain. Later, those contracts are serially re-executed
validators to verify that the smart contracts were executed correctly by miners.

Serial execution limits system throughput and fails to exploit today's concurrent mult
and cluster architectures. Nevertheless, serial execution appears to be required: contracts
and contract programming languages have a serial semantics.

presents a novel way to permit miners and validators to execute smart cont
ues adapted from software transactional memory. Miners execu
allowing non-conflicting contracts to proceed conc
hedule for a block's transactions, This s
m used by validators to re-e
eedu

Abstraction:
Distributed Ledger



Cash

Date	Description	Increase	Decrease	Balance
Jan. 1, 20X3	Balance forward			$ 50,000
Jan. 2, 20X3	Collected receivable	$ 10,000		60,000
Jan. 3, 20X3	Cash sale	5,000		65,000
Jan. 5, 20X3	Paid rent		7,000	58,000
Jan. 7, 20X3	Paid salary		3,000	55,000
Jan[illegible]	Cash sa[illegible]	4,000		59,000
Jan. 8, 20X3	Paid bills		2,000	57,000
Jan. 10, 20X3	Paid tax		1,000	56,000
Jan. 12, 20X3	Collected receivable	7,000		63,000

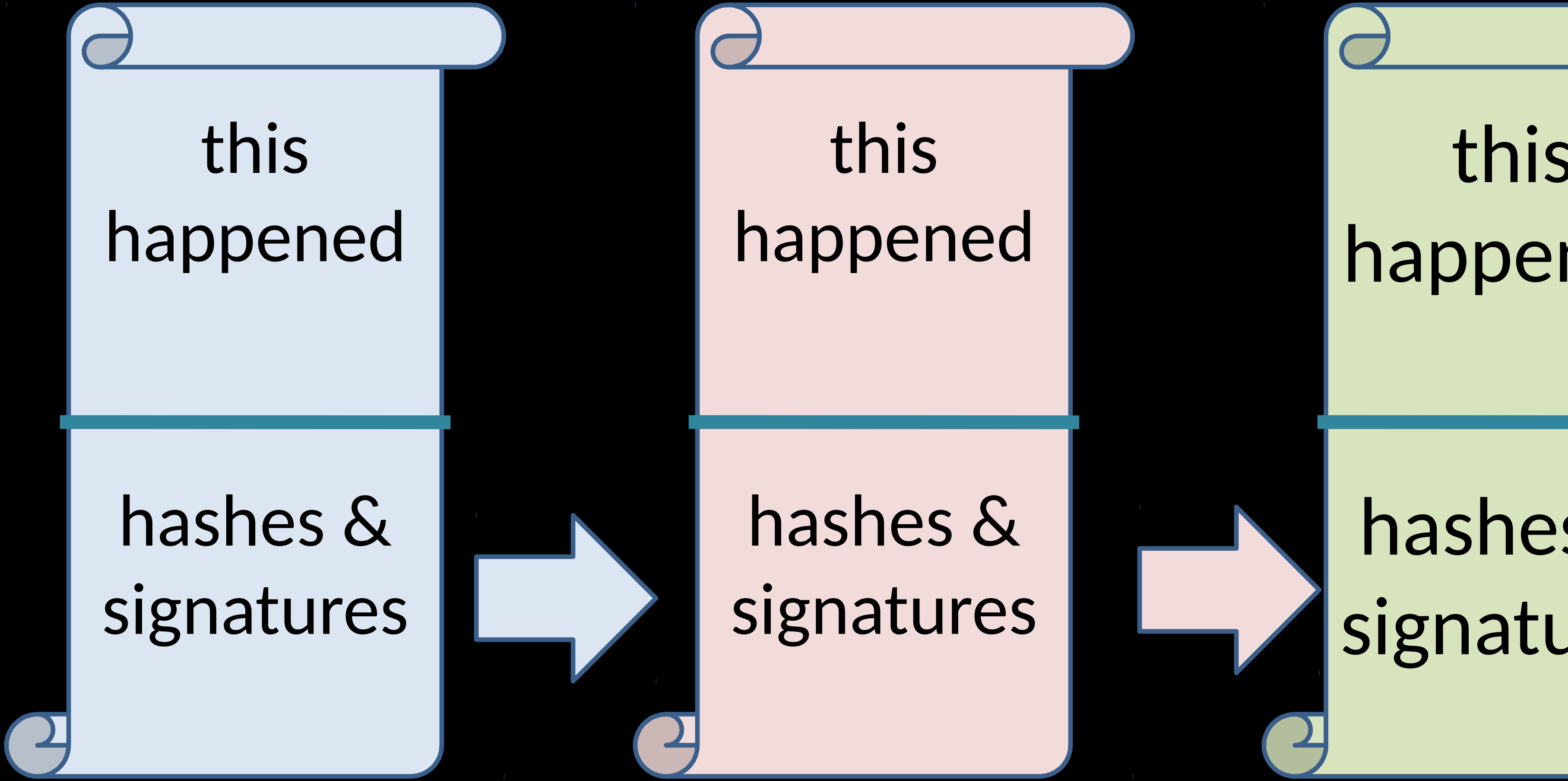
Append-only list of events

Not just financial

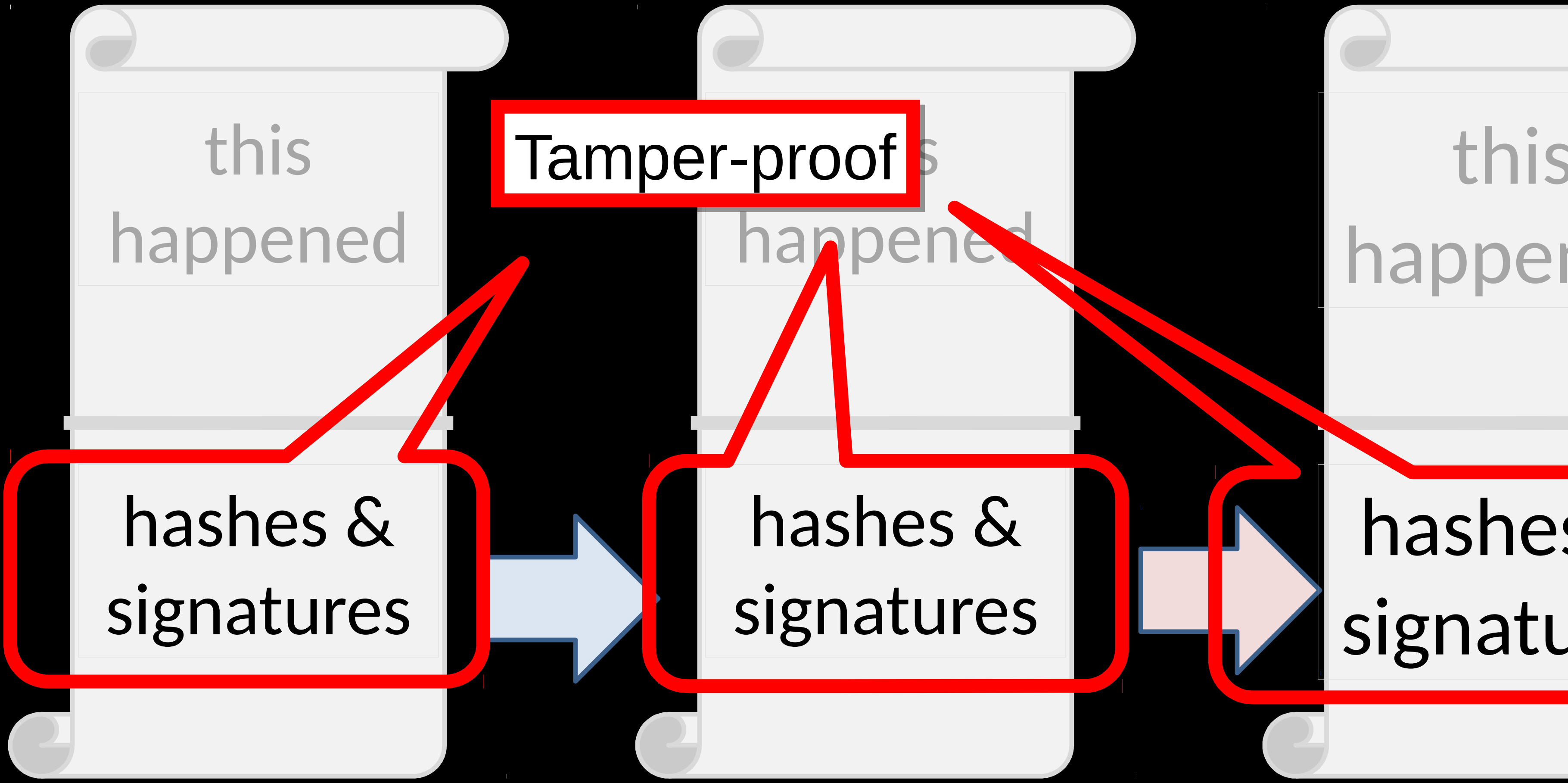
Everyone agrees on content

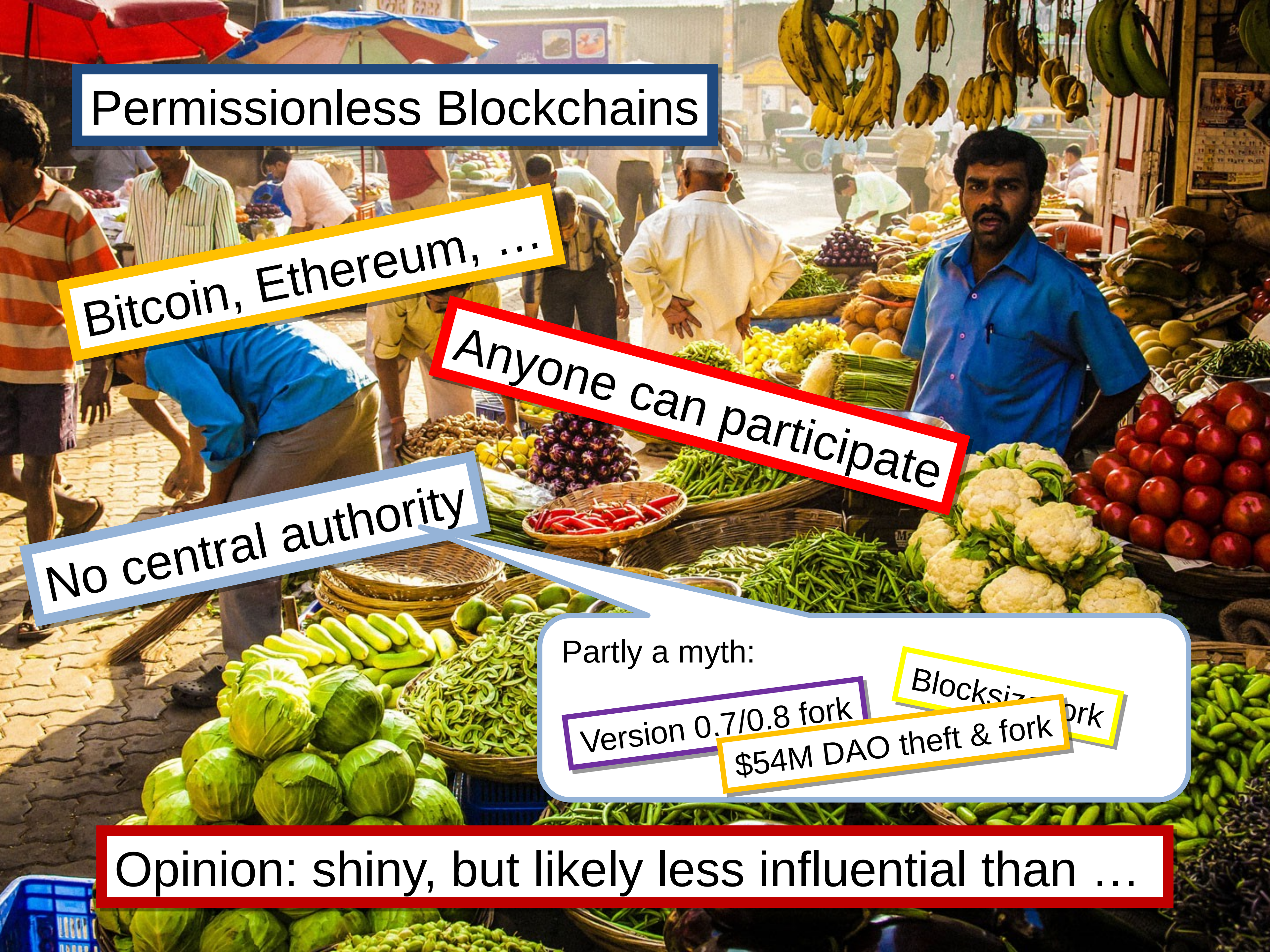
Tamper-proof!

Implementation: Blockchain



Implementation: Blockchain





Permissionless Blockchains

Bitcoin, Ethereum, …

Anyone can participate

No central authority

Partly a myth:

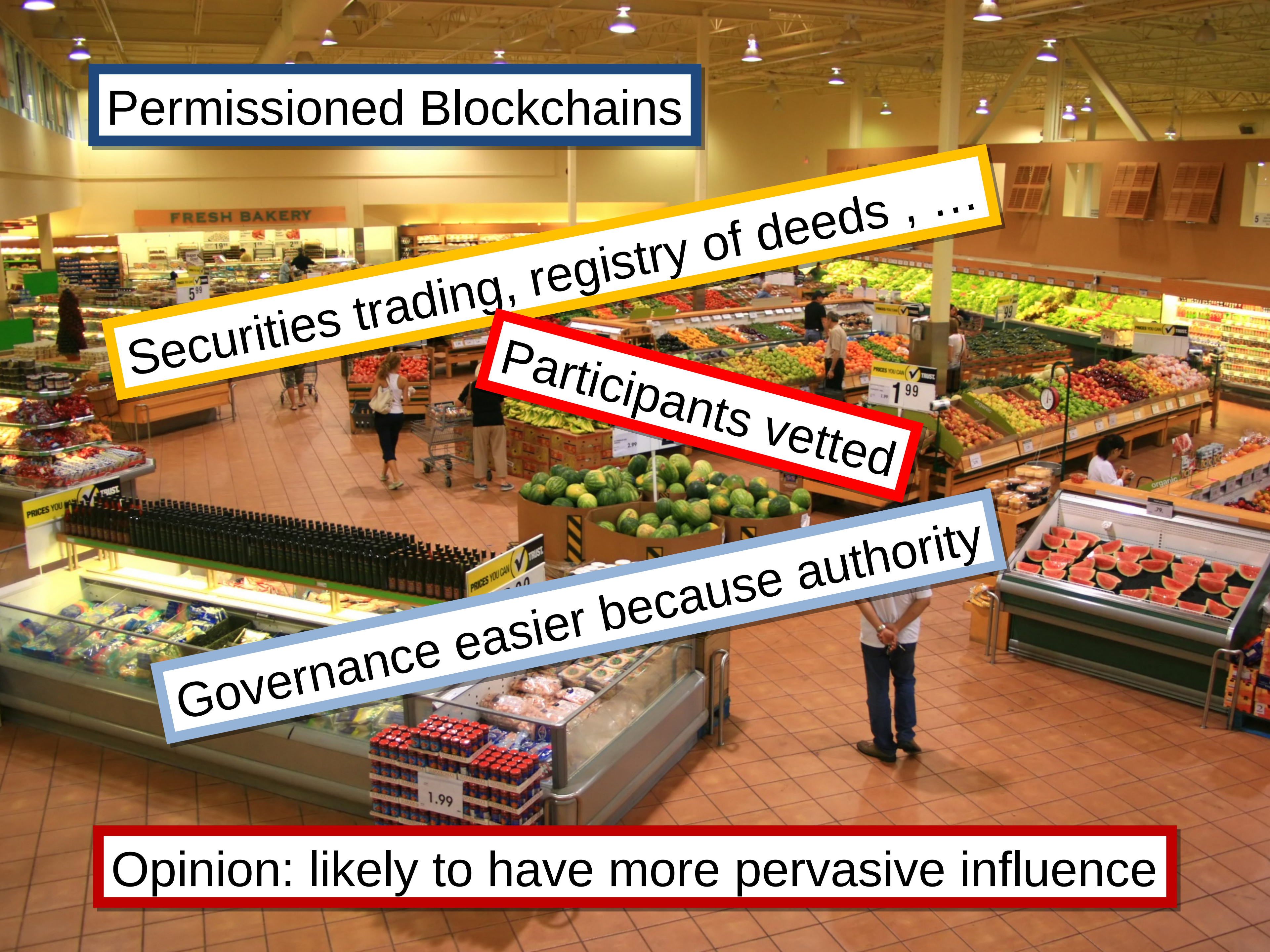
Version 0.7/0.8 fork

Blocksize fork

$54M DAO theft & fork

Opinion: shiny, but likely less influential than …

Permissioned Blockchains



Securities trading, registry of deeds , …

Participants vetted

Governance easier because authority

Opinion: likely to have more pervasive influence

Permissioned Blockchains

Securities trading, registry of deeds , …

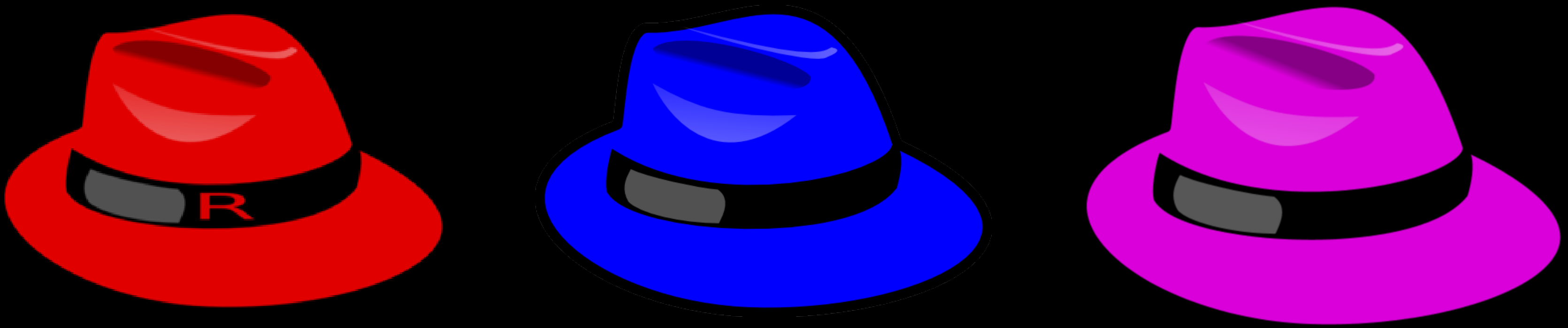
Particip

Discussion assumes permissionless because more challenging.

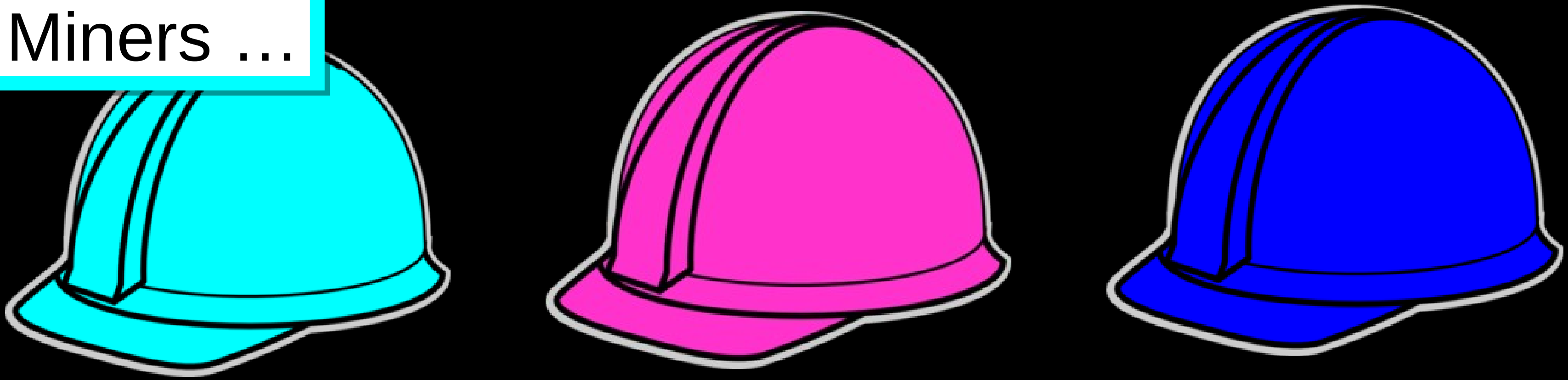
Governance easier because authority

Opinion: likely to have more pervasive influence

Clients ....

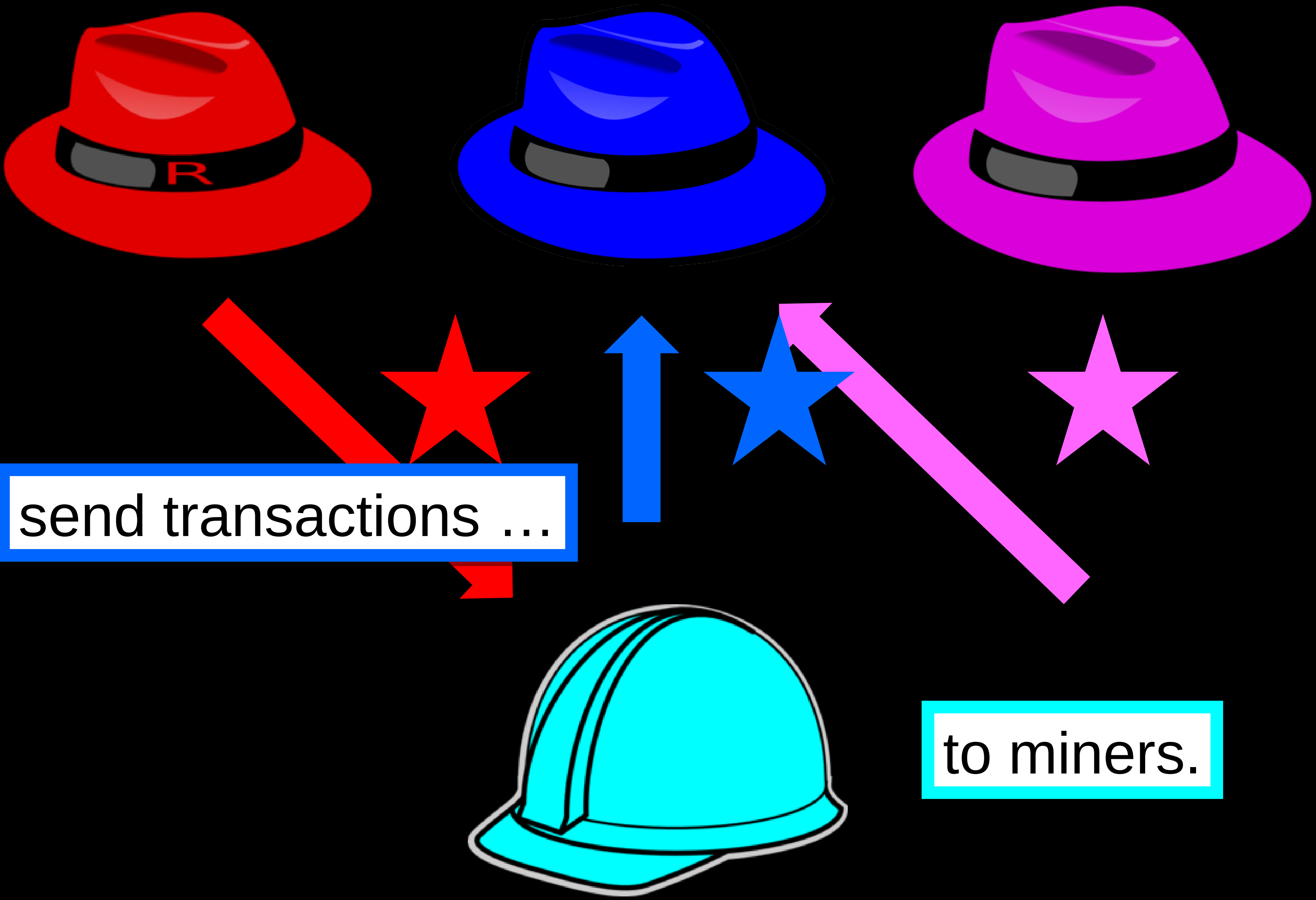


Miners ...



Validators ...

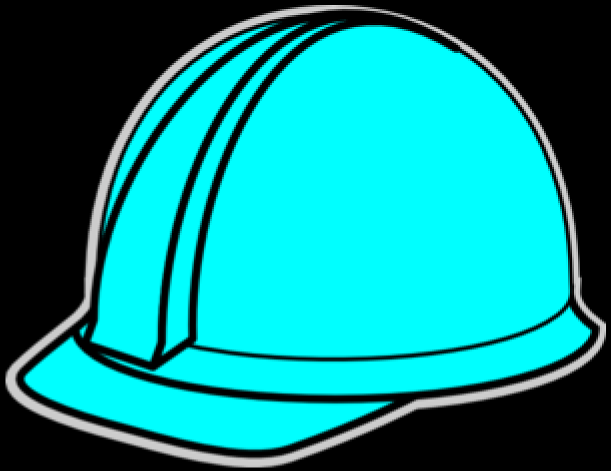
Clients ....

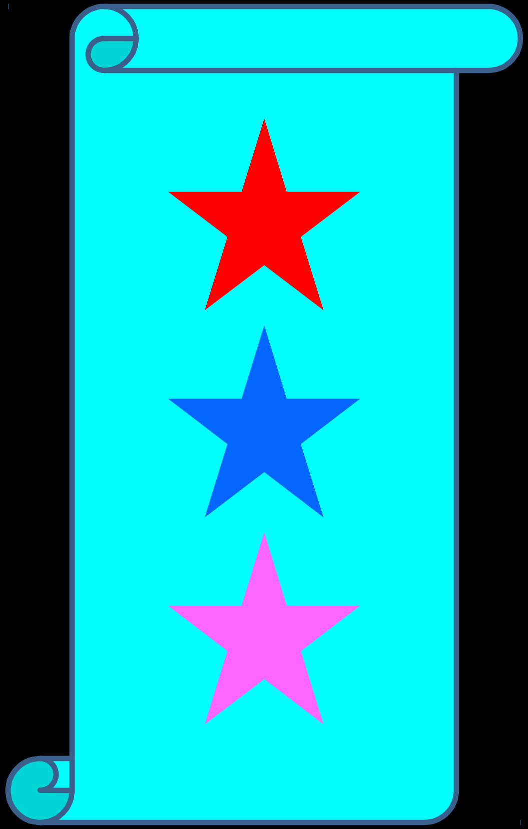


send transactions ...

to miners.

Miners …

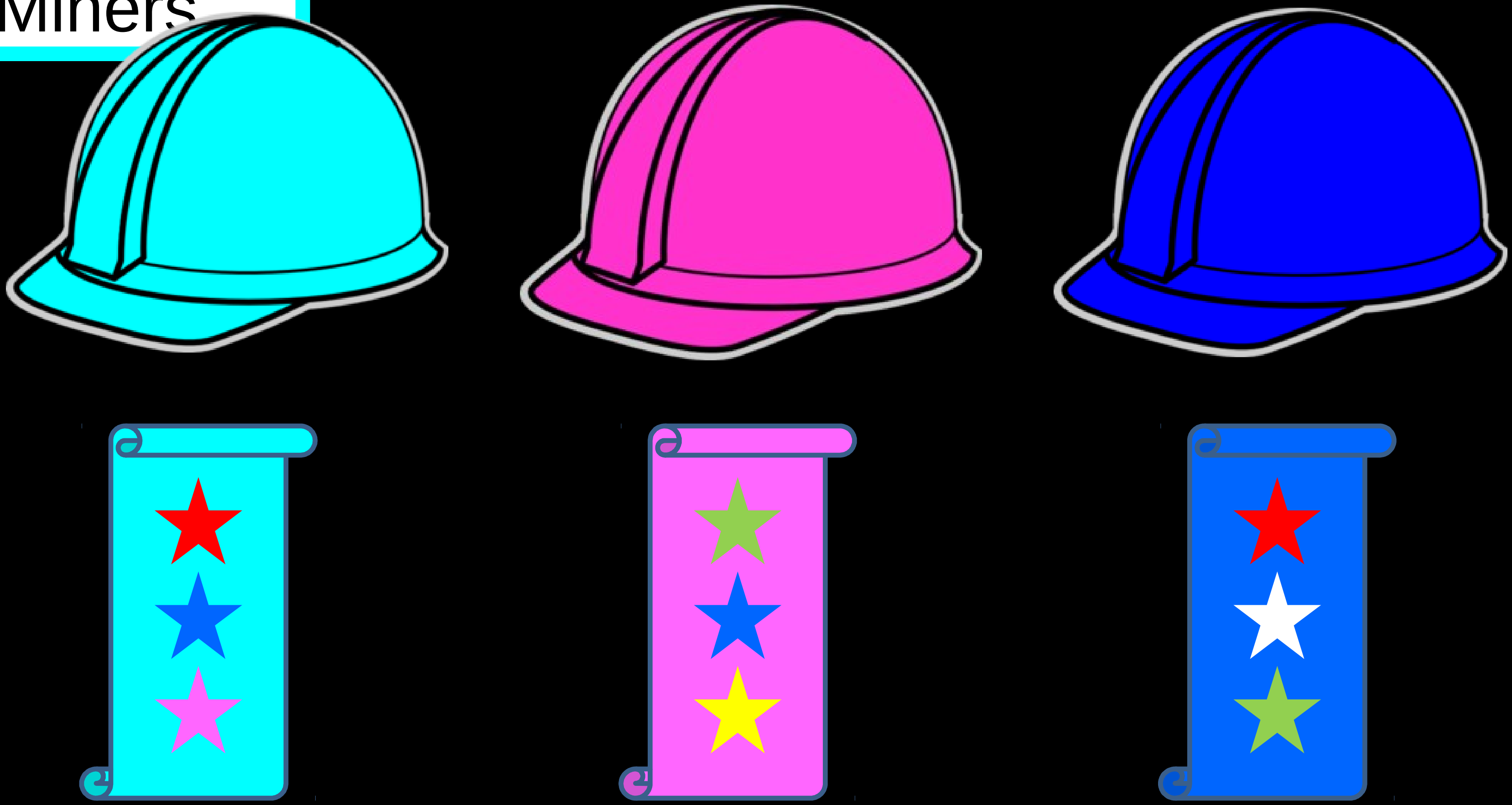




assemble transactions …

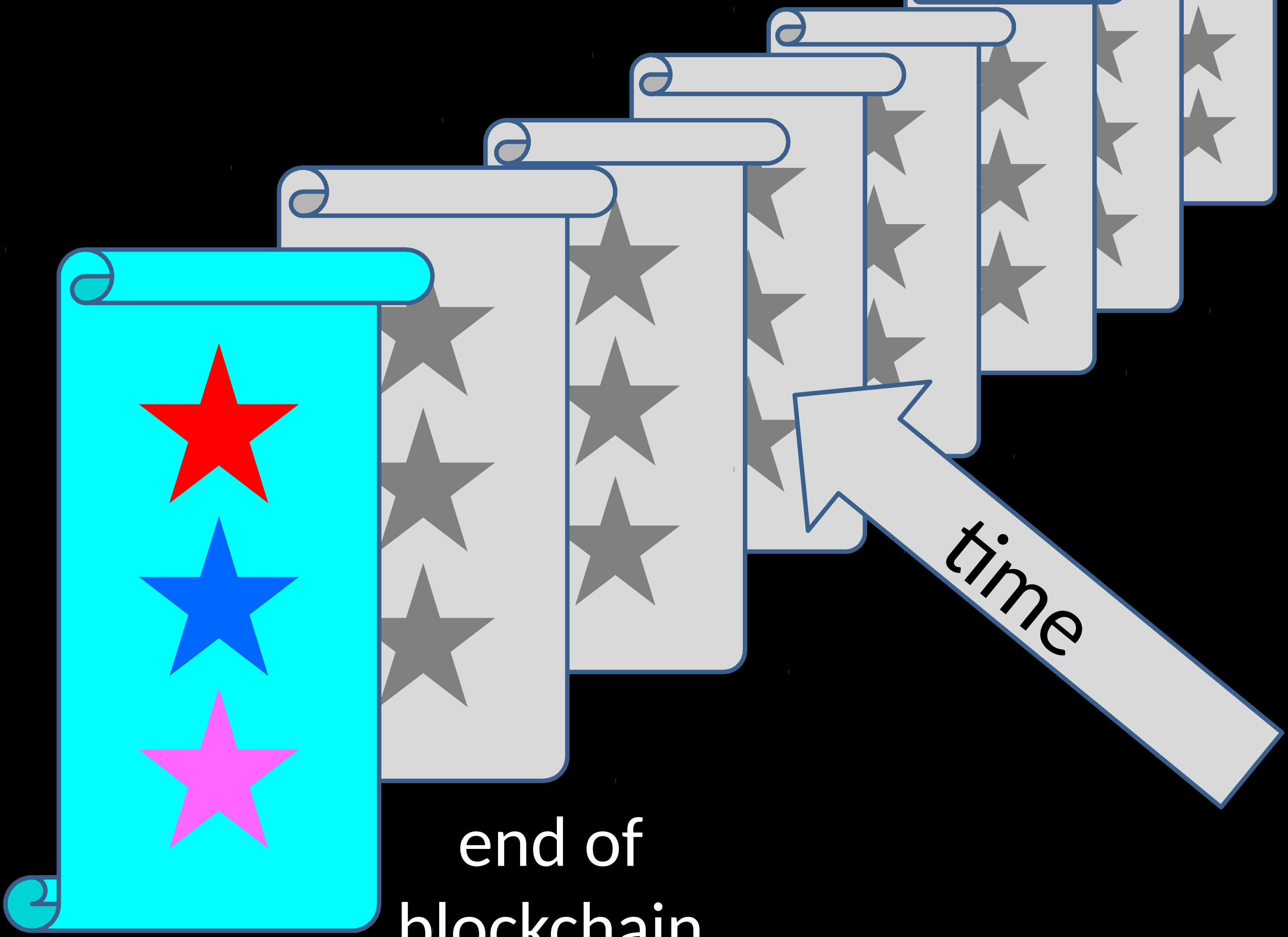
into blocks.

Miners



do distributed consensus to pick one block …

Agnostic about how.

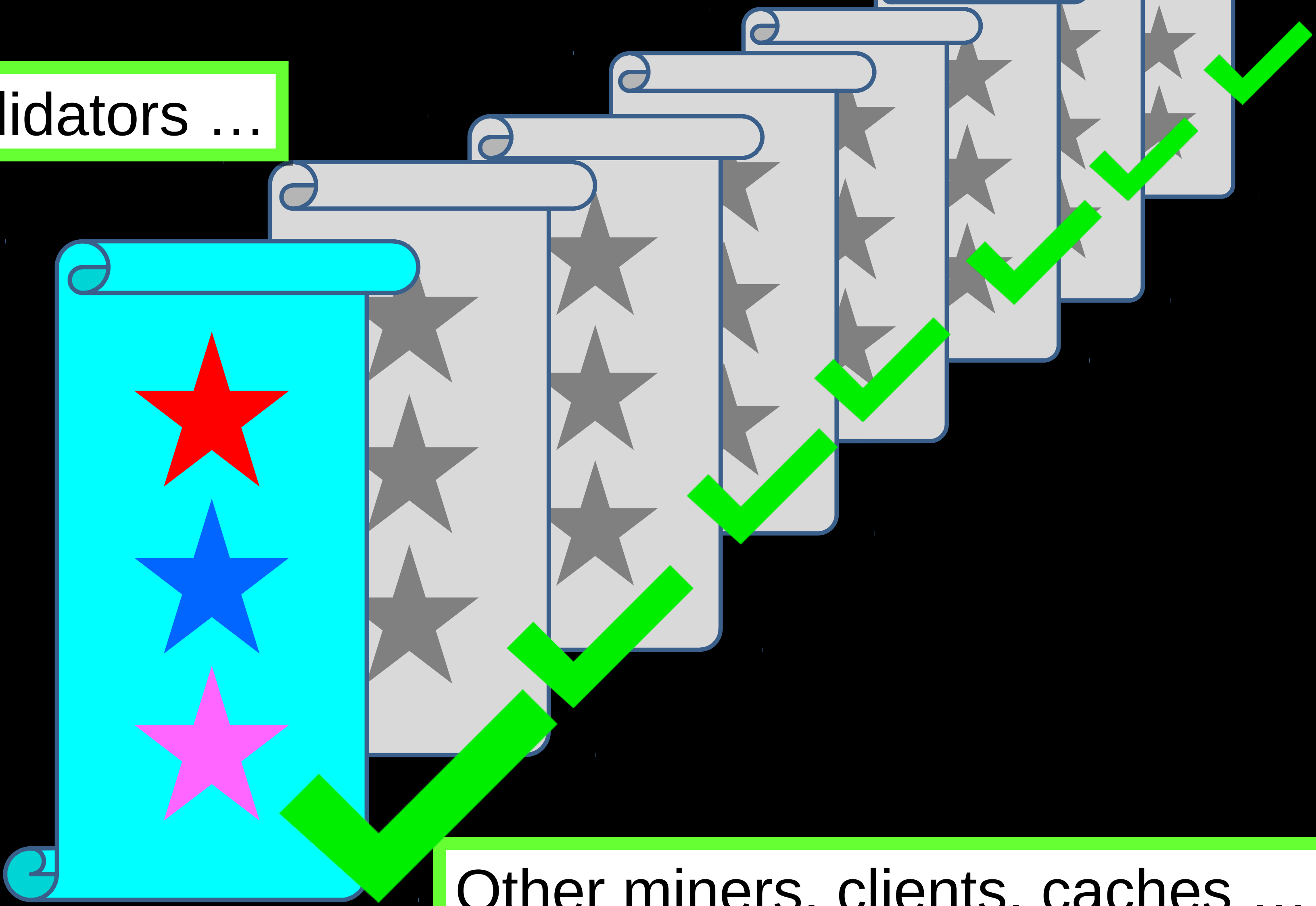


end of blockchain

consensus winnner

to append to the blockchain.

Validators …

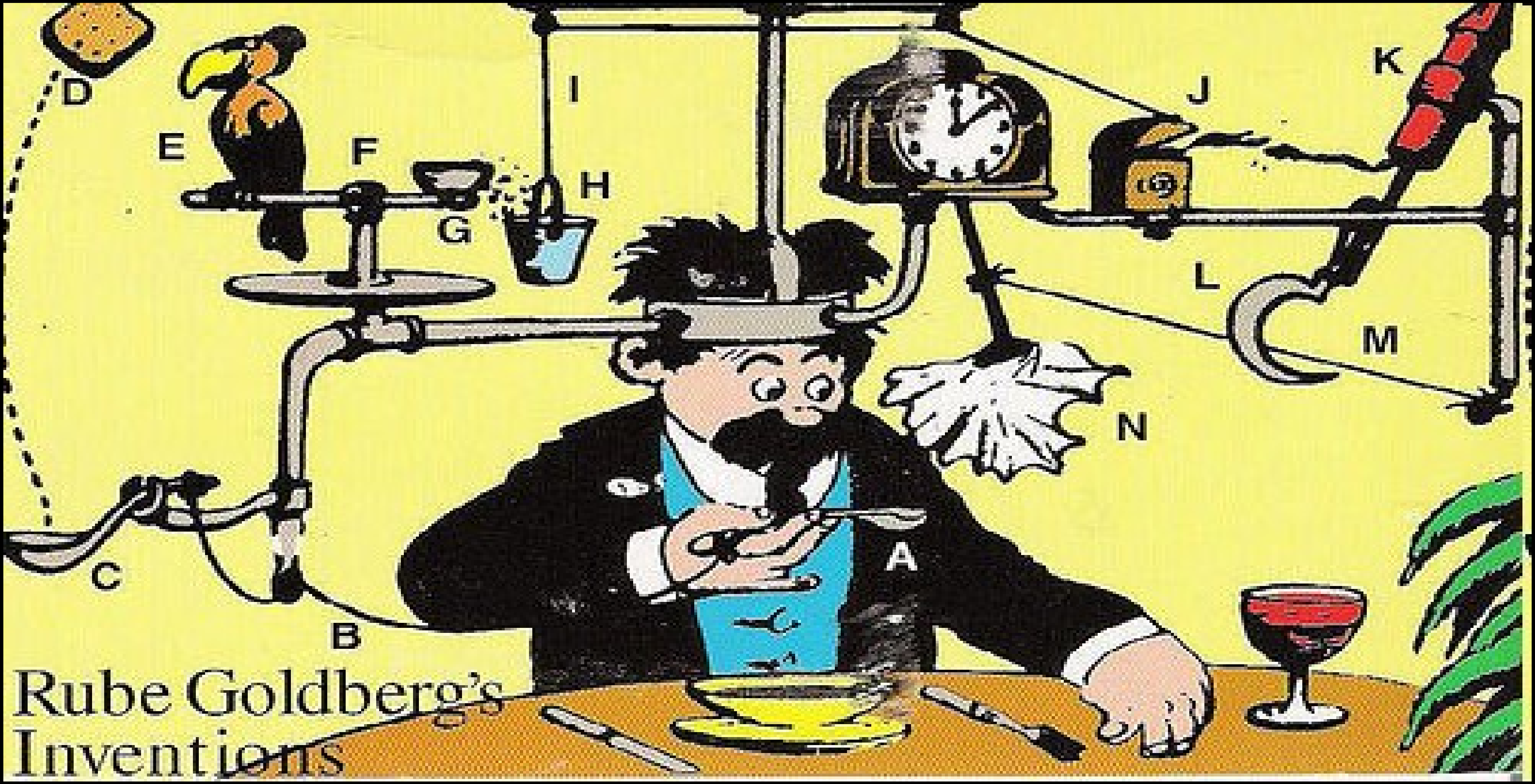


Other miners, clients, caches …

check hashes, signatures

end of
blockchain

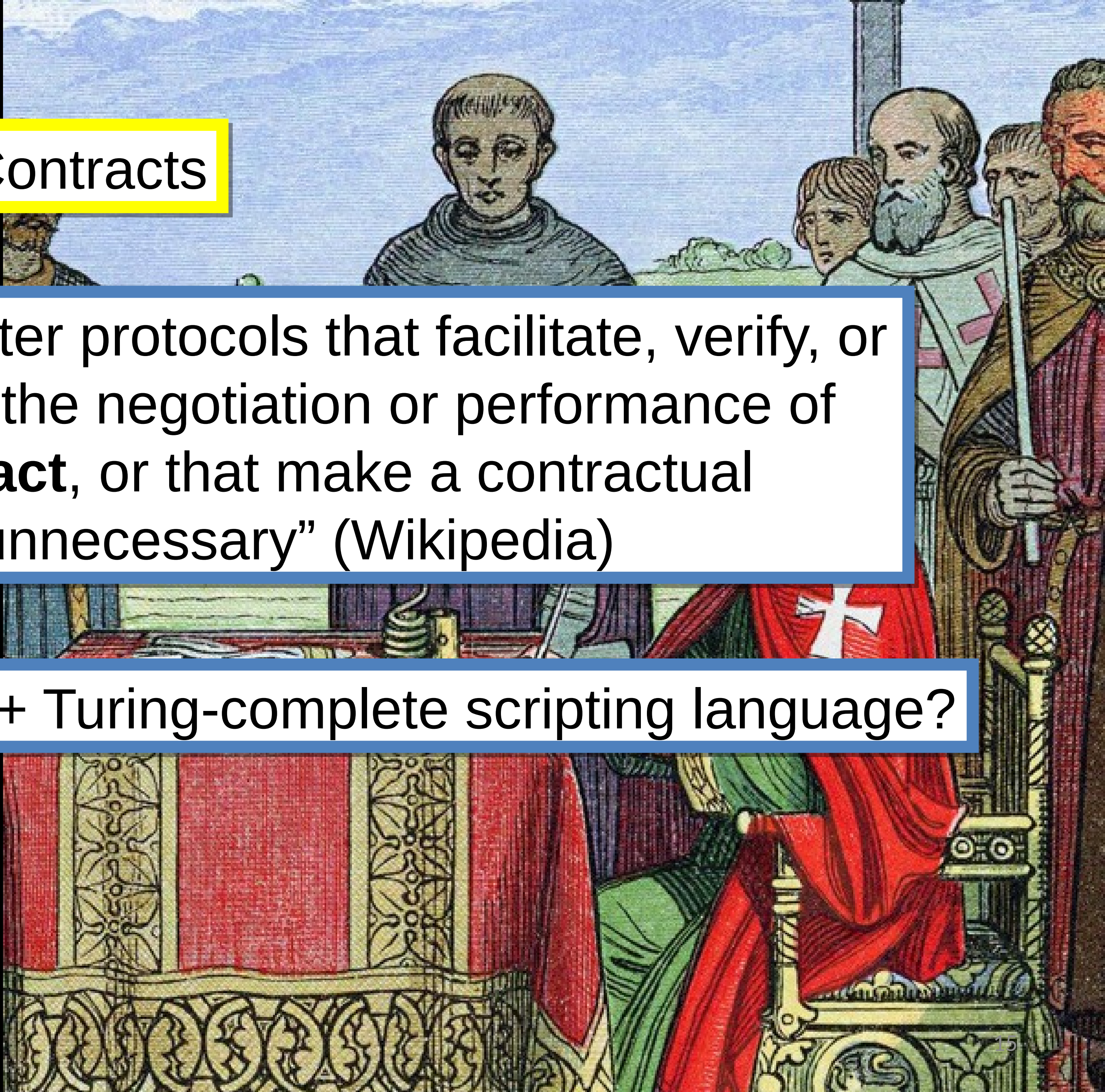
OK, we left out some details …



Smart Contracts

“Computer protocols that facilitate, verify, or enforce the negotiation or performance of a **contract**, or that make a contractual clause unnecessary” (Wikipedia)

Ledger + Turing-complete scripting language?



```
contract Ballot {
  mapping(address => Voter)
     public voters;
     … // more state decls
  function vote(uint proposal)
    Voter sender = voters[msg.sender];
    if (sender.voted)
      throw;
    sender.voted = true;
    sender.vote = proposal;
    proposals[proposal].voteCount
      += sender.weight;
  }
  …
}
```

Looks like an object in a language

```
contract Ballot {
  mapping(address => Voter)
     public voters;
     … // more state decls
  function vote(uint proposal)
    Voter sender = voters[msg.sender];
    if (sender.voted)

    sender.voted = true;

    proposals[proposal].voteCount

  }
  …
}
```

Long-lived state

Built-in data types: maps, arrays, scalars.

Tracks who can vote, who voted, choices.

```
contract Ballot {
  mapping(address => Voter)
     public voters;
     … // more state decls
  function vote(uint proposal)
    Voter sender = voters[msg.sender];
    if (sender.voted)

    sender.voted = true;
    sender.vote = proposal;
                       oteCount
      += sender.weight;
  }
  …
}
```

Functions to manipulate state

Vote for a particular proposal

```
contract Ballot {
  mapping(address => Voter)
     public voters;
     … // more state decls
  function vote(uint proposal)
    Voter sender = voters[msg.sender];
    if (sender.voted)
      throw;
    sender.voted = true;
    sender.vote = proposal;
    proposals[proposal].
      += sender.weight;
  }
  …
}
```

Who is voting?

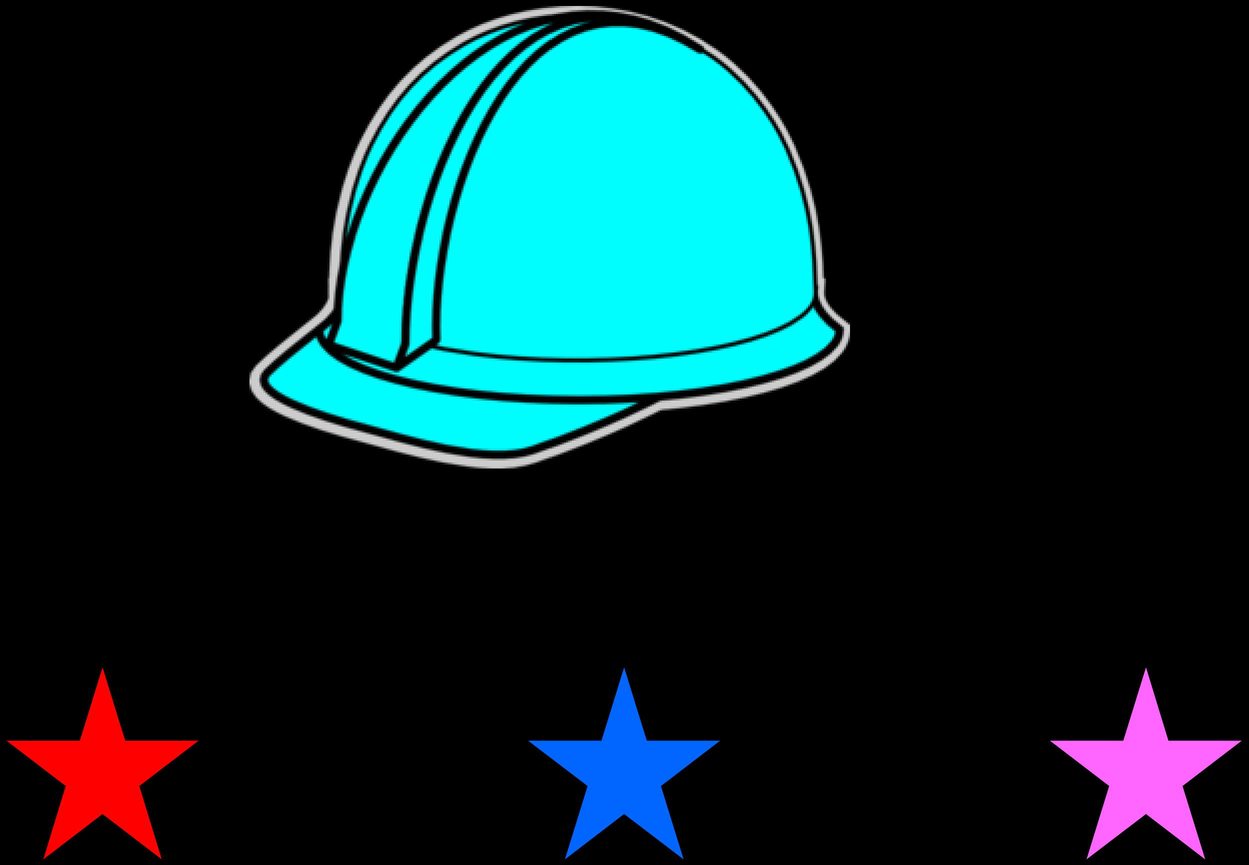
```
contract Ballot {
  mapping(address => Voter)
     public voters;
     … // more state decls
  function vote(uint proposal)
    Voter sender = voters[msg.sender];
    if (sender.voted)
      throw;
    sender.voted = true;
    sender.vote = proposal;
    proposals[proposal].voteCount
      += sender.weight;
  }
  …
}
```

No voting twice

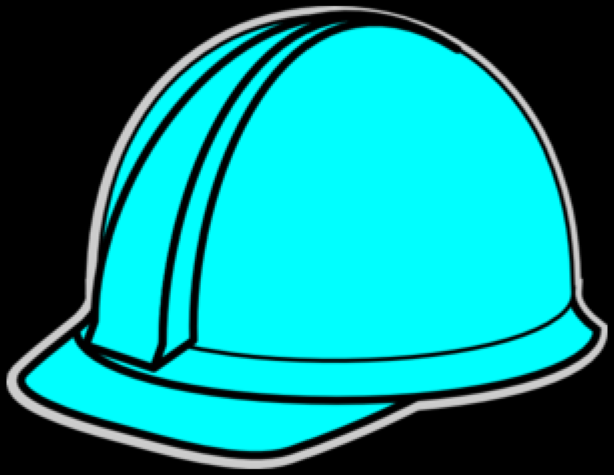
```
contract Ballot {
  mapping(address => Voter)
     public voters;
     … // more state decls
  function vote(uint p
    Voter sender = voters[msg.sender];
    if (sender.voted)
      throw;
    sender.voted = true;
    sender.vote = proposal;
    proposals[proposal].voteCount
      += sender.weight;
  }
  …
}
```

Record vote

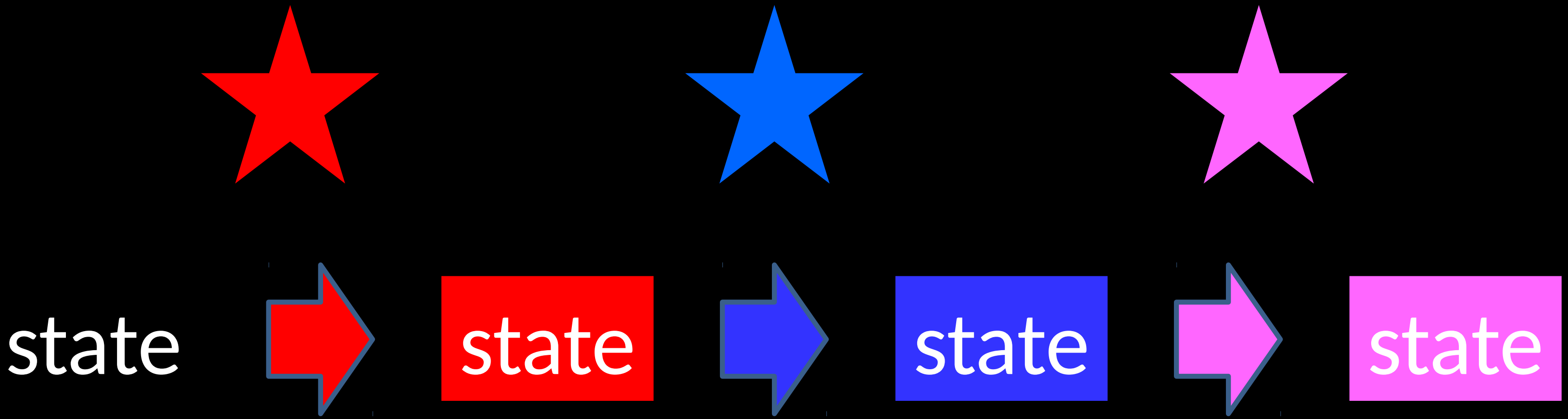
Miners assemble contracts …



Miners assemble contracts …

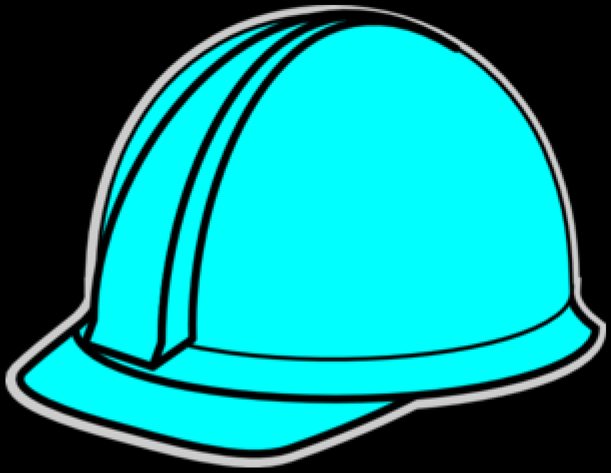


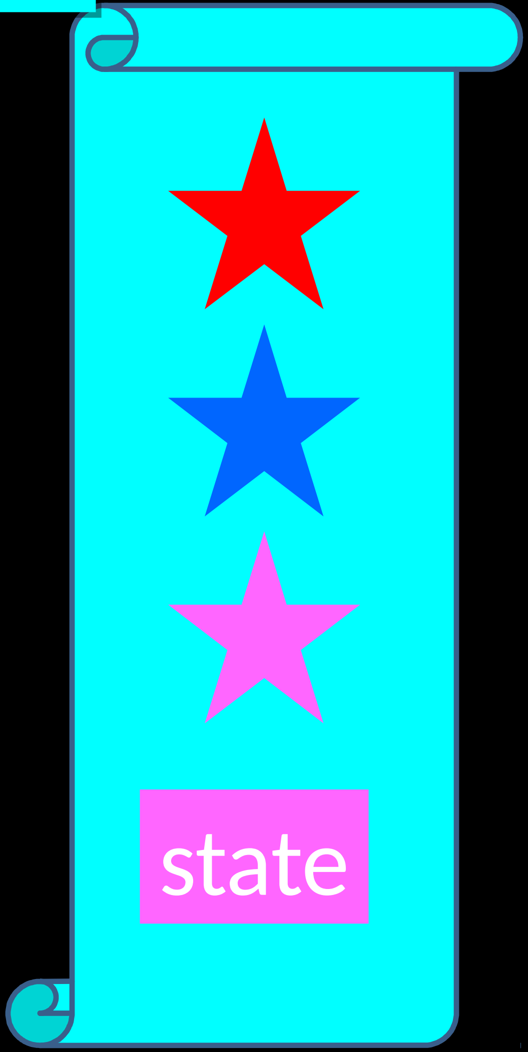
Apply them one-at-a-time to compute new state

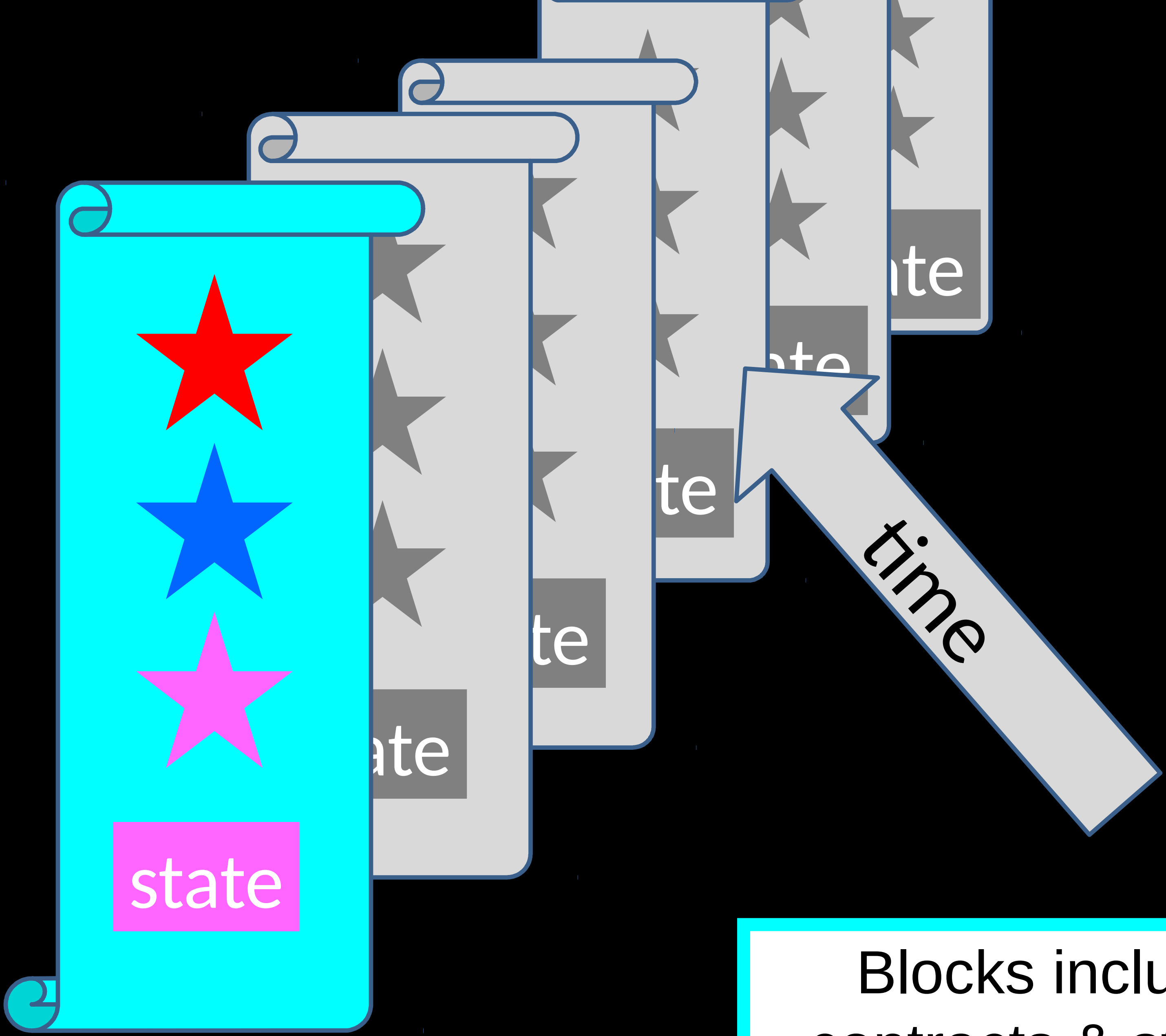


state → state → state → state

Block has contracts & new state

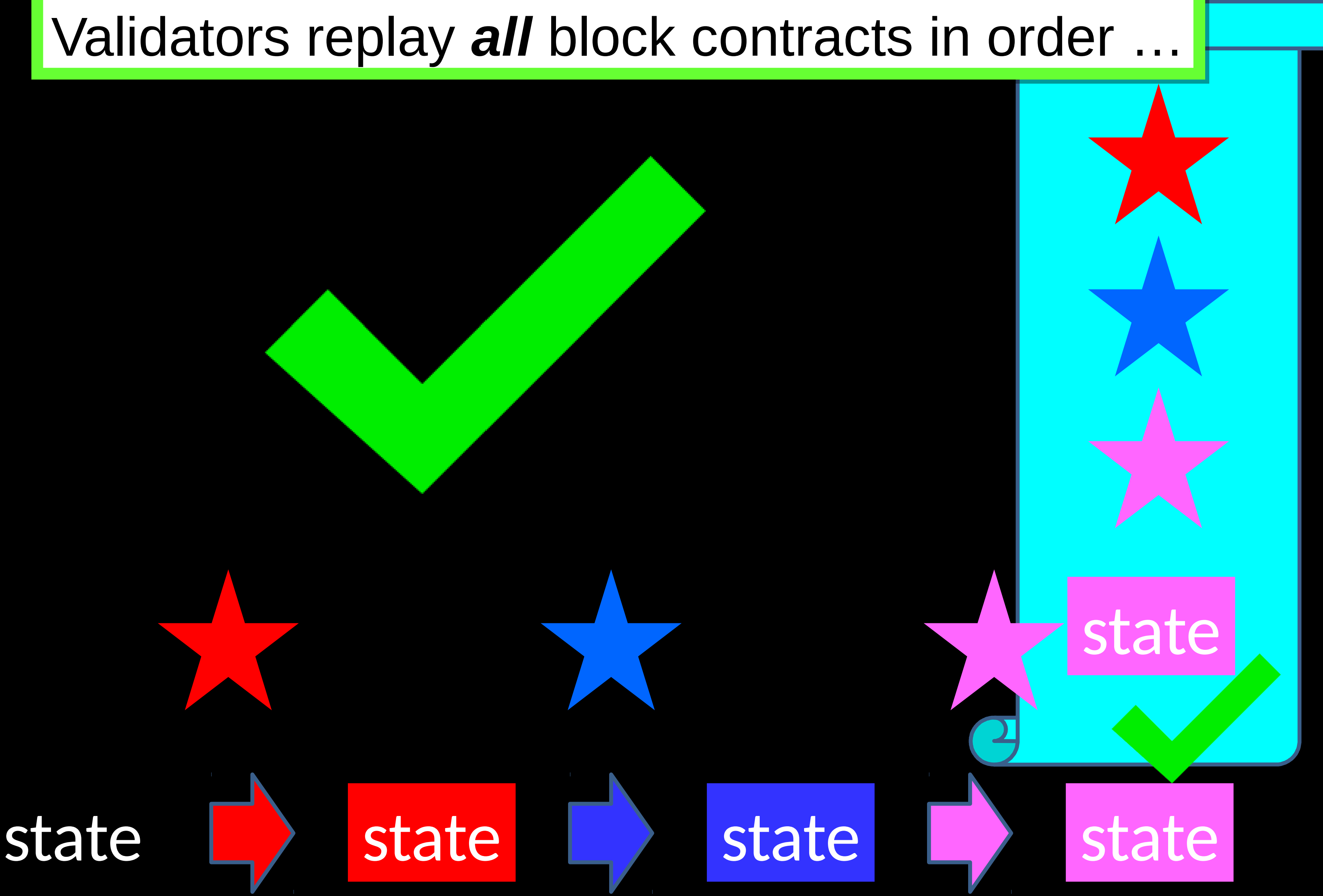




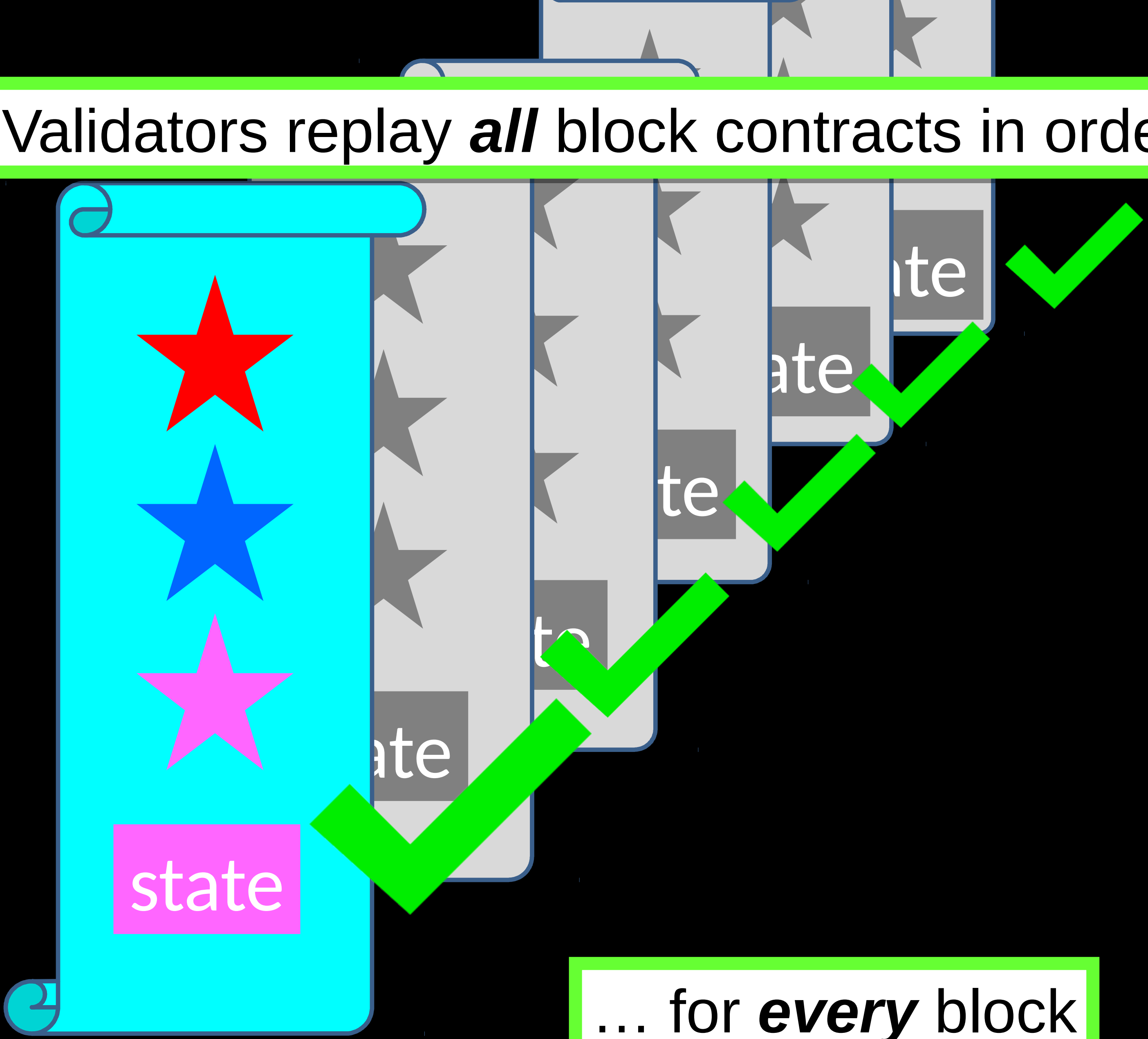


Blocks include contracts & states

Validators replay ***all*** block contracts in order …



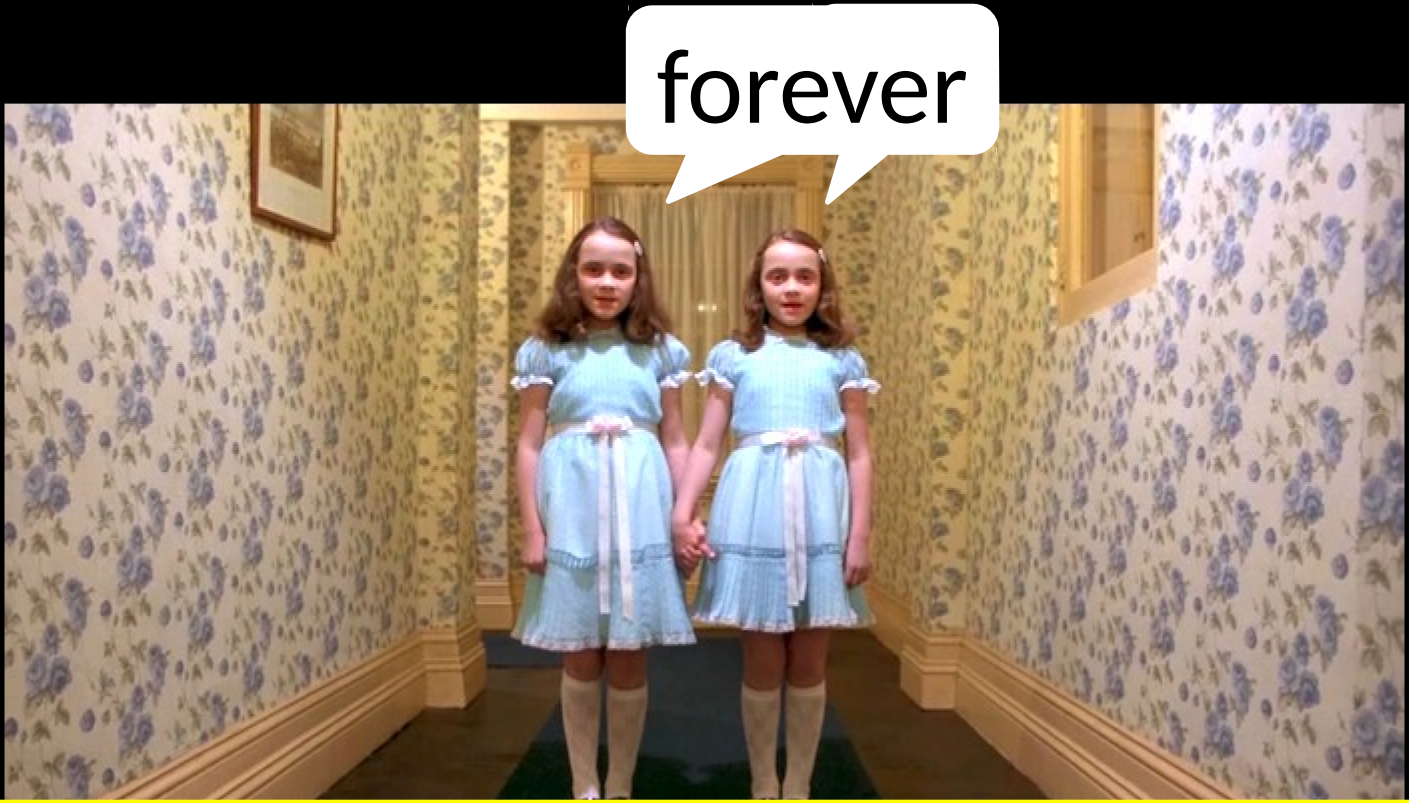
Validators replay ***all*** block contracts in order …



… for ***every*** block

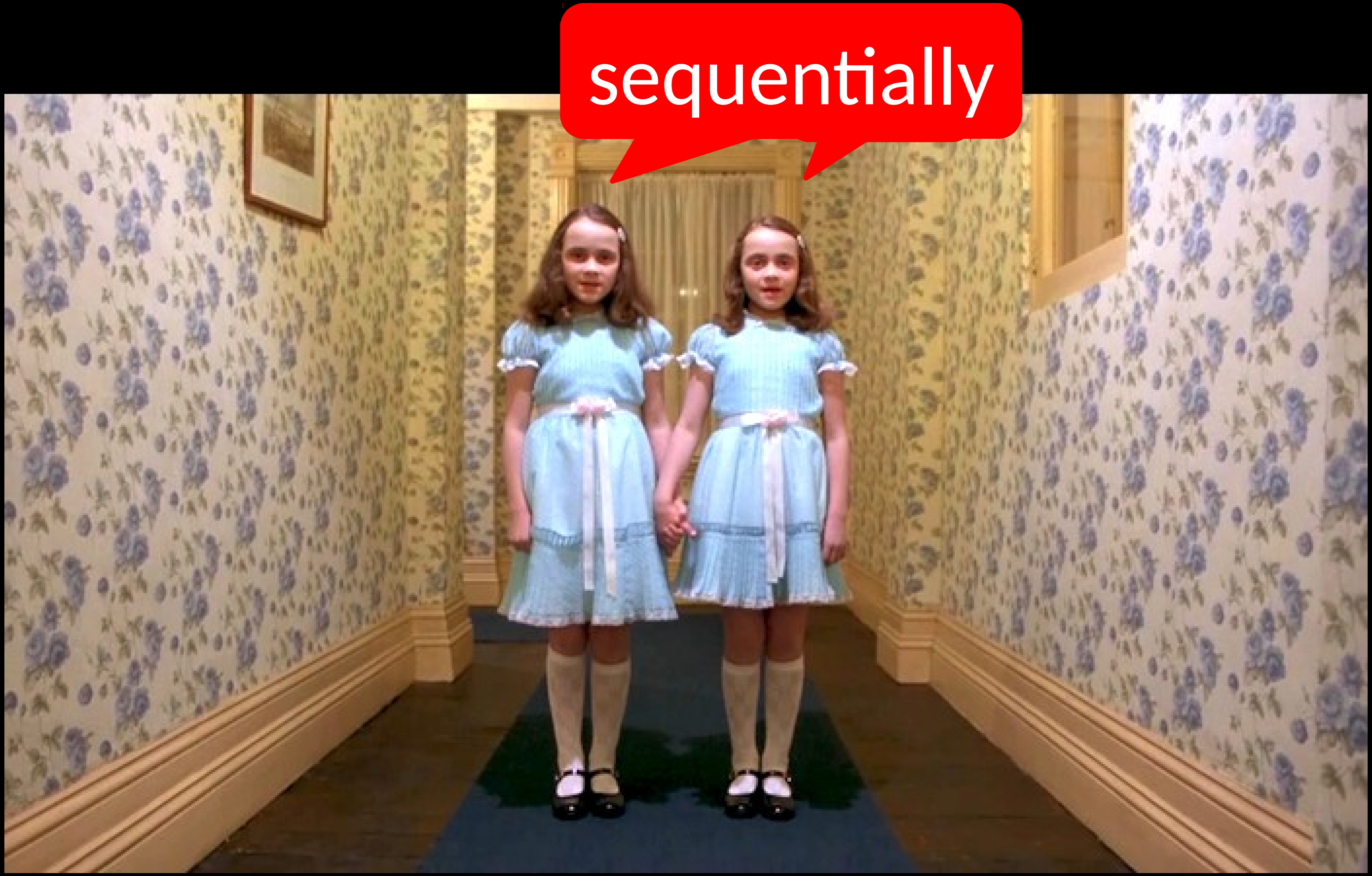
Contracts re-executed for How Long?

Contracts re-executed for How Long?

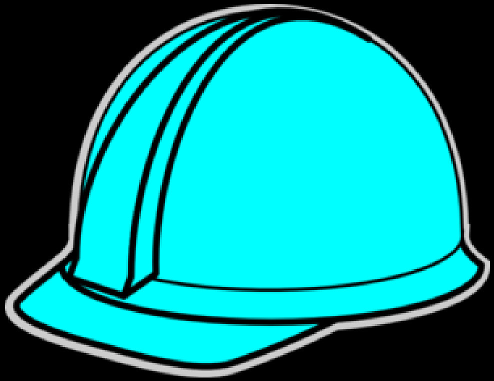


Every validator eventually executes every contract

Contracts re-executed for How Long?



Miners …



Execute block’s contracts **sequentially**

Paid by client per step

High latency = competitive disadvantage

Validators …

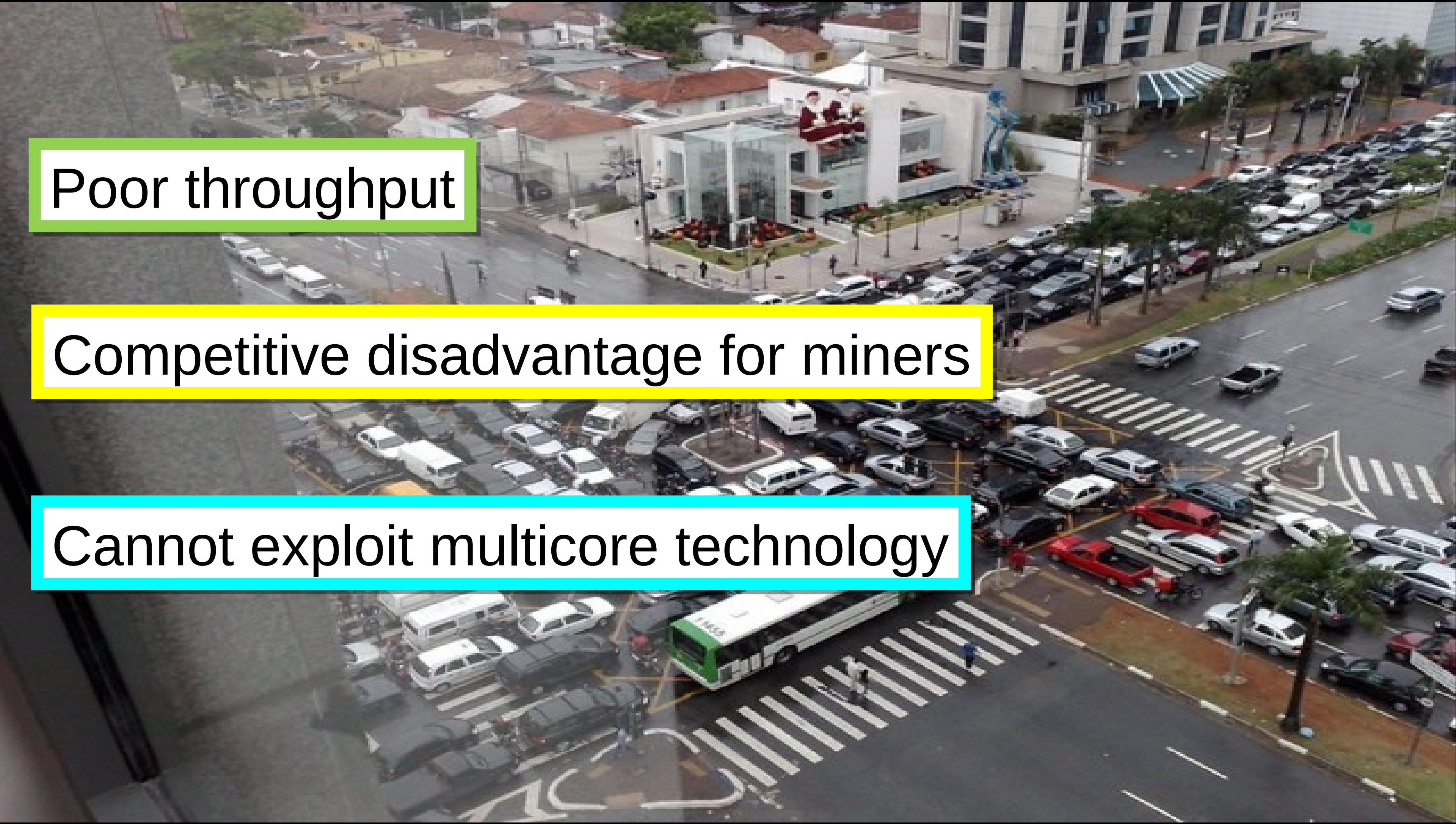
✓

Execute every block’s contracts **sequentially**

Not paid

Every validator, every contract, **forever**

Why is sequential execution so wrong?



Poor throughput

Competitive disadvantage for miners

Cannot exploit multicore technology

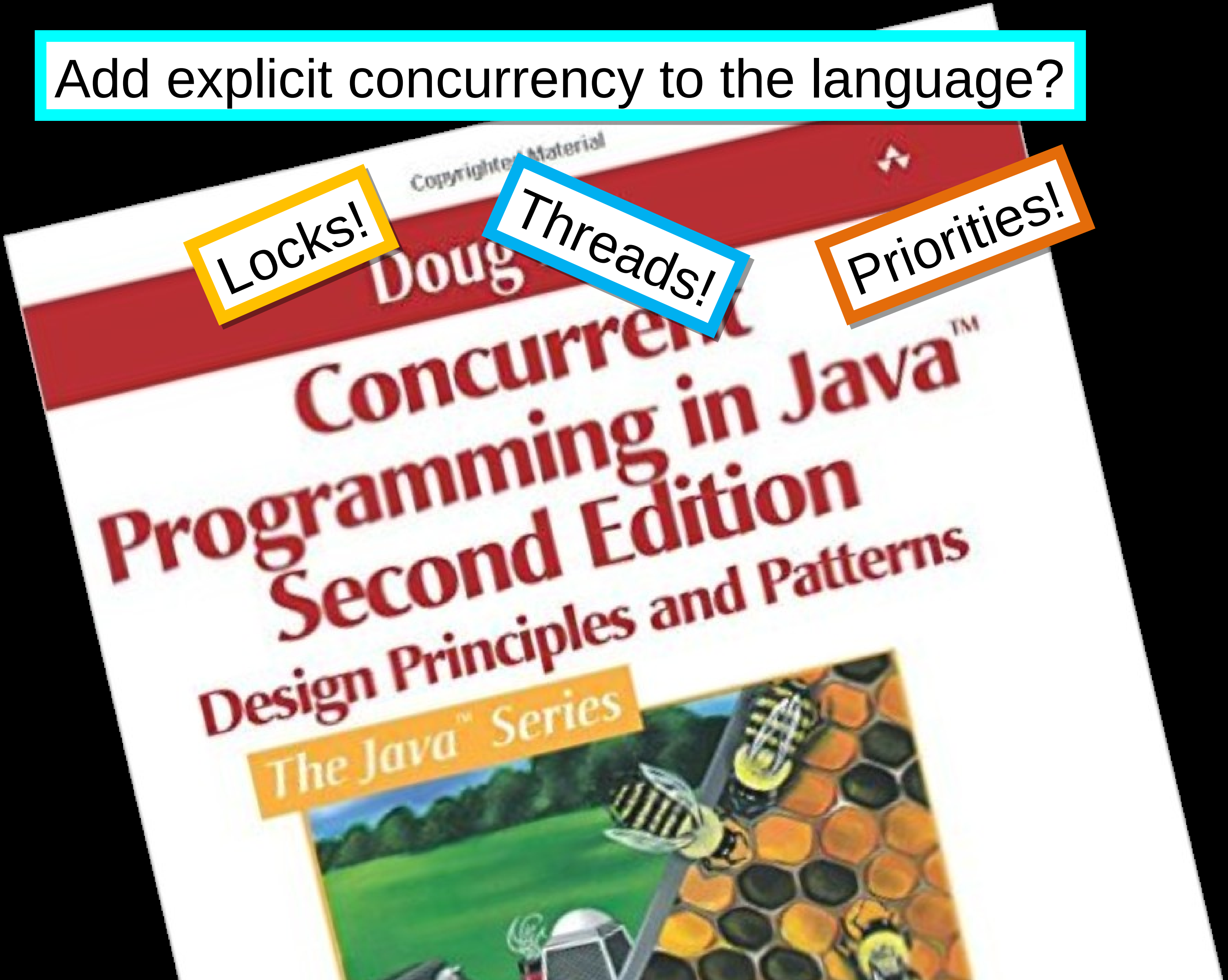
Naïve Concurrency?

Nope

Inconsistent shared state

Voters could vote twice

Add explicit concurrency to the language?



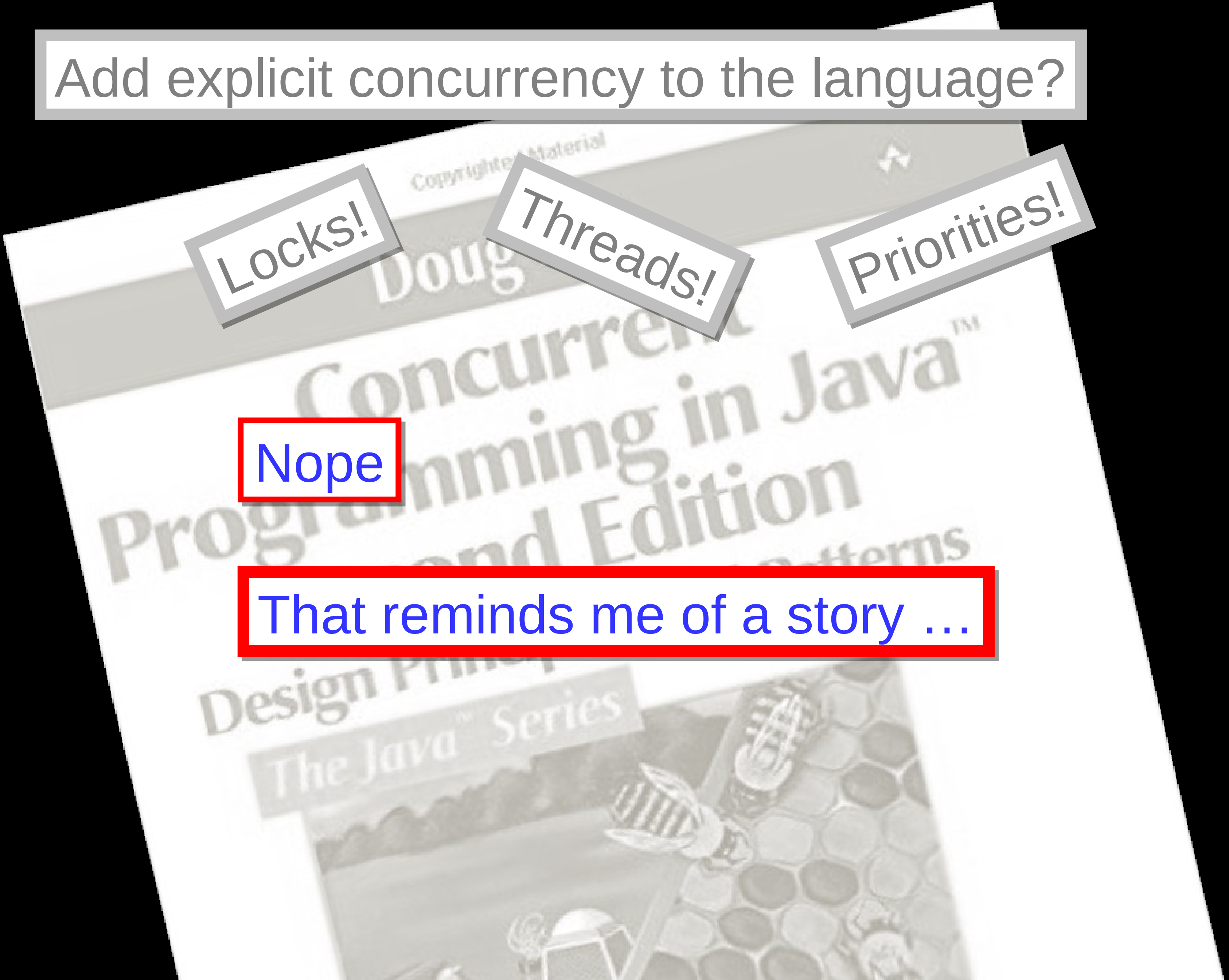
Locks!

Threads!

Priorities!

Copyrighted Material
Doug
Concurrent Programming in Java™
Second Edition
Design Principles and Patterns
The Java™ Series

Add explicit concurrency to the language?



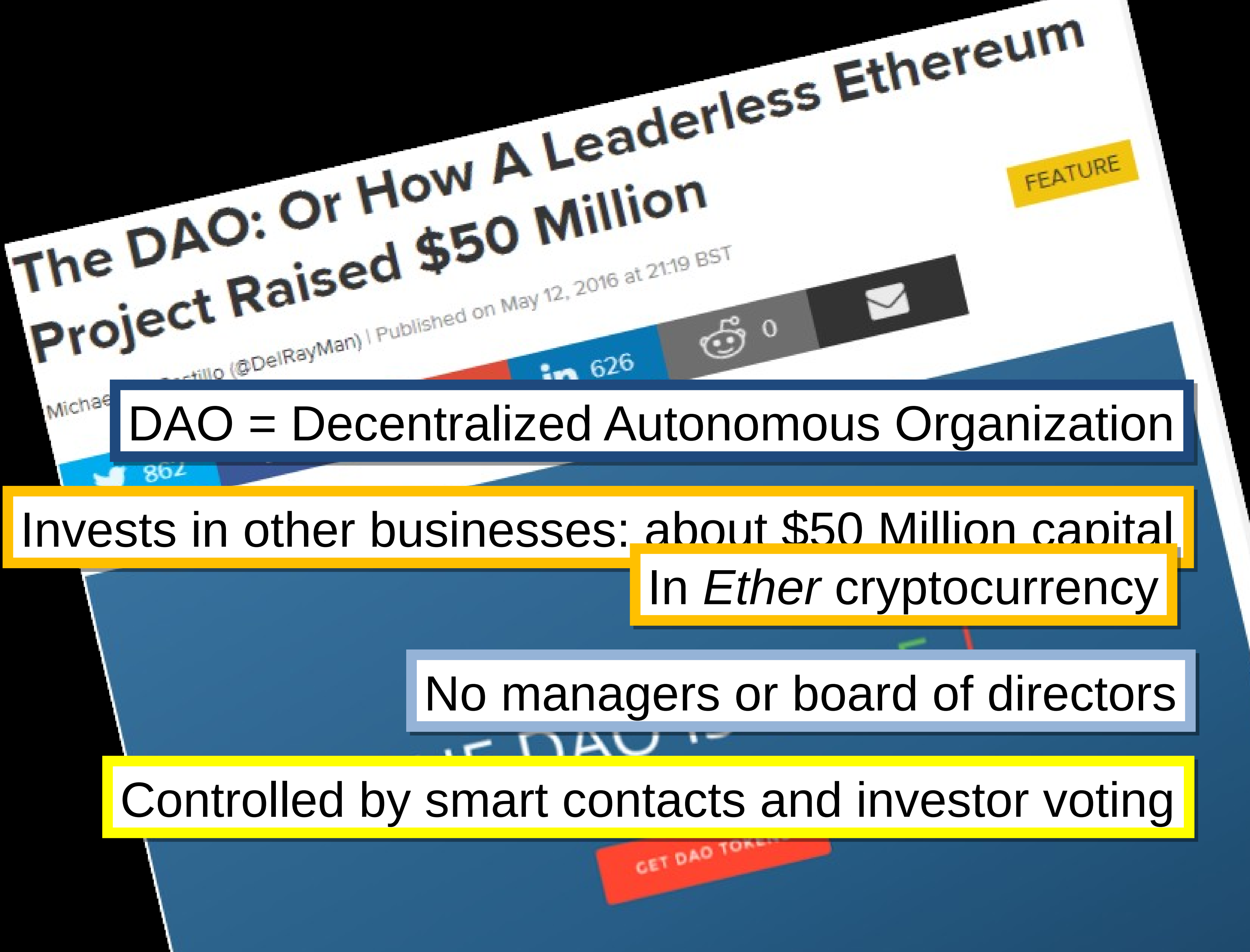
Locks!

Threads!

Priorities!

Nope

That reminds me of a story …



FEATURE

The DAO: Or How A Leaderless Ethereum Project Raised $50 Million

Michae rtillo (@DelRayMan) | Published on May 12, 2016 at 21:19 BST

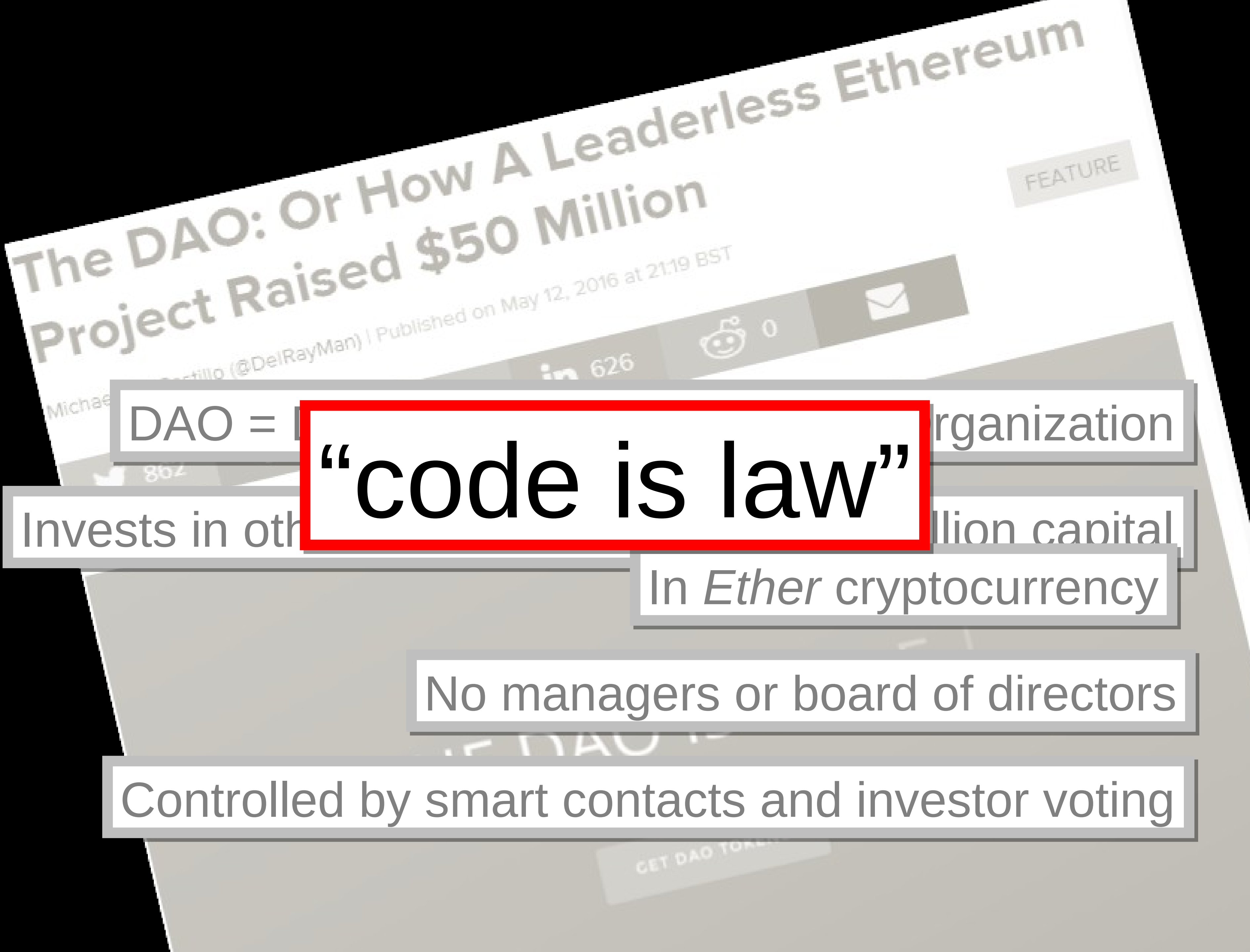
DAO = Decentralized Autonomous Organization

Invests in other businesses: about $50 Million capital

In *Ether* cryptocurrency

No managers or board of directors

Controlled by smart contacts and investor voting



The DAO: Or How A Leaderless Ethereum Project Raised $50 Million

FEATURE

Michae... ...stillo (@DelRayMan) | Published on May 12, 2016 at 21:19 BST

DAO = D ... rganization

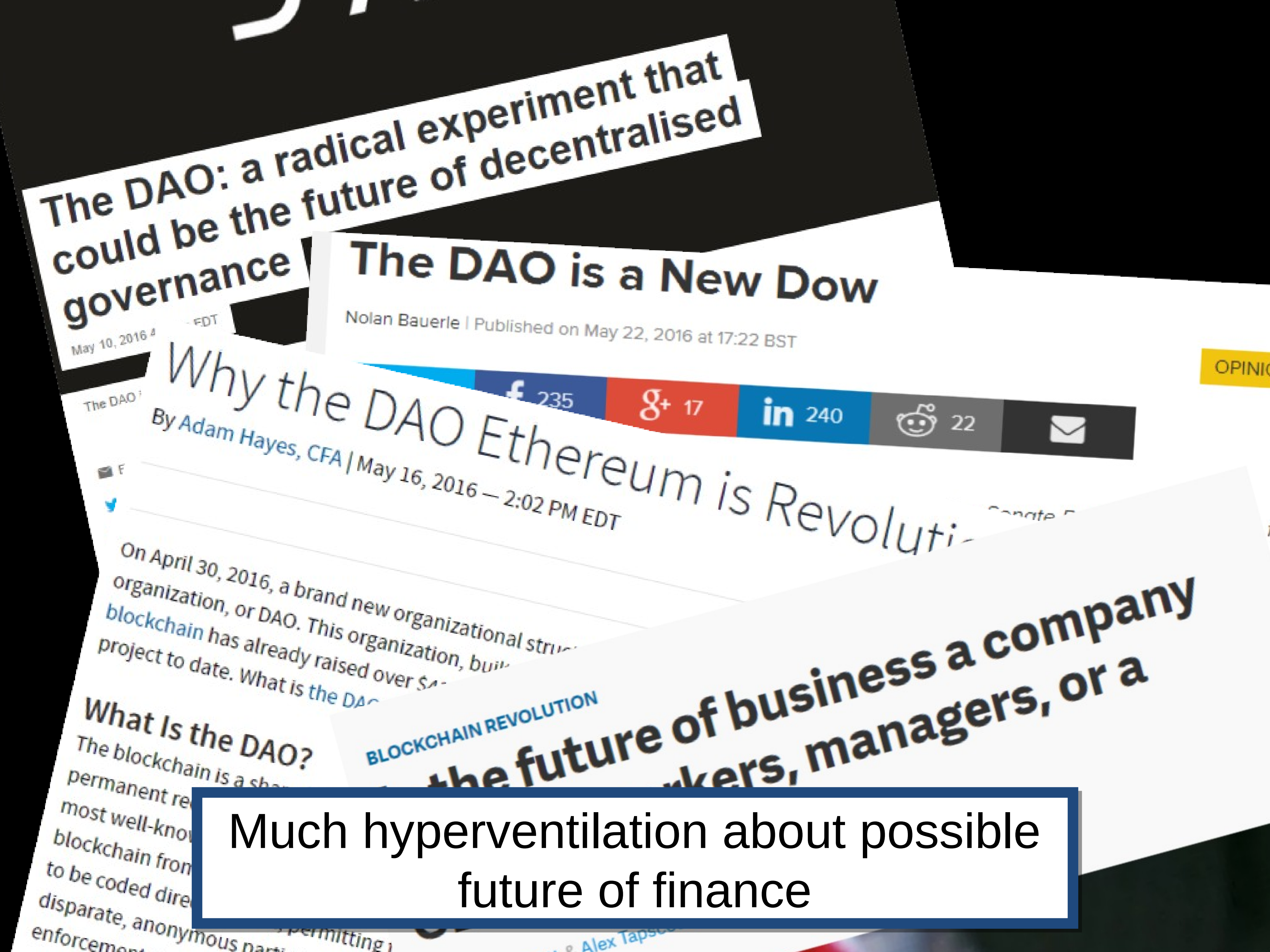
“code is law”

Invests in oth ... llion capital

In *Ether* cryptocurrency

No managers or board of directors

Controlled by smart contacts and investor voting



The DAO: a radical experiment that could be the future of decentralised governance

May 10, 2016 4 EDT

The DAO is a New Dow

Nolan Bauerle | Published on May 22, 2016 at 17:22 BST

Why the DAO Ethereum is Revoluti

By Adam Hayes, CFA | May 16, 2016 — 2:02 PM EDT

On April 30, 2016, a brand new organizational stru organization, or DAO. This organization, bui blockchain has already raised over $4 project to date. What is the DA

What Is the DAO?

The blockchain is a sh permanent re most well-kno blockchain fron to be coded dire disparate, anonymous parti permitting enforcemen

BLOCKCHAIN REVOLUTION

the future of business a company rkers, managers, or a

Alex Tapsc

Much hyperventilation about possible future of finance

Schematic DAO Code

```
function withdraw(uint amount) {
  client = msg.sender;
  if (balance[client] >= amount} {
    if (client.call.sendMoney(amount)) {
      balance[client] -= amount;
    }}}
```

Schematic DAO Code

```
function withdraw(uint amount) {
  client = msg.sender;
  if (balance[client] >= amount} {
    if (client.call.sendMoney(amount)) {
      balance[client] -= amount;
    }}}
```

Client wants to transfer own money

Schematic DAO Code

```
function withdraw(uint amount) {
  client = msg.sender;
  if (balance[client] >= amount} {
    if (client.call.sendMoney(amount)) {
      balance[client] -= amount;
    }}}
```

Which client?

Schematic DAO Code

```
function withdraw(uint amount) {
  client = msg.sender;
  if (balance[client] >= amount} {
    if (client.call.sendMoney(amount)) {
      balance[client] -= amount;
    }}}
```

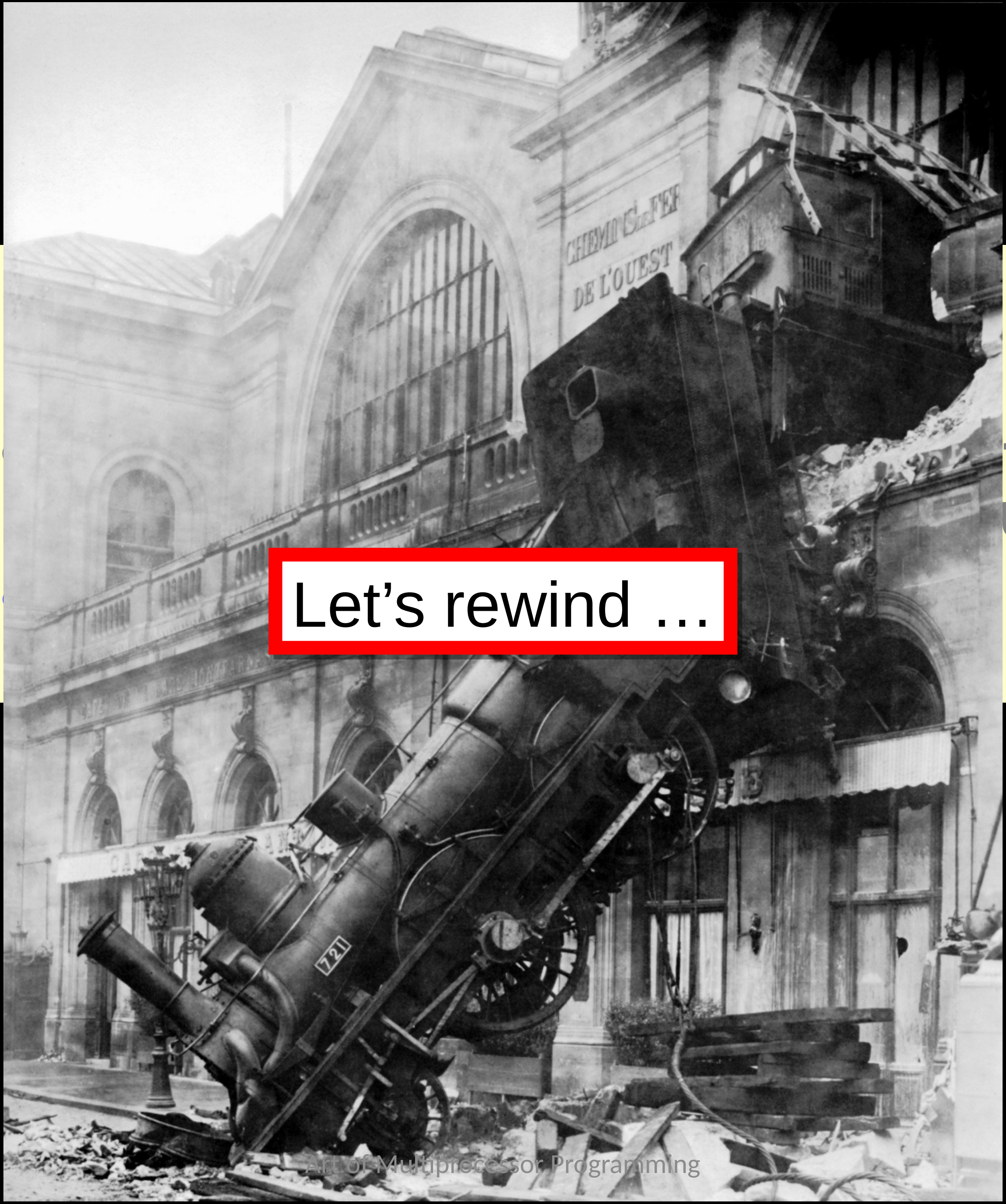
Does client have enough money?

Schematic DAO Code

```
function withdraw(uint amount) {
  client = msg.sender;
  if (balance[client] >= amount} {
    if (client.call.sendMoney(amount)) {
      balance[client] -= amount;
    }}}
```

Transfer the money by calling another contract …

```
functio
  clien
  if (b                          {
    if                          unt)) {
      b
    }}}
```



Let’s rewind …

```
function withdraw(uint amount) {
  client = msg.sender;
  if (balance[client] >= amount} {
    if (client.call.sendMoney(amount)) {
      balance[client] -= amount;
    }}}
```

Transfer the money by calling another contract …

```
function sendMoney(uint amount) {
  balance += amount
  msg.sender.call.transfer(amount)
  ...
}
```

```
function withdraw(uint amount) {
  client = msg.sender;
  if (balance[client] >= amount} {
    if (client.call.sendMoney(amount)) {
      balance[client] -= amount;
    }}}
```

Credit account

```
function sendMoney(uint amount) {
  balance += amount
  msg.sender.call.transfer(amount)
  ...
}
```

```
function withdraw(uint amount) {
  client = msg.sender;
  if (balance[client] >= amount} {
    if (client.call.sendMoney(amount)) {
      balance[client] -= amount;
    }}}
```

Wait, what?

Client makes re-entrant withdraw request!

```
function sendMoney(uint amount) {
  balance += amount
  msg.sender.call.withdraw(amount)
  ...
}
```

```
function withdraw(uint amount) {
  client = msg.sender;
  if (balance[client] >= amount} {
    if (client.call.sendMoney(amount)) {
      balance[client] -= amount;
    }}}
```

```
function sendMoney(uint amount) {
  balance += amount
  msg.sender.call.withdraw(amount)
  ...
}
```

```
function withdraw(uint amount) {
  client = msg.sender;
  if (balance[client] >= amount} {
    if (client.call.sendMoney(amount)) {
      balance[client] -= amount;
    }}}
```

Second time around, balance still looks OK …

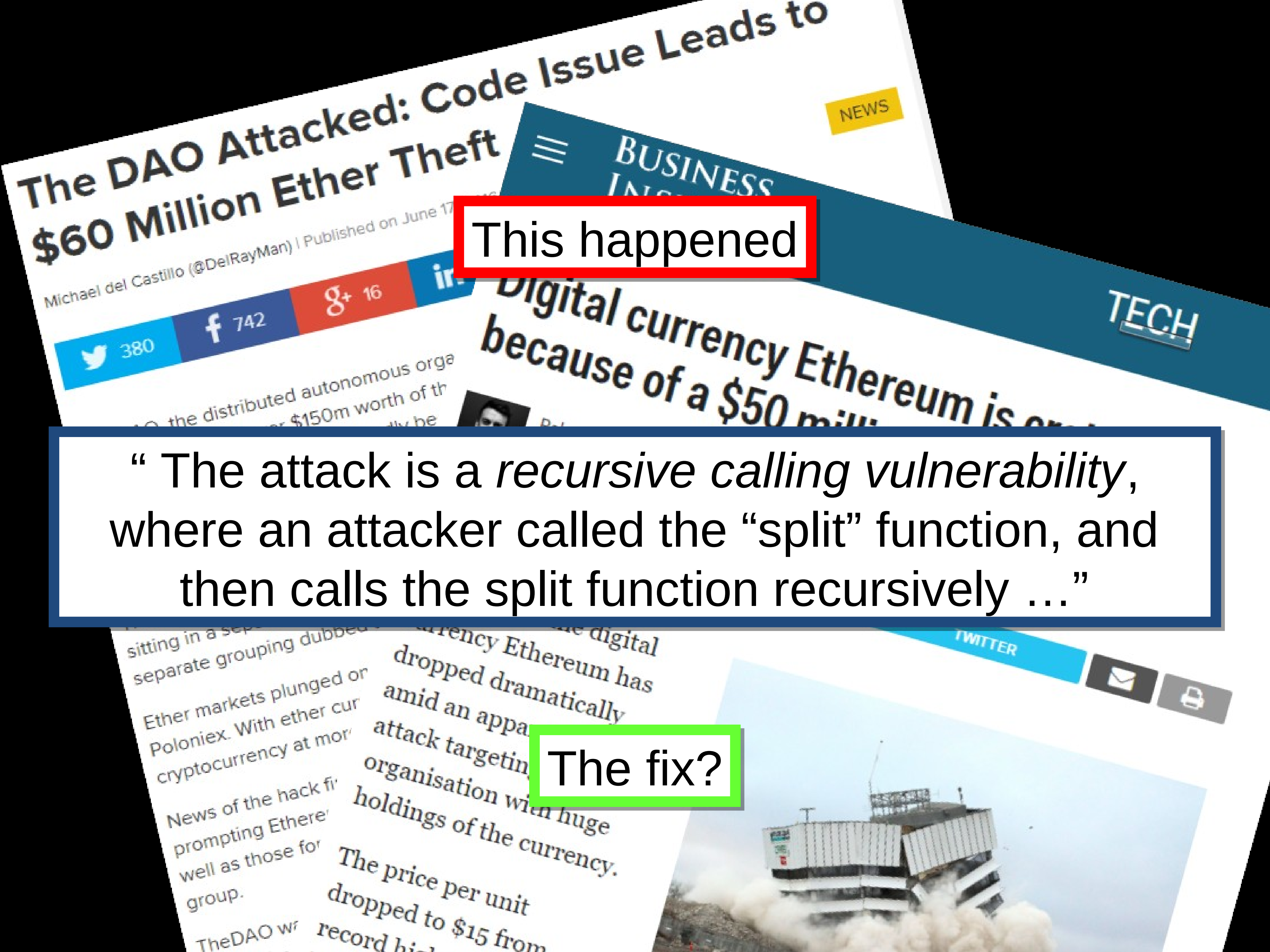
```
function sendMoney(uint amount) {
  balance += amount
  msg.sender.call.withdraw(amount)
  ...
}
```

```
function withdraw(uint amount) {
  client = msg.sender;
  if (balance[client] >= amount} {
    if (client.call.sendMoney(amount)) {
      balance[client] -= amount;
    }}}
```

Send money again …

and again and again …

```
function sendMoney(uint amount) {
  balance += amount
  msg.sender.call.withdraw(amount)
  ...
}
```



The DAO Attacked: Code Issue Leads to $60 Million Ether Theft

NEWS

Michael del Castillo (@DelRayMan) | Published on June 17

380 742 16

...O, the distributed autonomous orga... ...r $150m worth of th... ...lly be...

...sitting in a sep... separate grouping dubbed...

Ether markets plunged on... Poloniex. With ether cur... cryptocurrency at mor...

News of the hack fi... prompting Ethere... well as those for... group.

TheDAO w...

BUSINESS INS...

TECH

Digital currency Ethereum is cra... because of a $50 milli...

...rrency Ethereum has ...e digital dropped dramatically amid an appa... attack targetin... organisation with huge holdings of the currency.

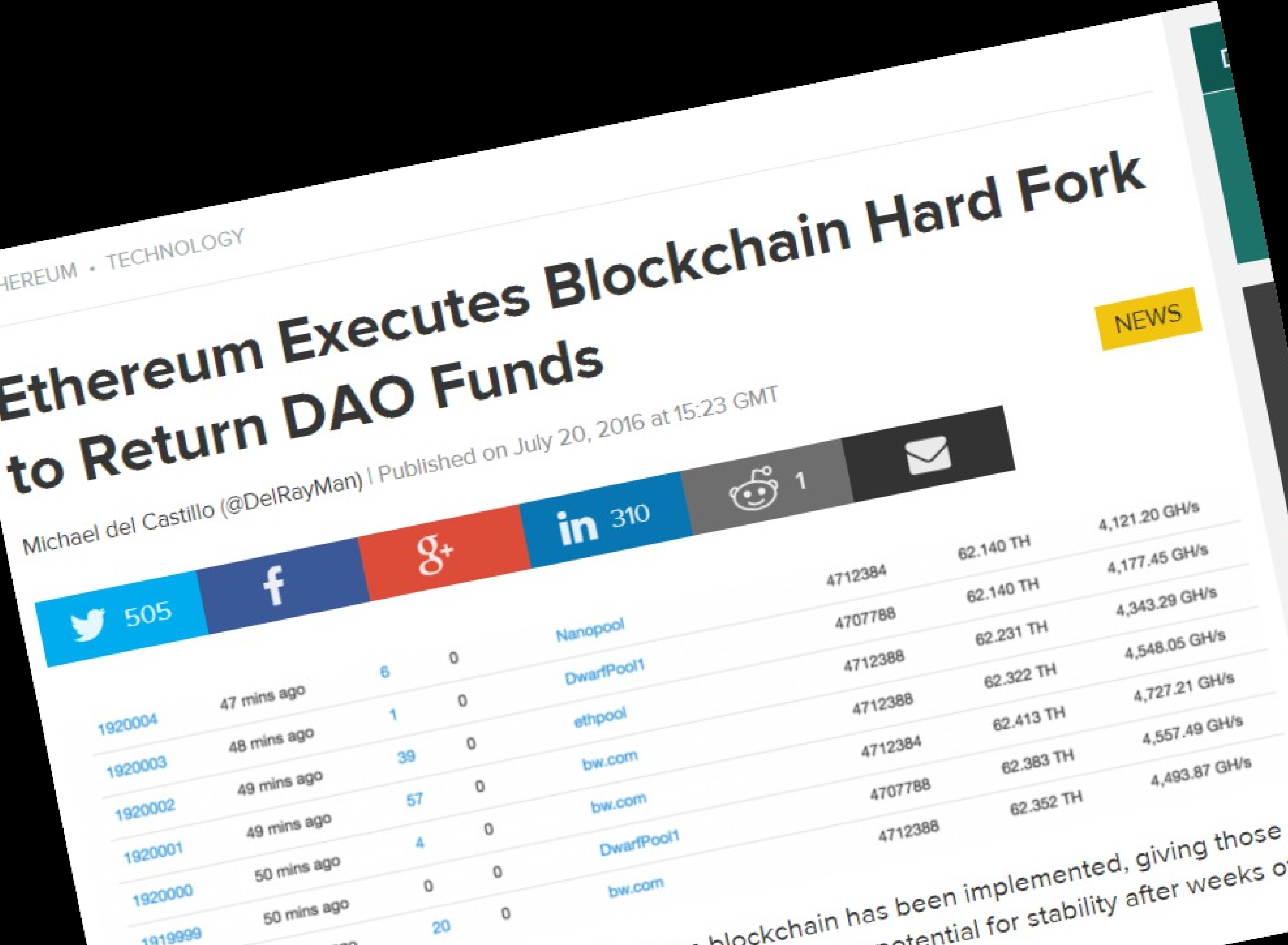
The price per unit dropped to $15 from record hi...

TWITTER

This happened

“ The attack is a *recursive calling vulnerability*, where an attacker called the “split” function, and then calls the split function recursively …”

The fix?



HEREUM • TECHNOLOGY

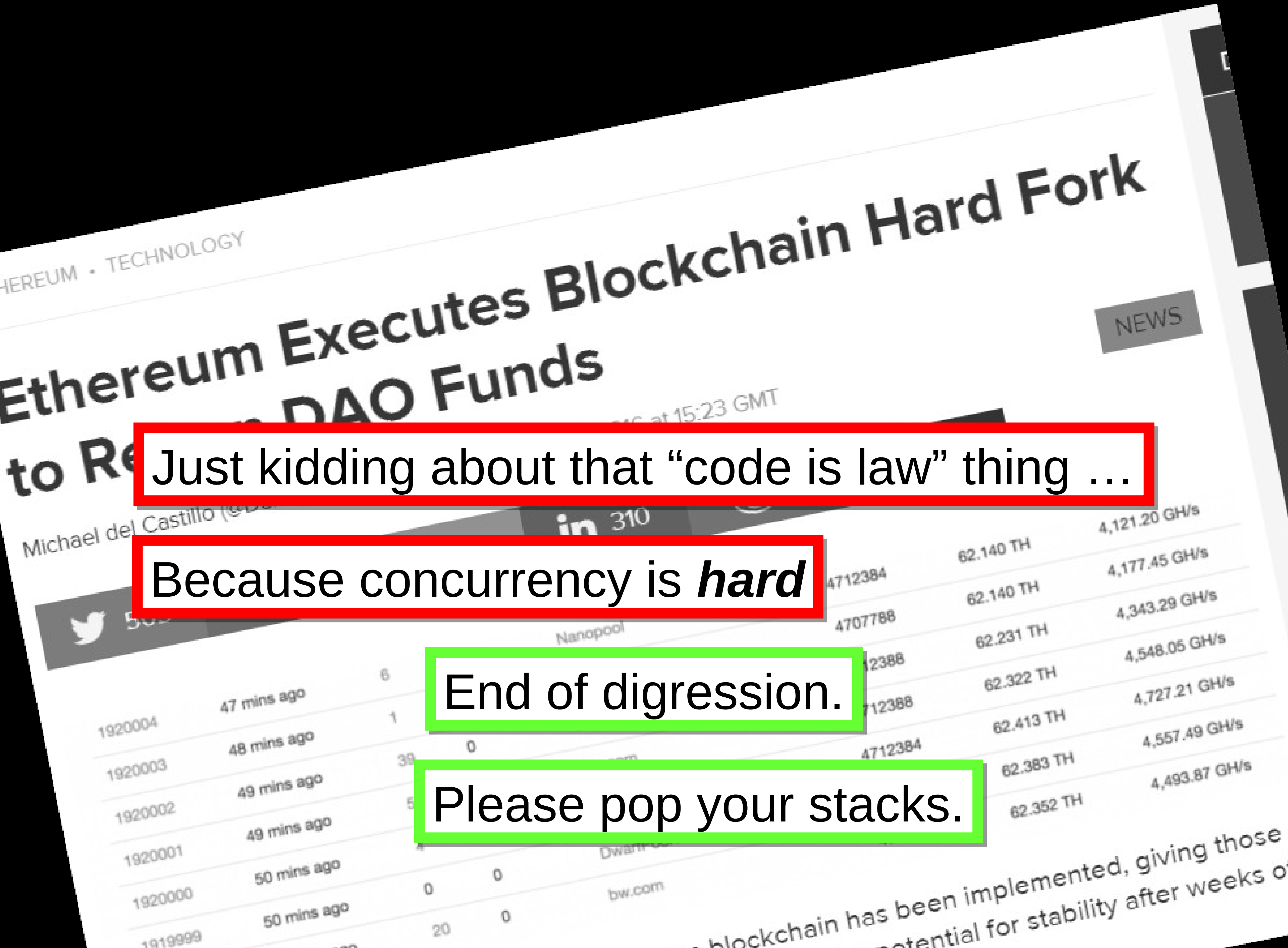
NEWS

Ethereum Executes Blockchain Hard Fork to Return DAO Funds

Michael del Castillo (@DelRayMan) | Published on July 20, 2016 at 15:23 GMT

1920004	47 mins ago	6	0	Nanopool	4712384	62.140 TH	4,121.20 GH/s
1920003	48 mins ago	1	0	DwarfPool1	4707788	62.140 TH	4,177.45 GH/s
1920002	49 mins ago	39	0	ethpool	4712388	62.231 TH	4,343.29 GH/s
1920001	49 mins ago	57	0	bw.com	4712388	62.322 TH	4,548.05 GH/s
1920000	50 mins ago	4	0	bw.com	4712384	62.413 TH	4,727.21 GH/s
1919999	50 mins ago	0	0	DwarfPool1	4707788	62.383 TH	4,557.49 GH/s
		20	0	bw.com	4712388	62.352 TH	4,493.87 GH/s

blockchain has been implemented, giving those
otential for stability after weeks o



HEREUM · TECHNOLOGY

NEWS

Ethereum Executes Blockchain Hard Fork to Re DAO Funds

16 at 15:23 GMT

Michael del Castillo (@D

50

in 310

1920004	47 mins ago	6		Nanopool	4712384	62.140 TH	4,121.20 GH/s
1920003	48 mins ago	1			4707788	62.140 TH	4,177.45 GH/s
1920002	49 mins ago	39	0		12388	62.231 TH	4,343.29 GH/s
1920001	49 mins ago	5		om	712388	62.322 TH	4,548.05 GH/s
1920000	50 mins ago	4			4712384	62.413 TH	4,727.21 GH/s
1919999	50 mins ago	0	0	DwarfPool		62.383 TH	4,557.49 GH/s
	ago	20	0	bw.com		62.352 TH	4,493.87 GH/s

blockchain has been implemented, giving those otential for stability after weeks o

Just kidding about that “code is law” thing …

Because concurrency is ***hard***

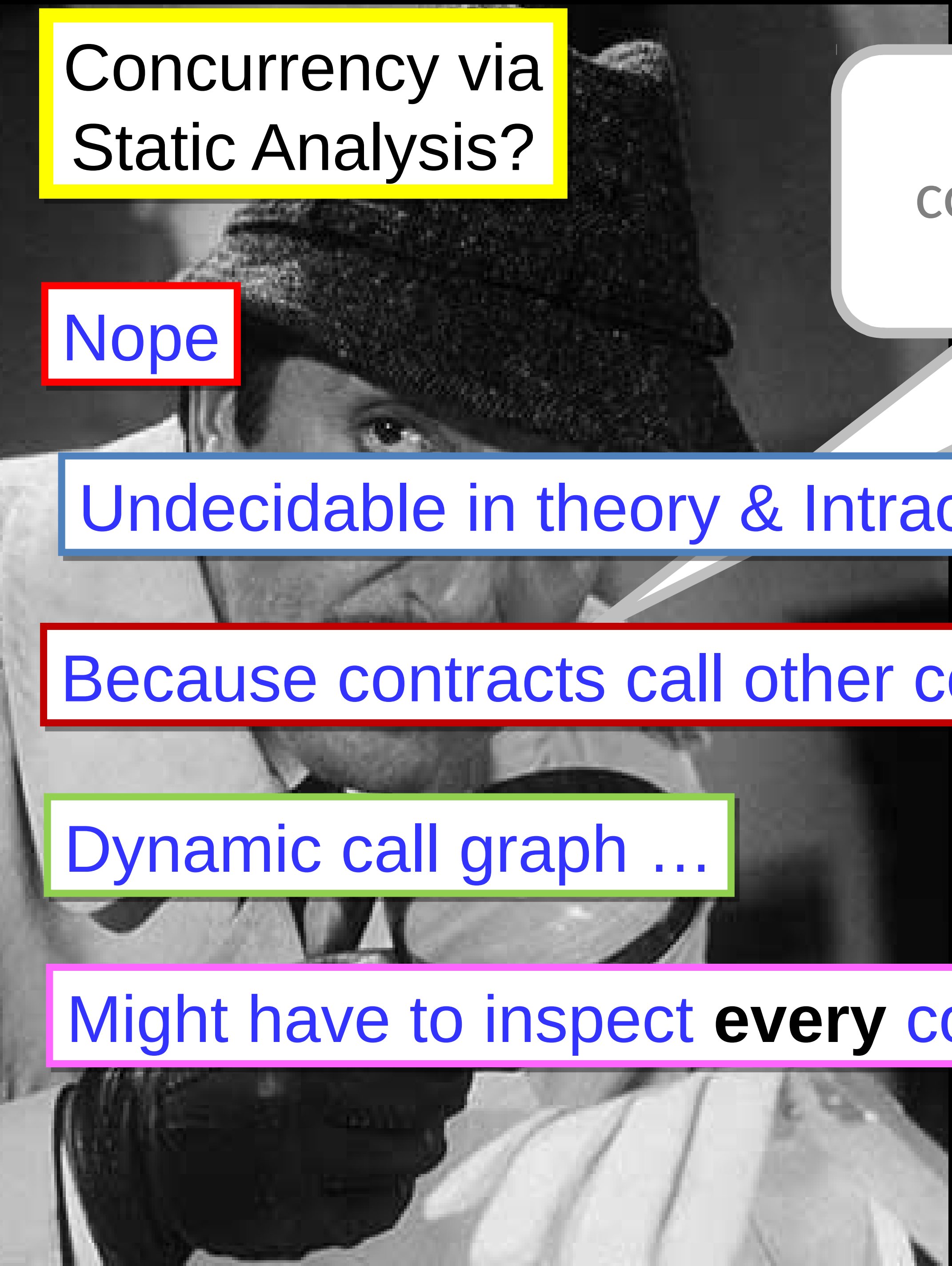
End of digression.

Please pop your stacks.

Concurrency via Static Analysis?

These contracts never conflict, so it’s safe to run them concurrently

Concurrency via Static Analysis?



These contracts never conflict, so it’s safe to run them concurrently

Nope

Undecidable in theory & Intractable in practice

Because contracts call other contracts …

Dynamic call graph …

Might have to inspect **every** contract **ever**

Transactions

Transactional Memory

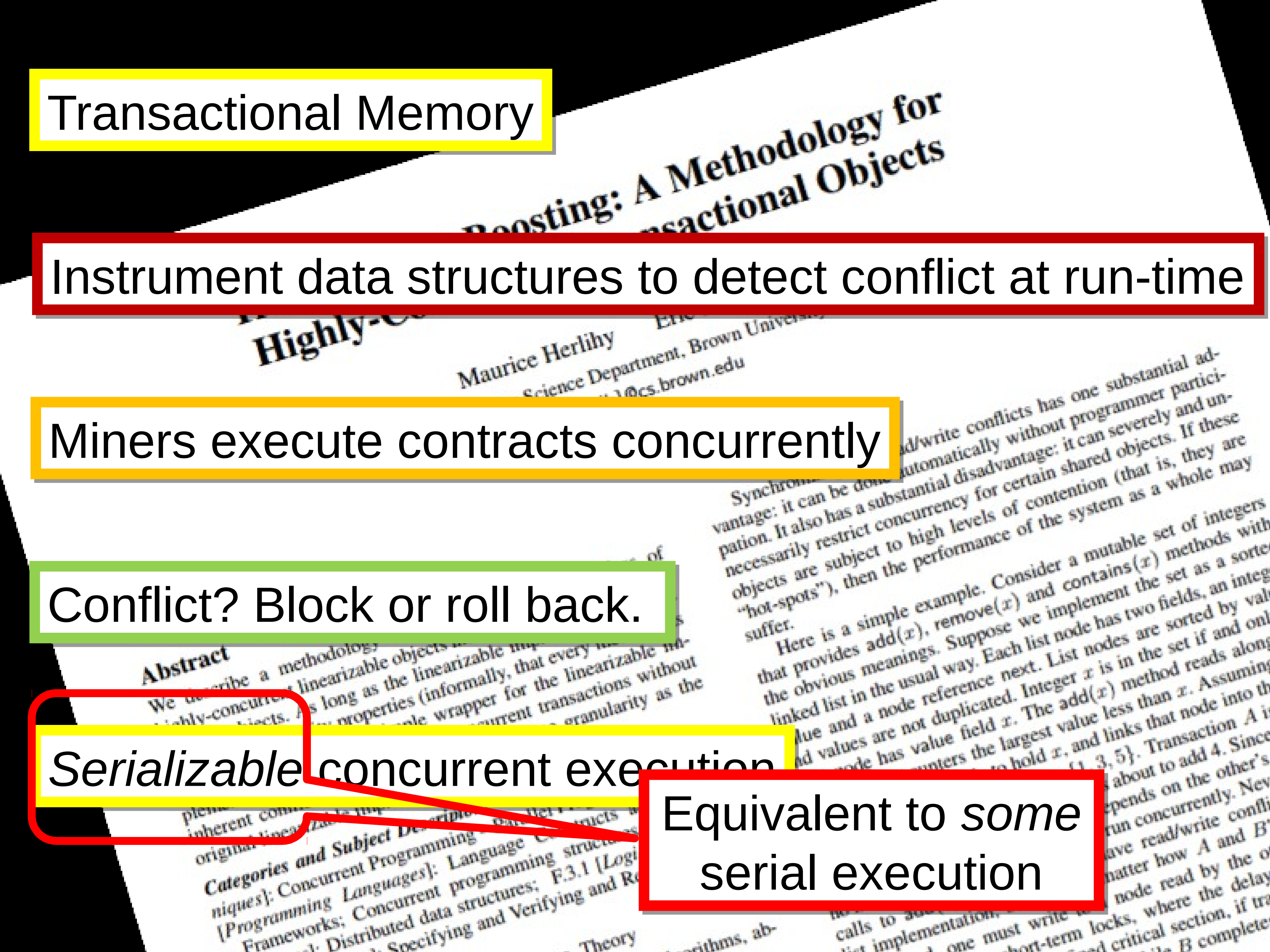
Instrument data structures to detect conflict at run-time

Miners execute contracts concurrently

Conflict? Block or roll back.

Serializable concurrent execution

Equivalent to *some* serial execution



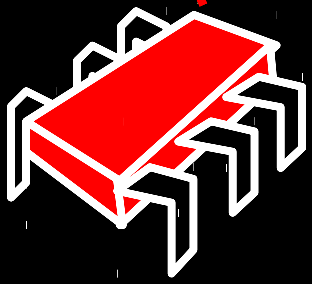
Boosting: A Methodology for
sactional Objects
Highly-C

Maurice Herlihy
Science Department, Brown Univers
@cs.brown.edu

Abstract

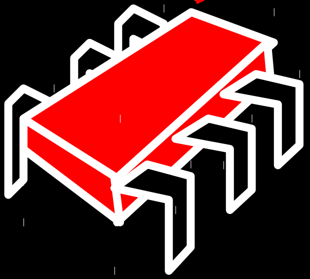
```
balance["Alice"] += sum
```

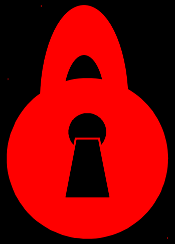
Intention



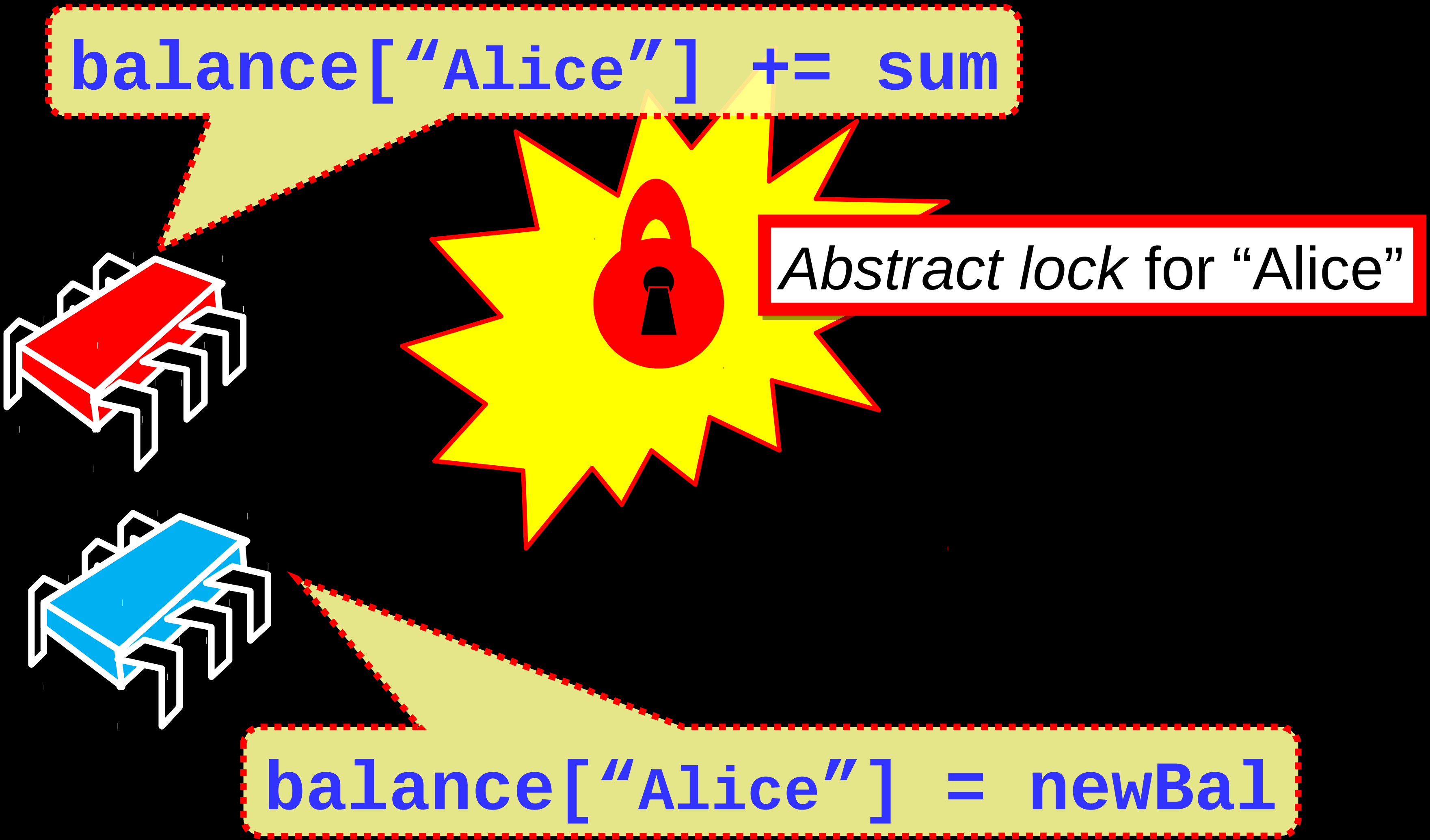
Miner thread

```
balance[“Alice”] += sum
```





Abstract lock for “Alice”

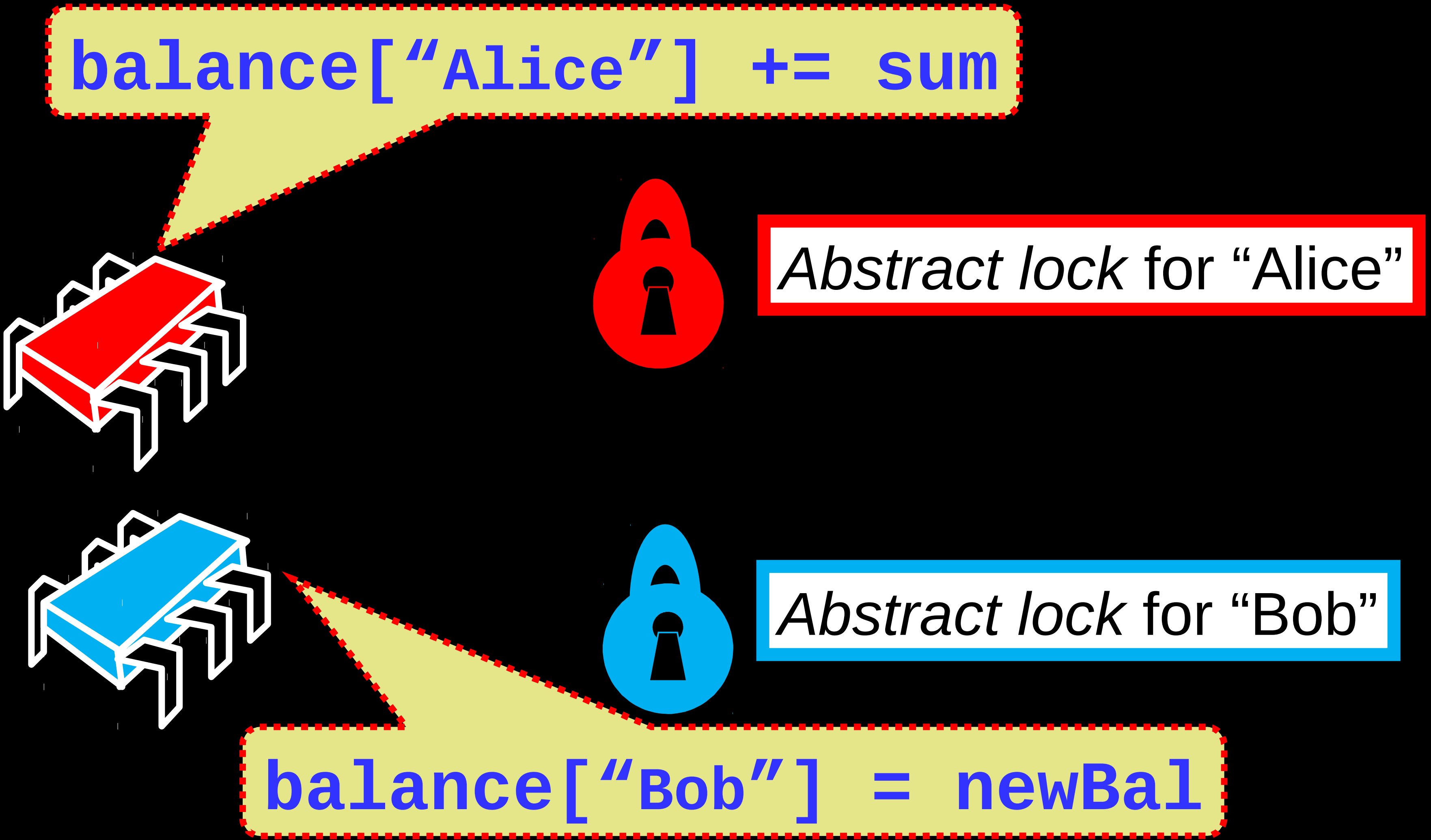


```
balance[“Alice”] += sum
```

Abstract lock for “Alice”

```
balance[“Alice”] = newBal
```

Blocks operations that do not ***commute***



```
balance[“Alice”] += sum
```

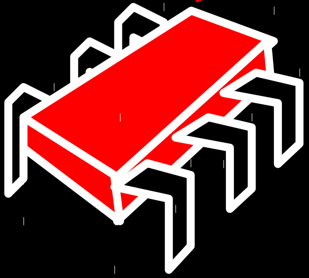
Abstract lock for “Alice”

Abstract lock for “Bob”

```
balance[“Bob”] = newBal
```

Allow concurrent operations that do ***commute***

```
balance[“Alice”] += sum
```



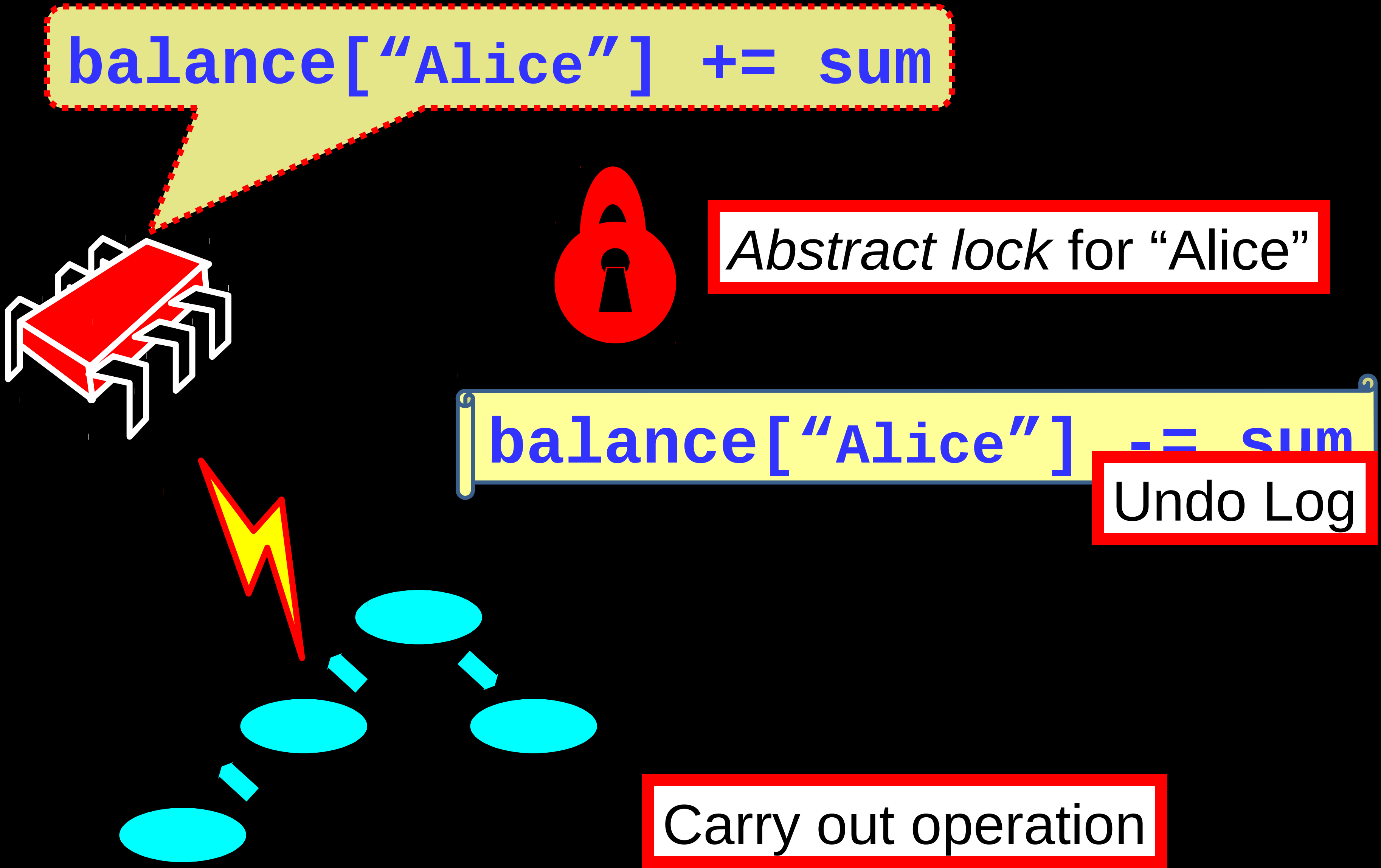
Abstract lock for “Alice”

```
balance[“Alice”] -= sum
```

Undo Log

Register ***inverse*** in undo log

Undoes updates to “Alice” only



```
balance[“Alice”] += sum
```

Abstract lock for “Alice”

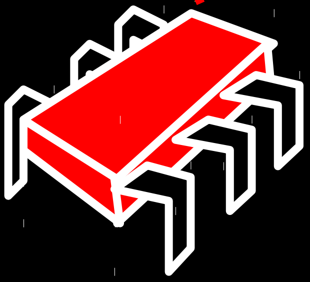
```
balance[“Alice”] -= sum
```

Undo Log

Carry out operation

Success (commit)

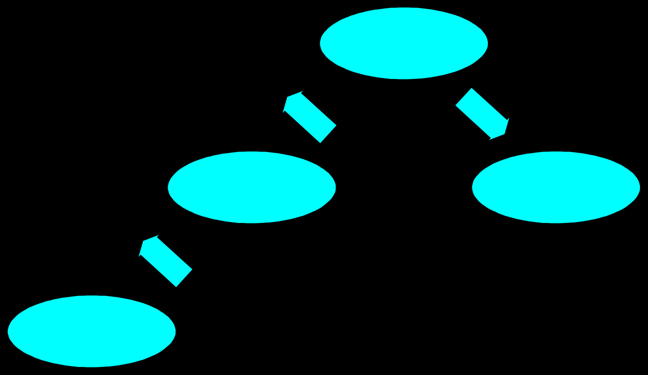
```
balance[“Alice”] += sum
```

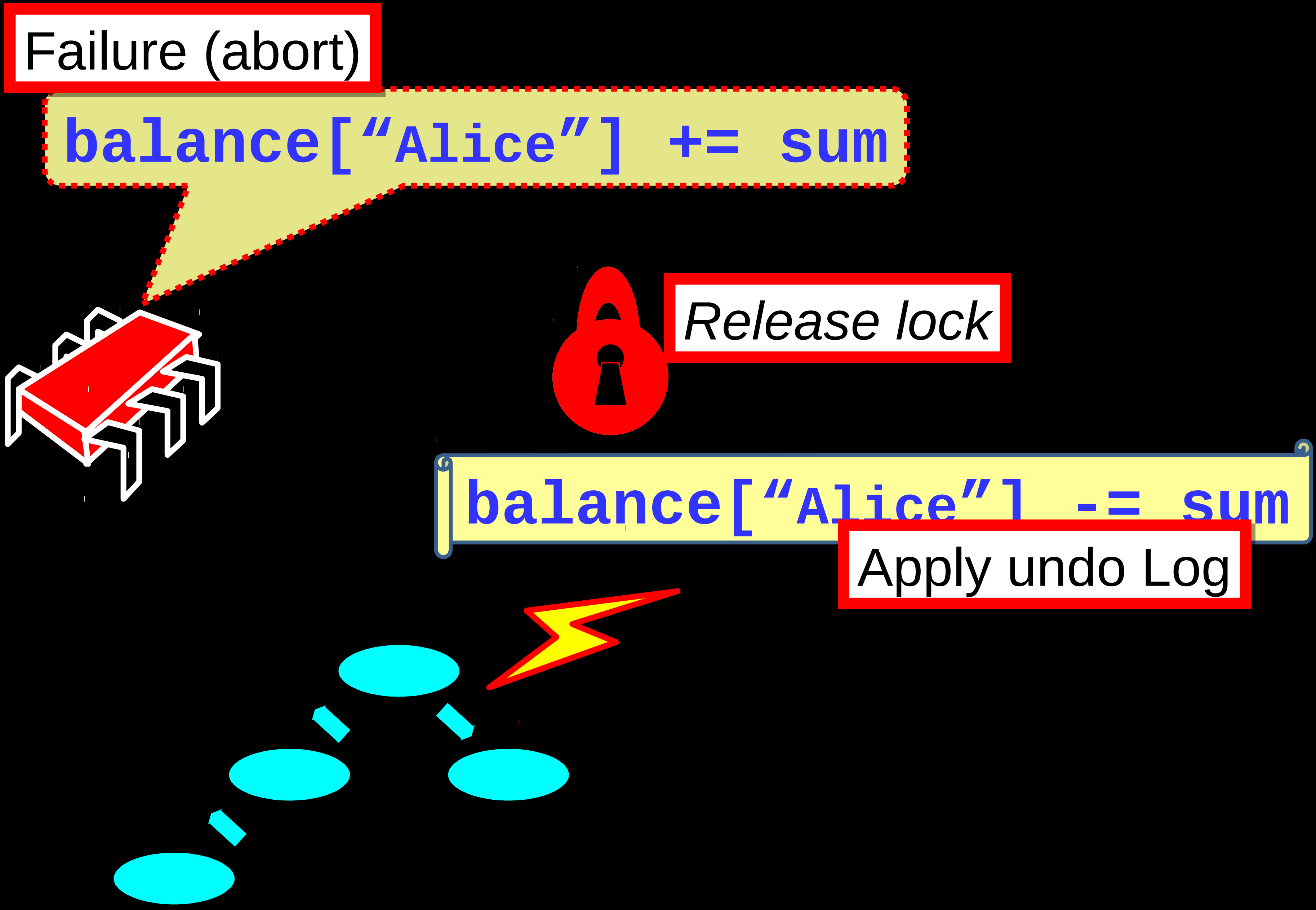


Release lock

```
balance[“Alice”] -= sum
```

Discard Log





Failure (abort)

```
balance["Alice"] += sum
```

Release lock

```
balance["Alice"] -= sum
```

Apply undo Log

Risks to miners

Conflict resolution means delay

Delay = competitive disadvantage vs rivals

Not paid for aborted steps

Benefits to miners

Low conflict means low latency

Low latency = competitive advantage vs rivals

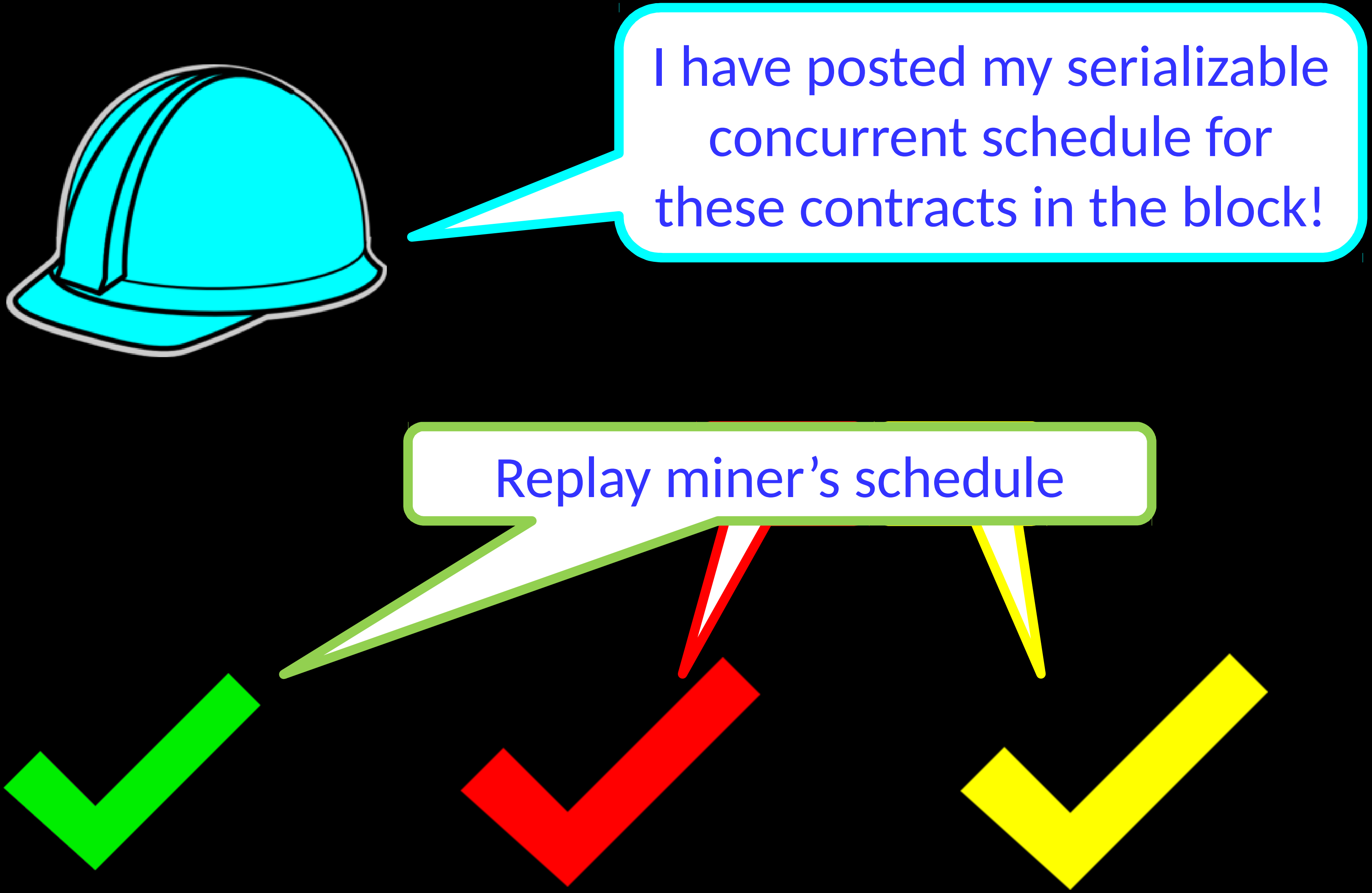
Lower energy, better HW usage, etc.

Validators

Cannot mimic miners

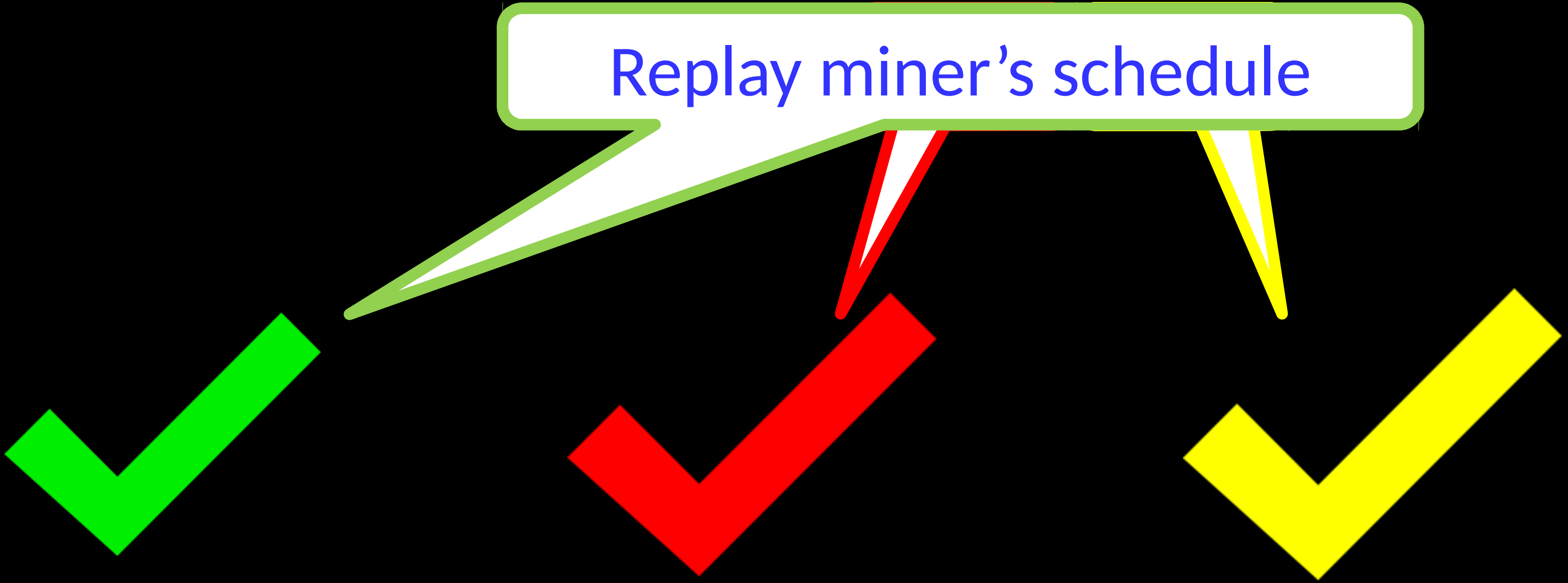
Because parallel executions non-deterministic

Might find a different serializable schedule



I have posted my serializable concurrent schedule for these contracts in the block!

Replay miner’s schedule



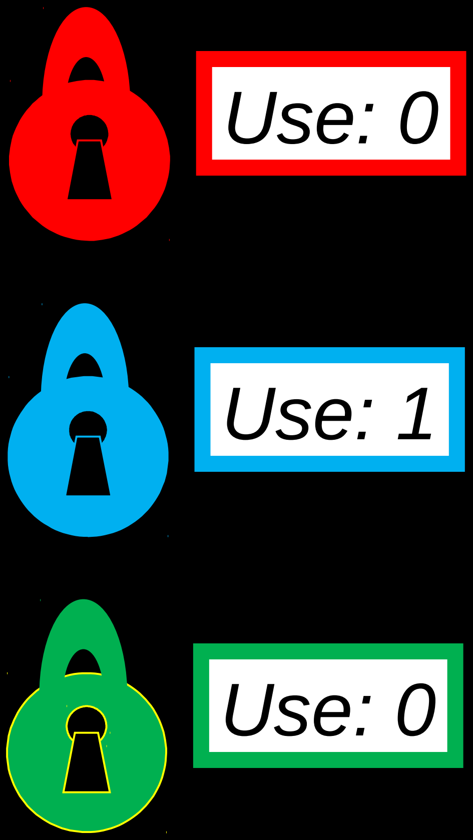
Replay miner’s schedule

Deterministic

Checkable

No locks or synchronization

Work-stealing for flexibility

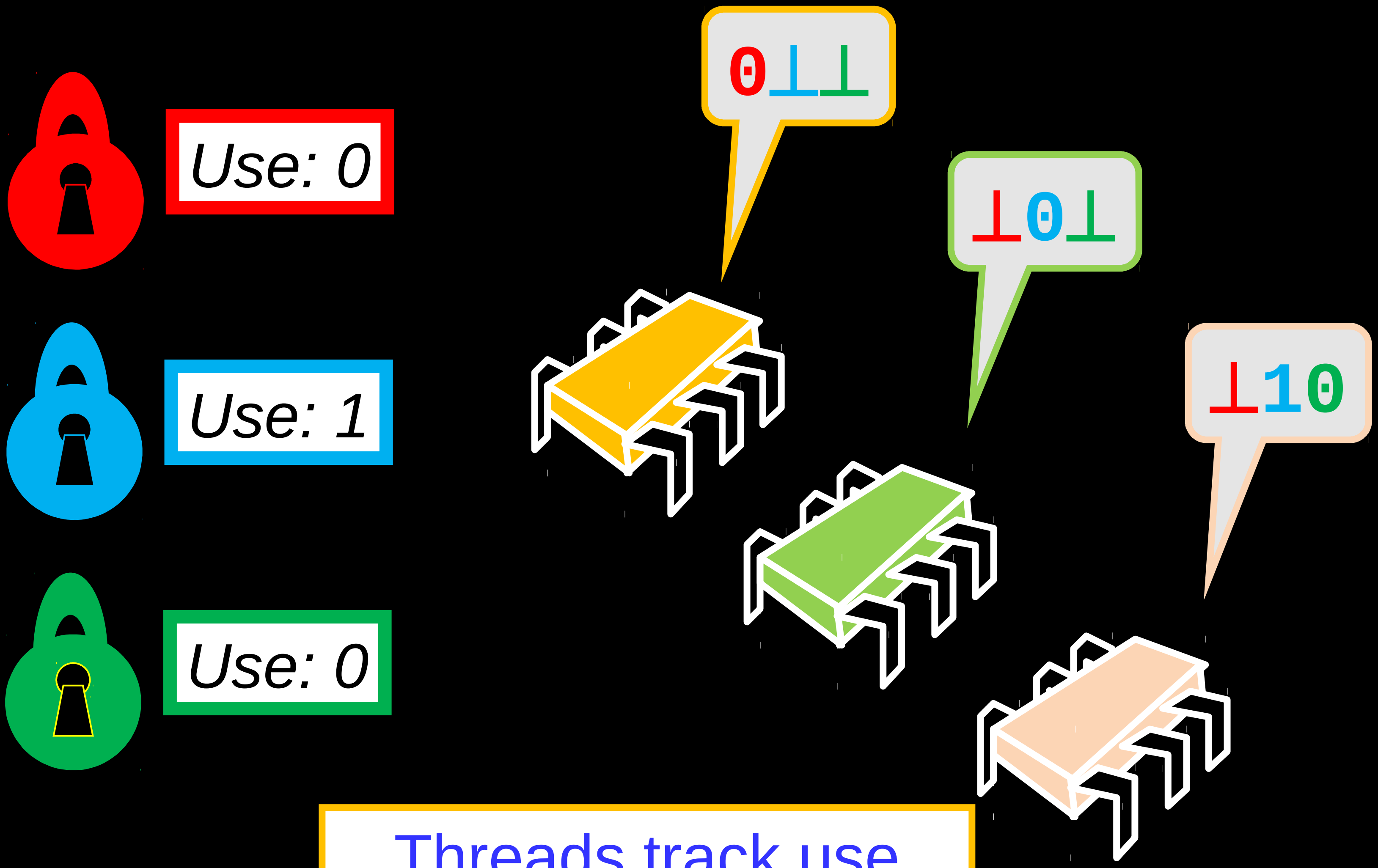


Use: 0

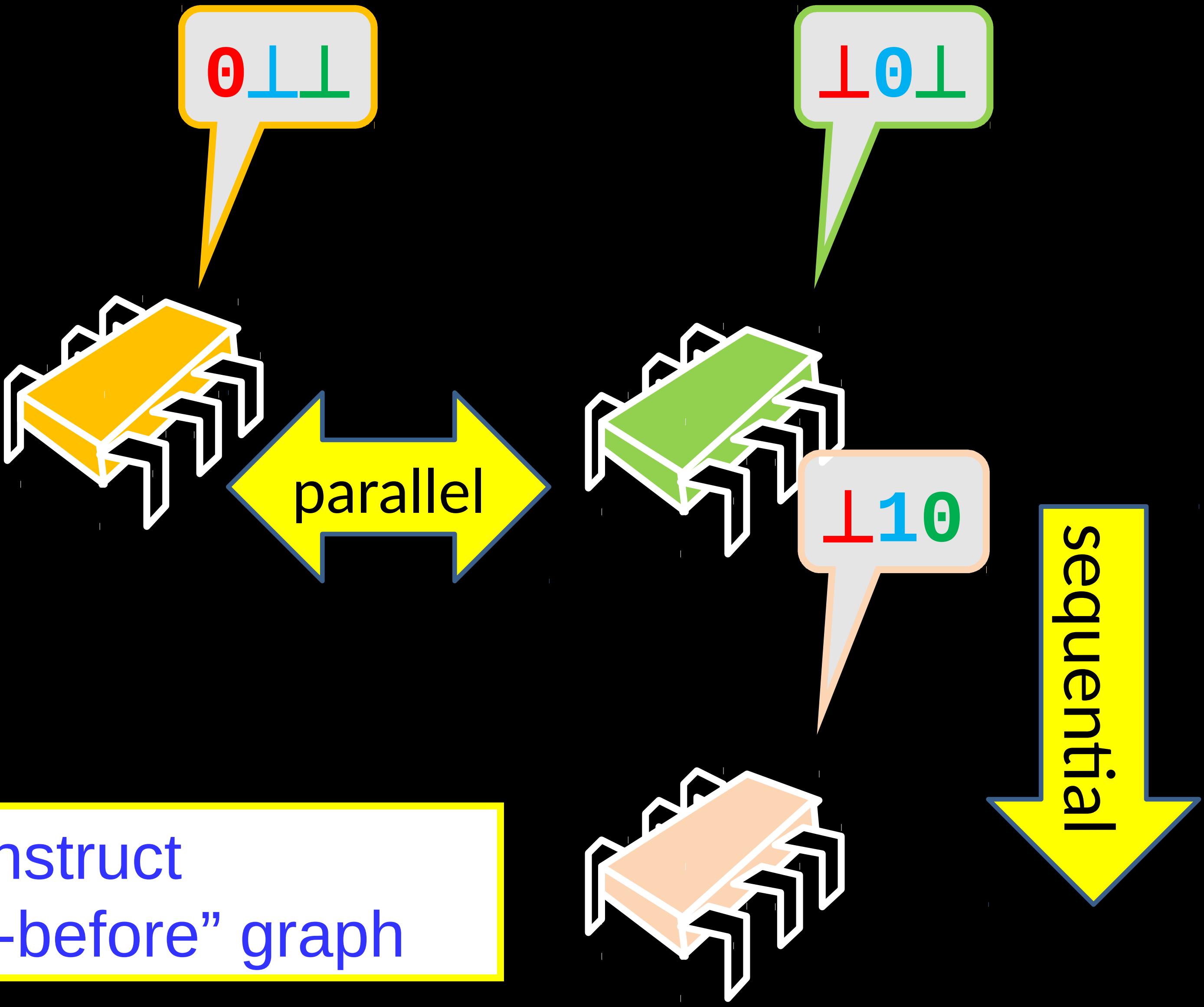
Use: 1

Use: 0

Locks track number of times acquired

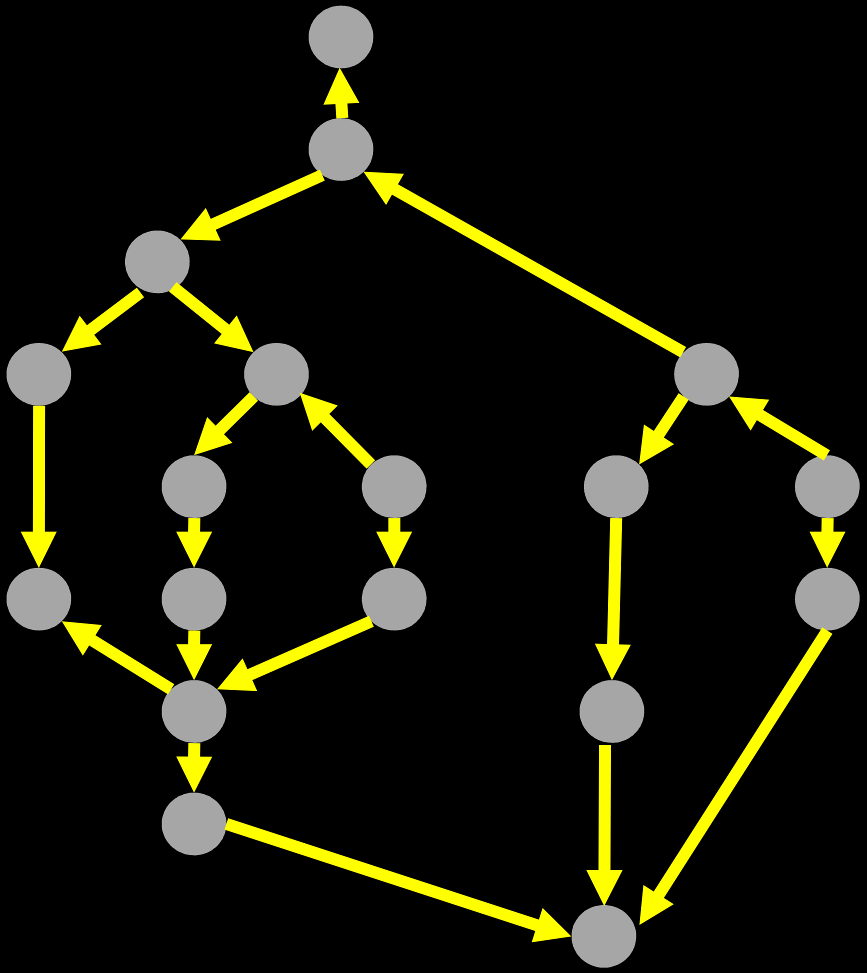


Threads track use counts seen



Can reconstruct
“happens-before” graph

Fork-Join Parallelism

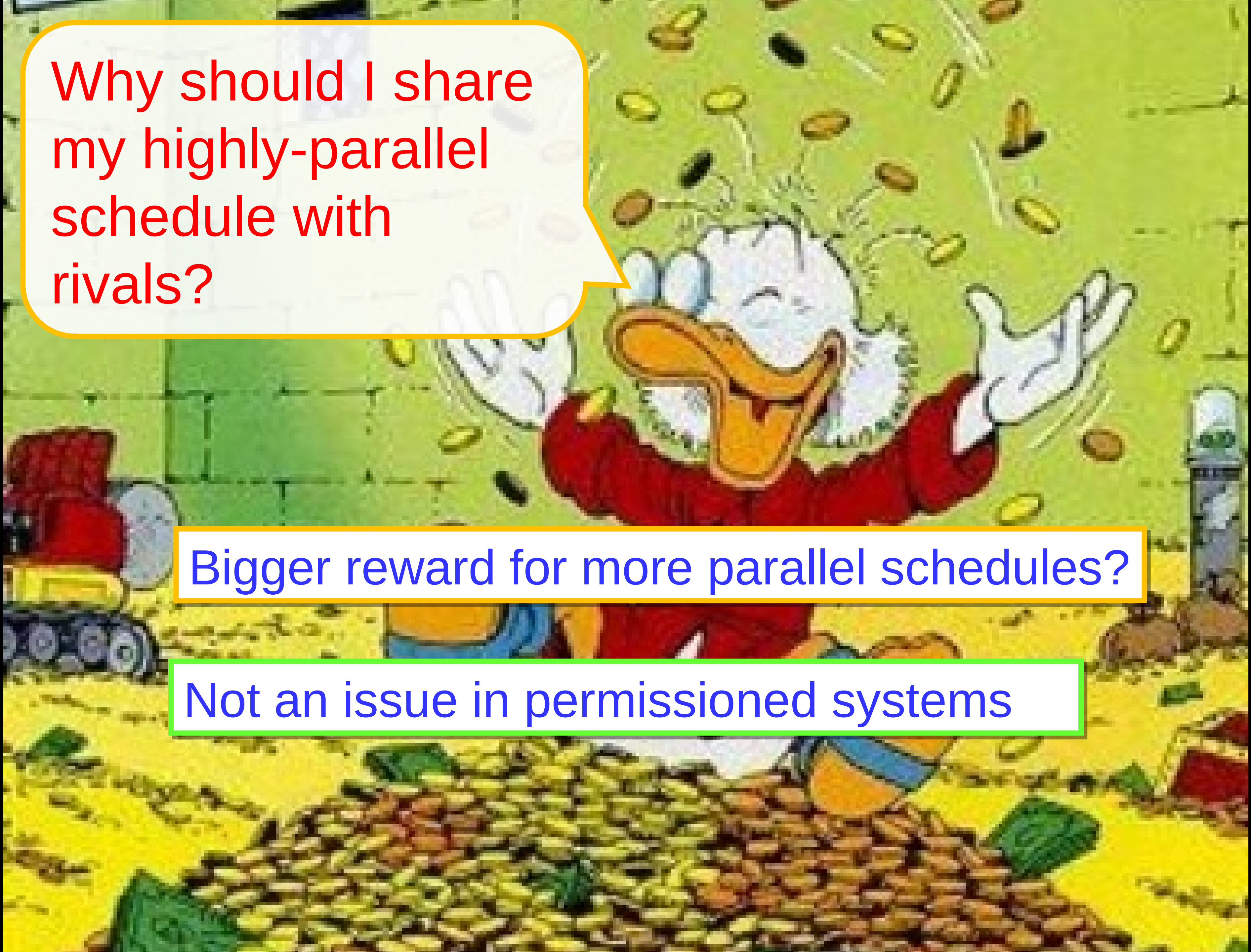


Similar to CILK model

Easy to schedule efficiently

Can check validity

No locks, undo, etc.



Why should I share my highly-parallel schedule with rivals?

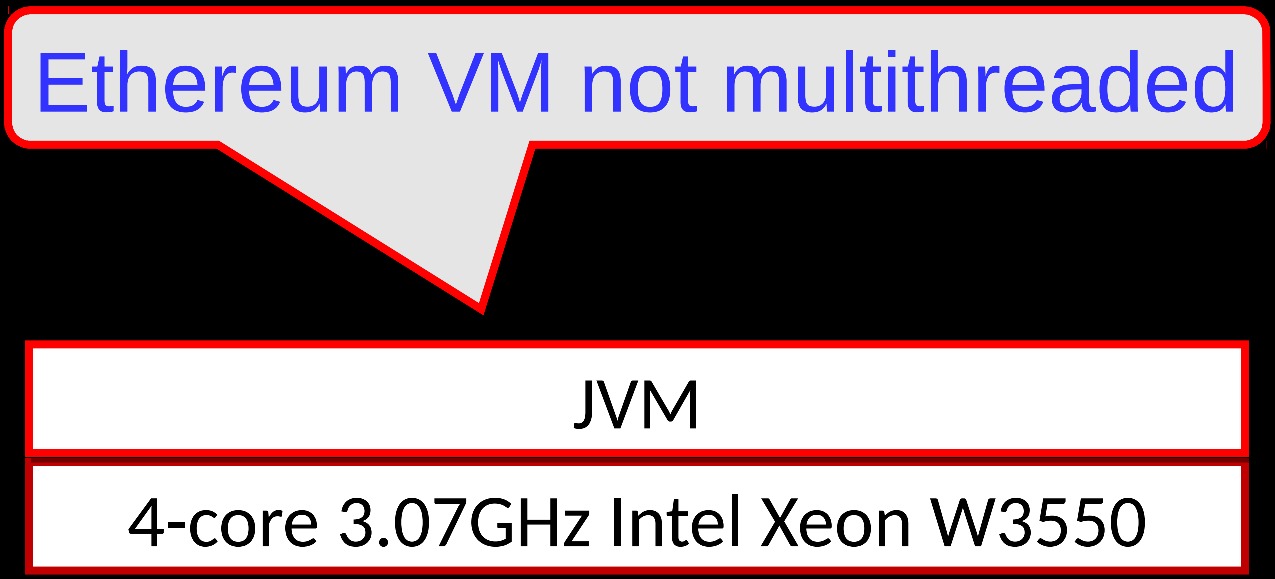
Bigger reward for more parallel schedules?

Not an issue in permissioned systems

BUNDESREPUBLIK DEUTSCHLAND
· J ·

Available hardware

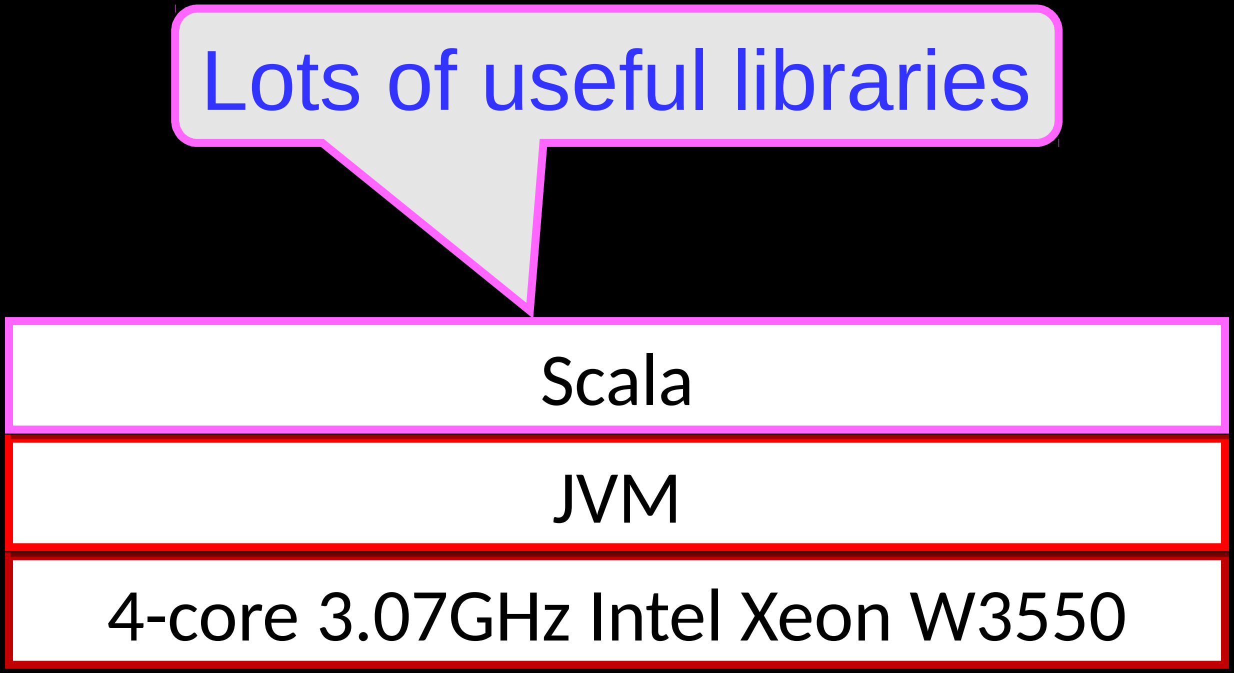
4-core 3.07GHz Intel Xeon W3550



Ethereum VM not multithreaded

JVM

4-core 3.07GHz Intel Xeon W3550

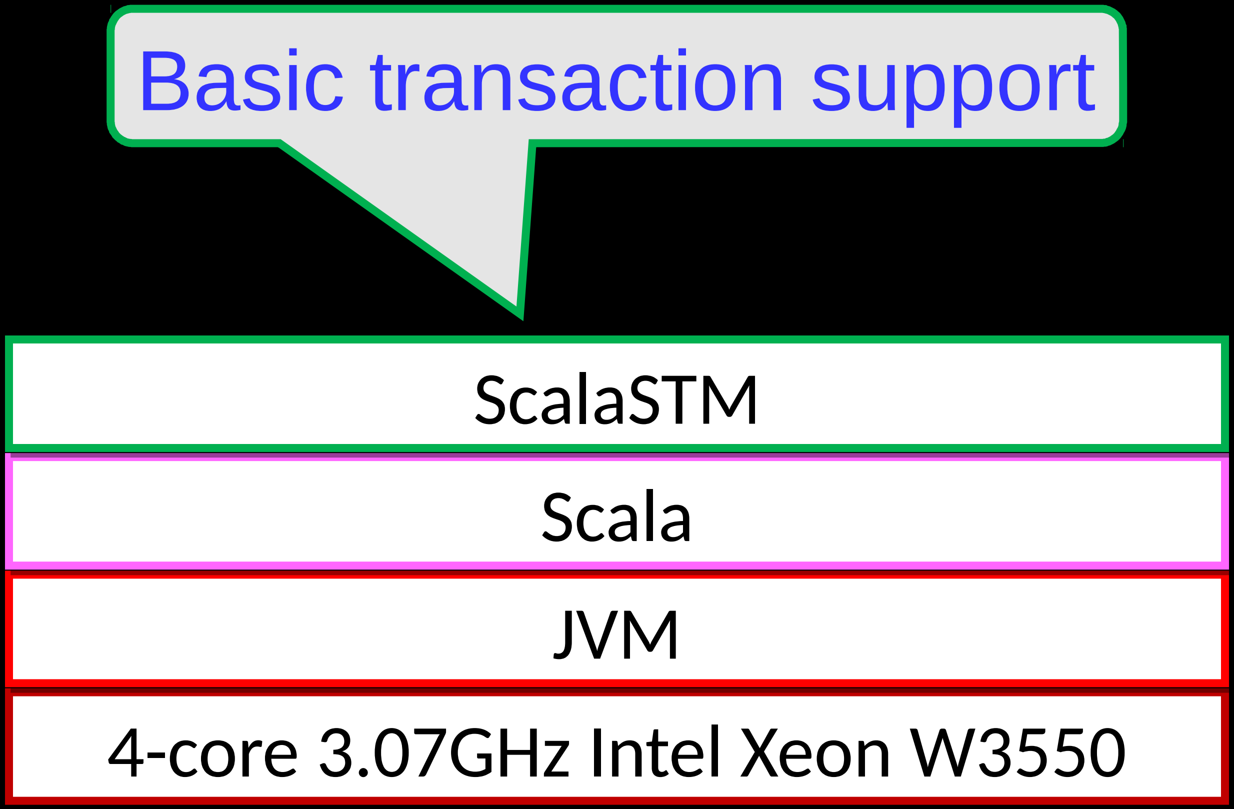


Lots of useful libraries

Scala

JVM

4-core 3.07GHz Intel Xeon W3550



Basic transaction support

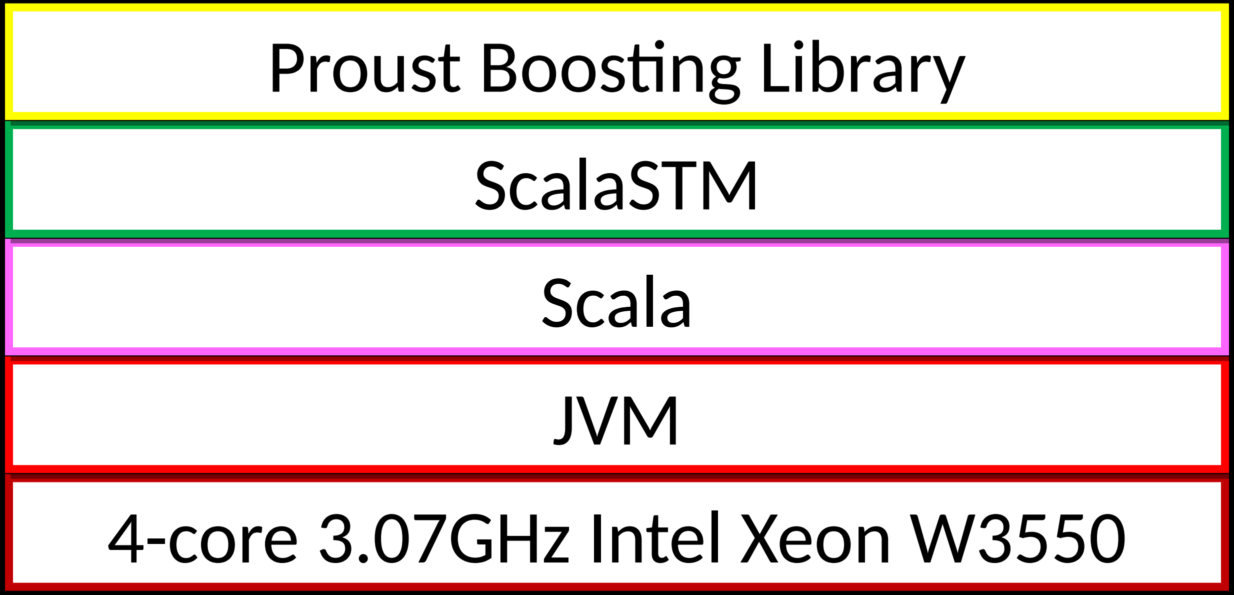
ScalaSTM

Scala

JVM

4-core 3.07GHz Intel Xeon W3550

Abstract locks, undo logs, etc….



Proust Boosting Library

ScalaSTM

Scala

JVM

4-core 3.07GHz Intel Xeon W3550

Benchmarks

JVM with JIT turned off

3 cores (1 more reserved for GC)

Single-benchmark blocks

Mixed-benchmark blocks

Tunable Conflict rate

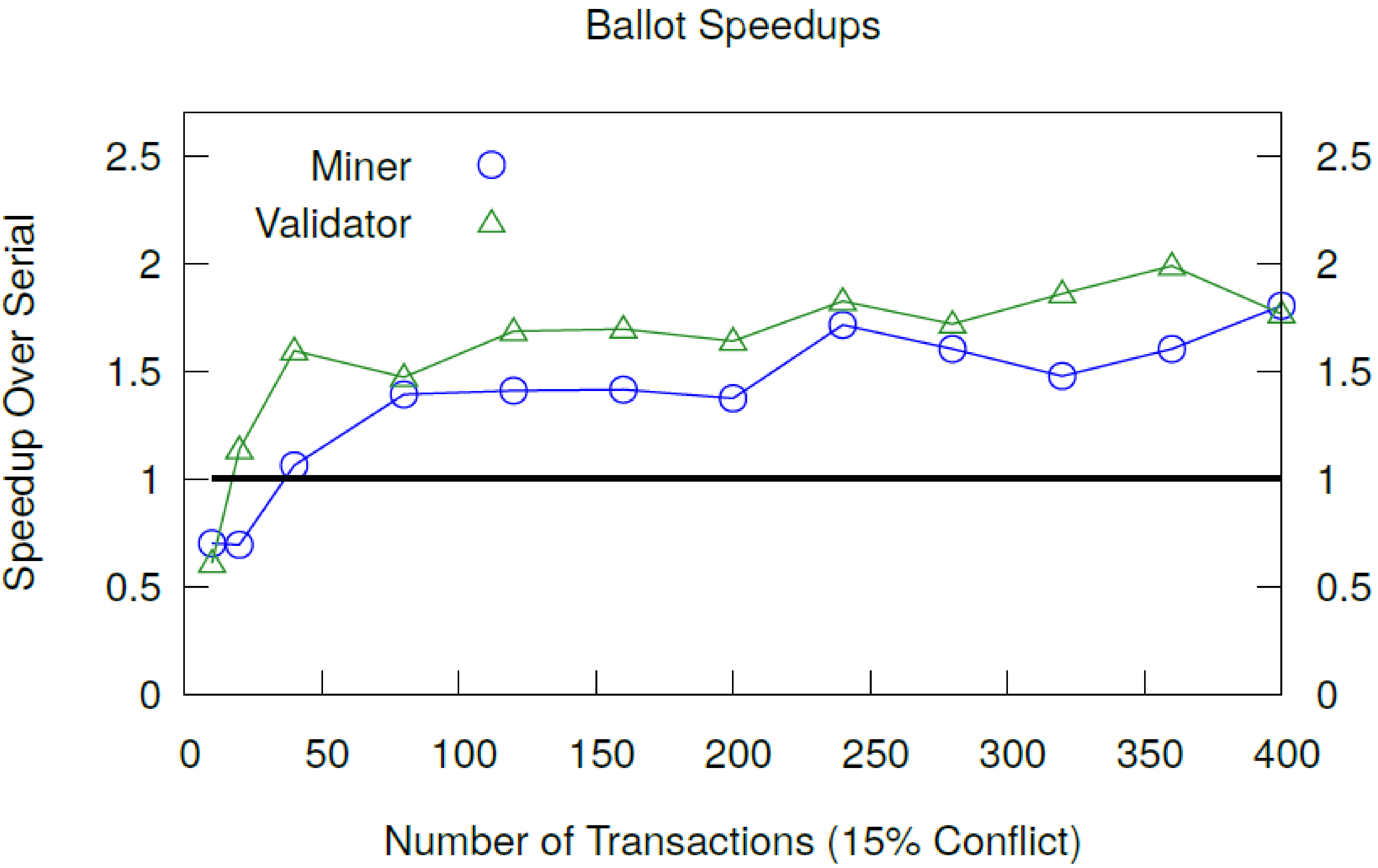
Ballot

From Solidity documentation

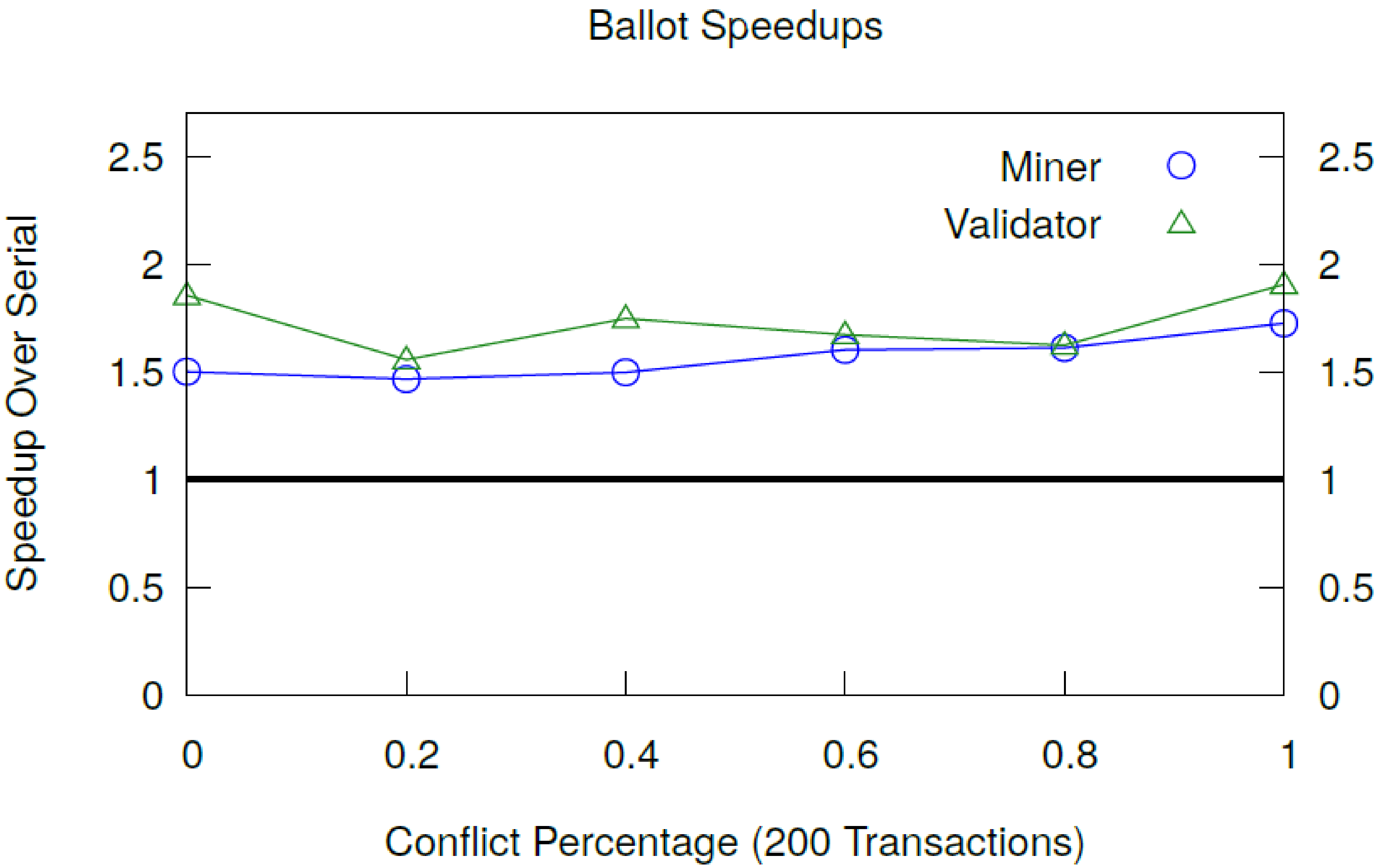
Voters register, vote

Benchmark: all voters registered, vote only

Tunable Conflict = double voting



Varying Transactions per Block



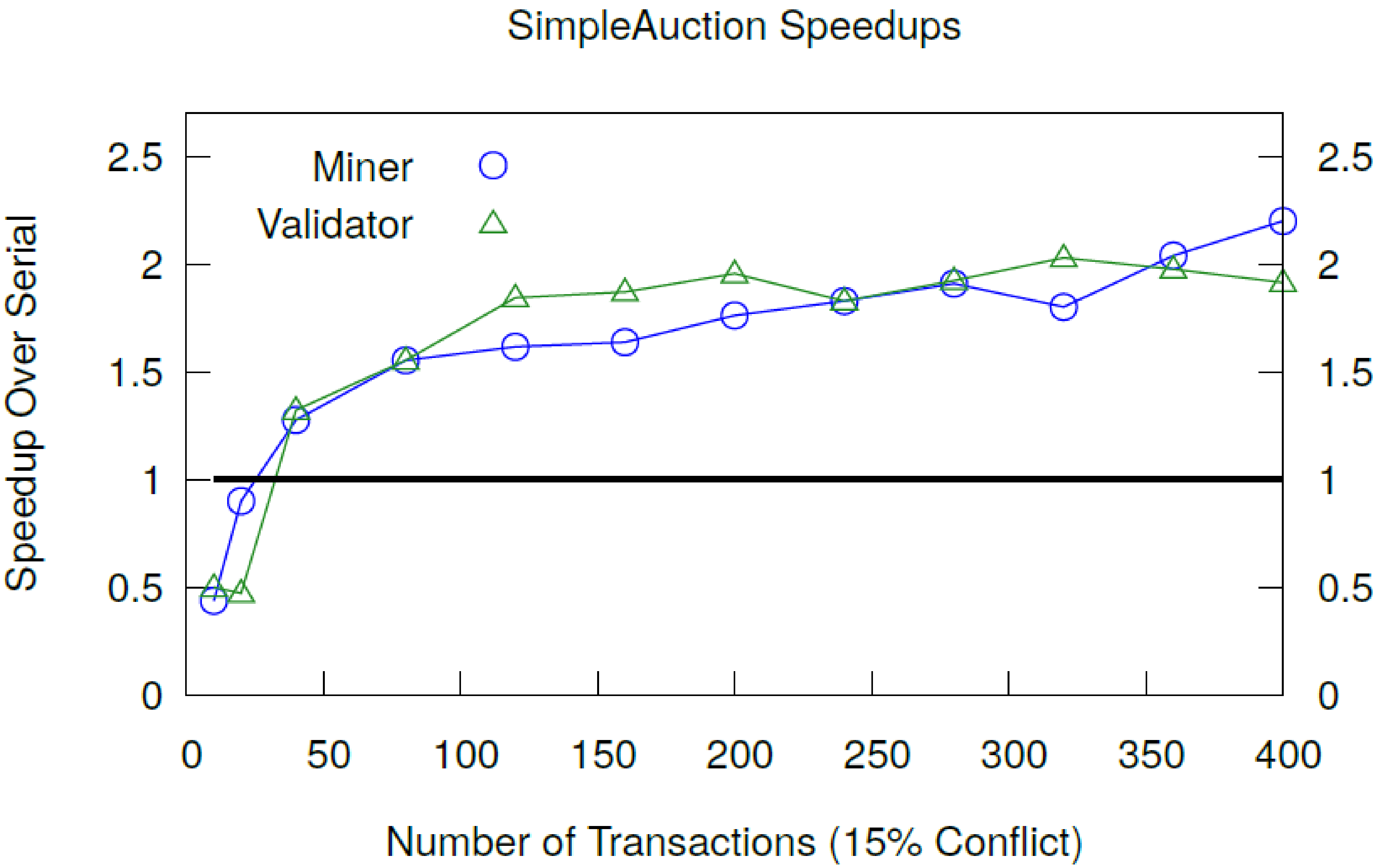
Varying Levels of Conflict

SimpleAuction

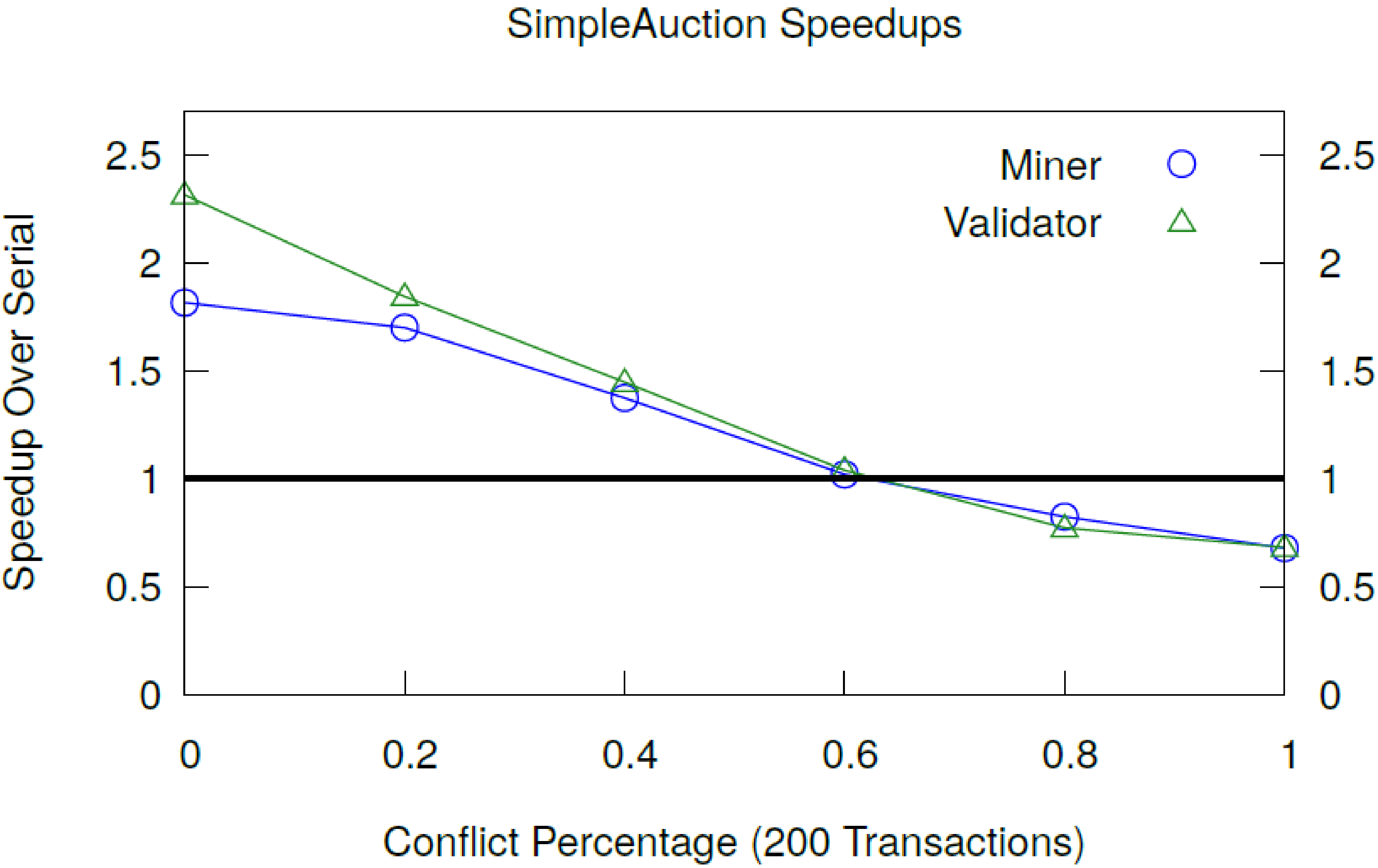
From Solidity documentation

Bidders bid, request refunds when over

Tunable Conflict = bidPlusOne() vs refund



Varying Transactions per Block



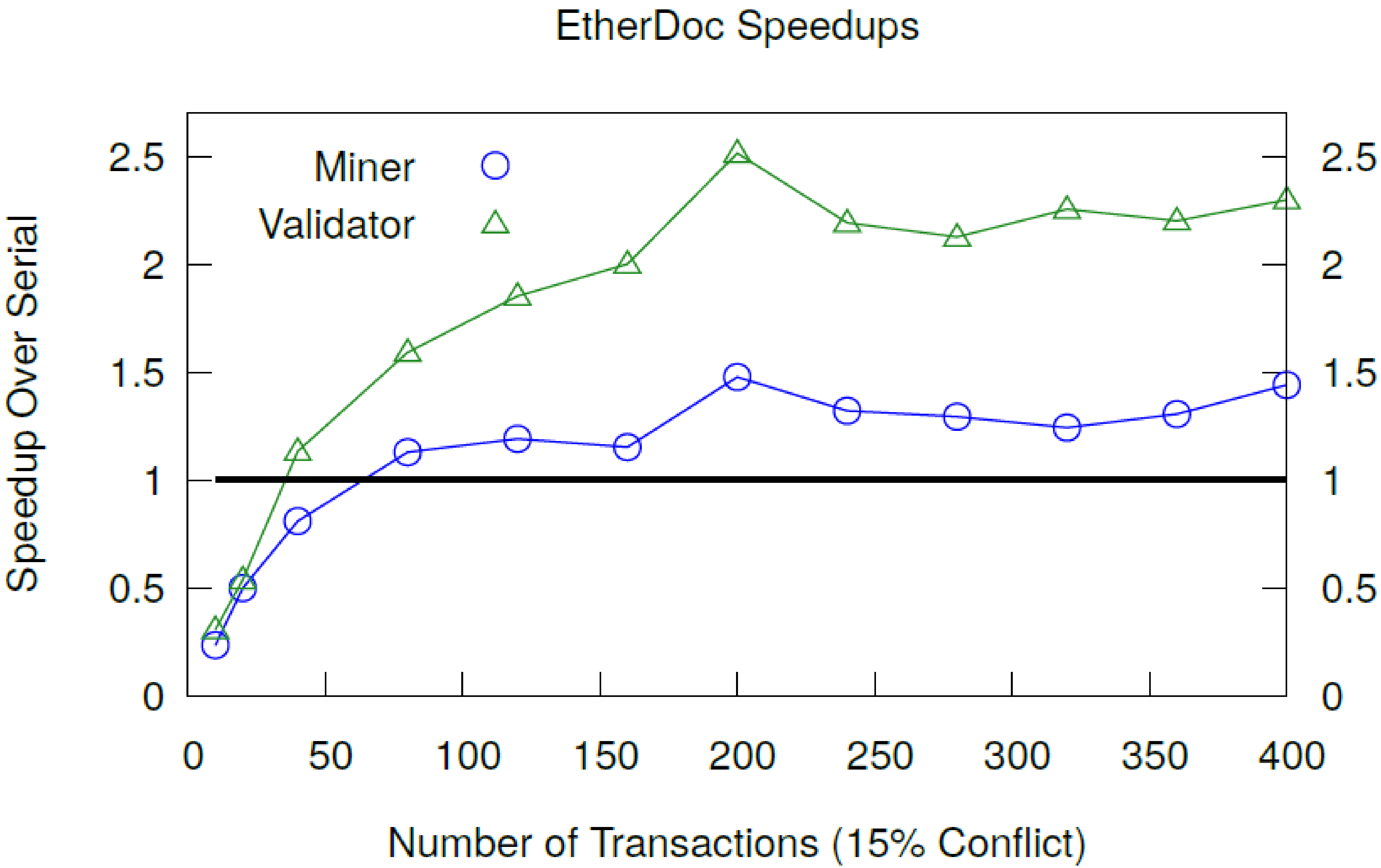
Varying Levels of Conflict

EtherDoc

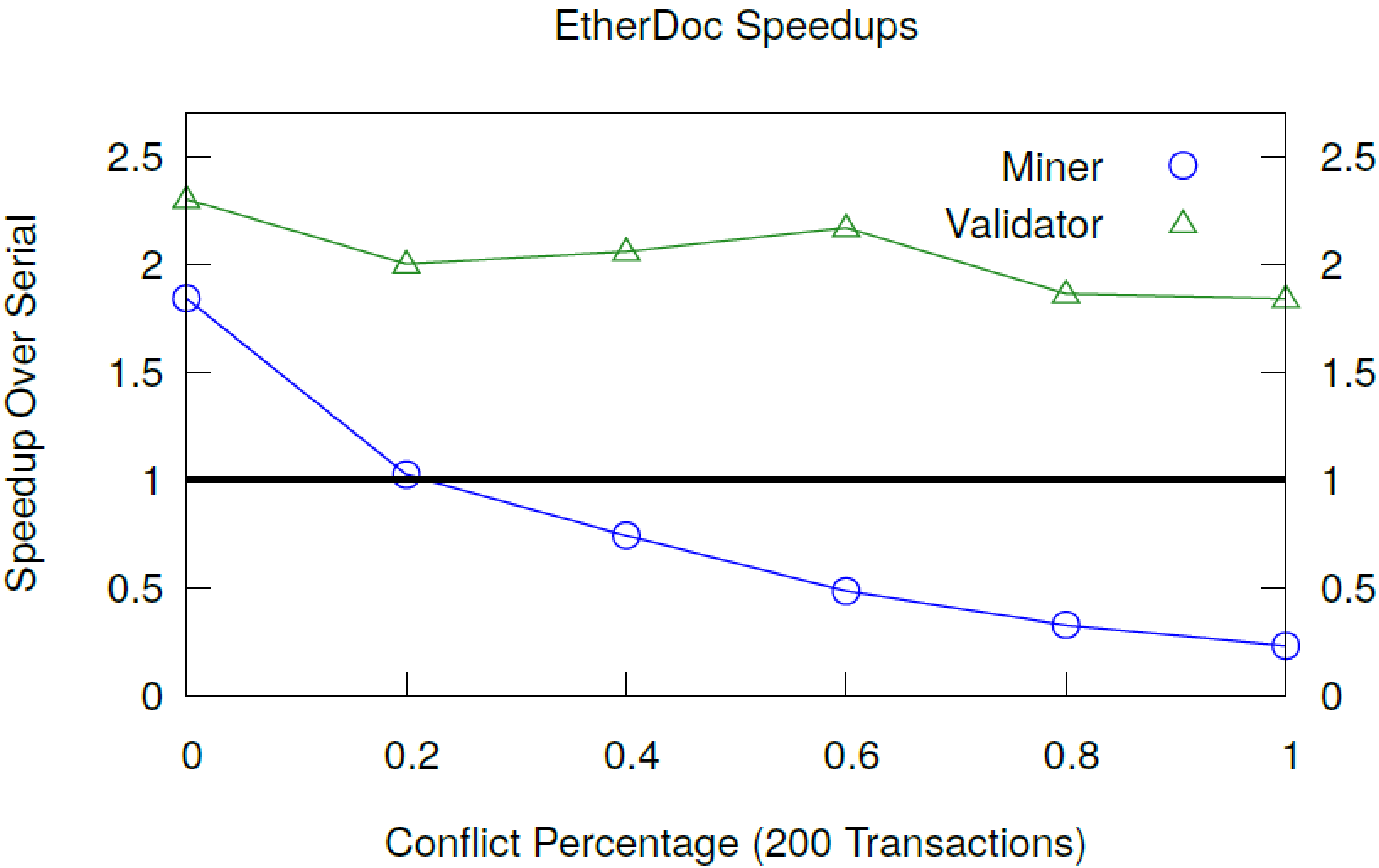
From website

Tracks Document Metadata (including owner)

Tunable Conflict = transfer vs query



Varying Transactions per Block

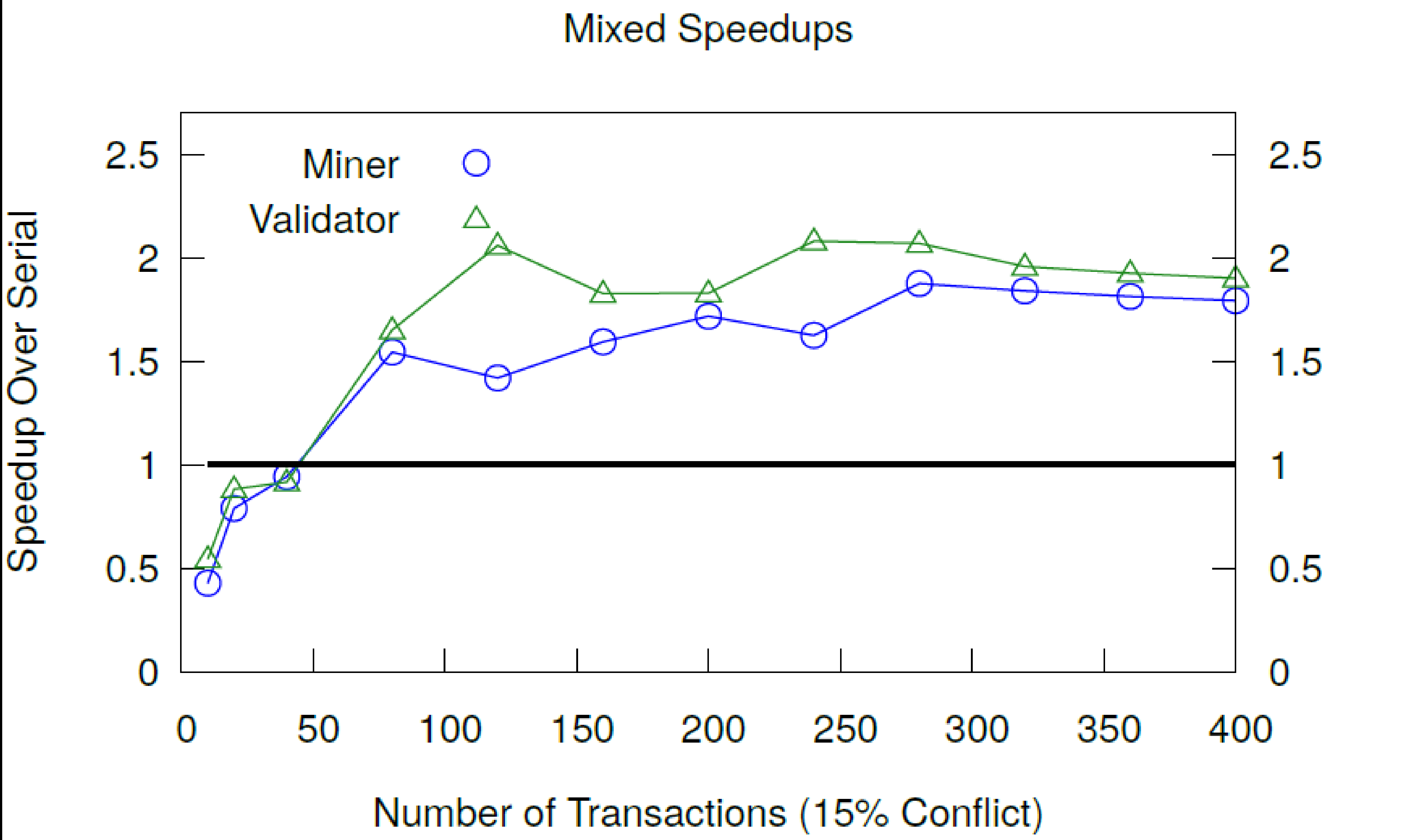


Varying Levels of Conflict

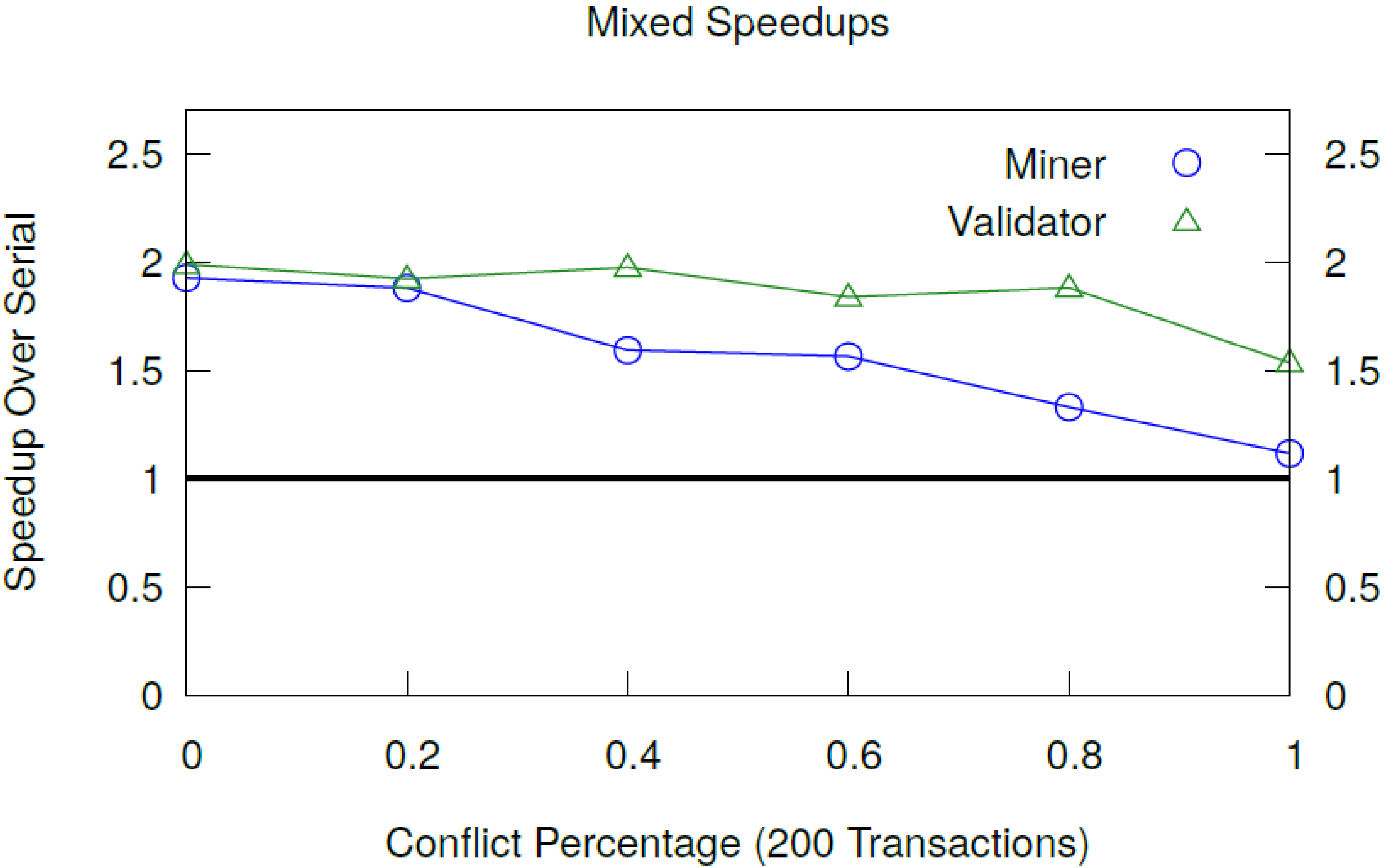
Mixed

All of the above

Equal proportions



Varying Transactions per Block



Varying Levels of Conflict

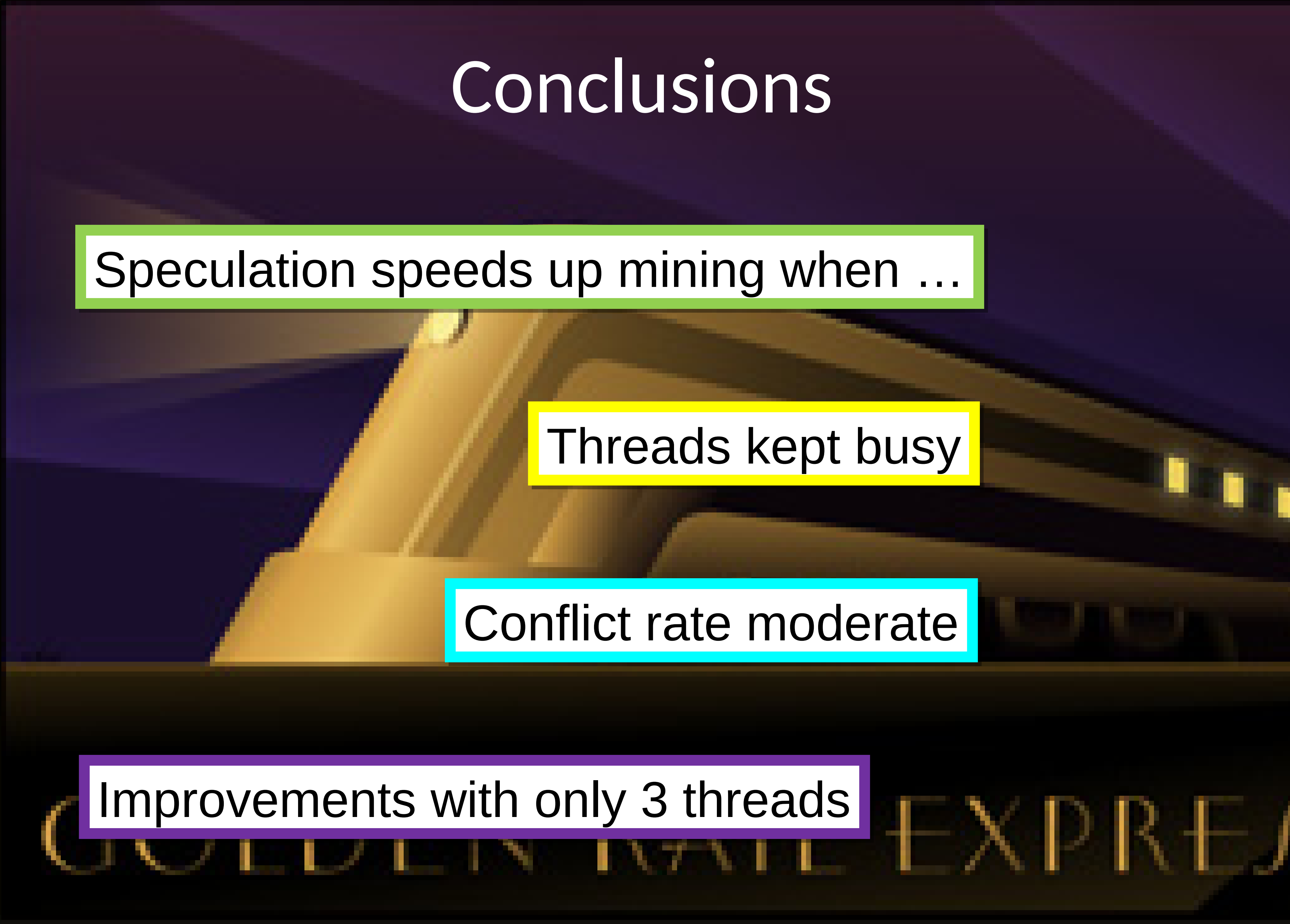
Conclusions

Speculation speeds up mining when …

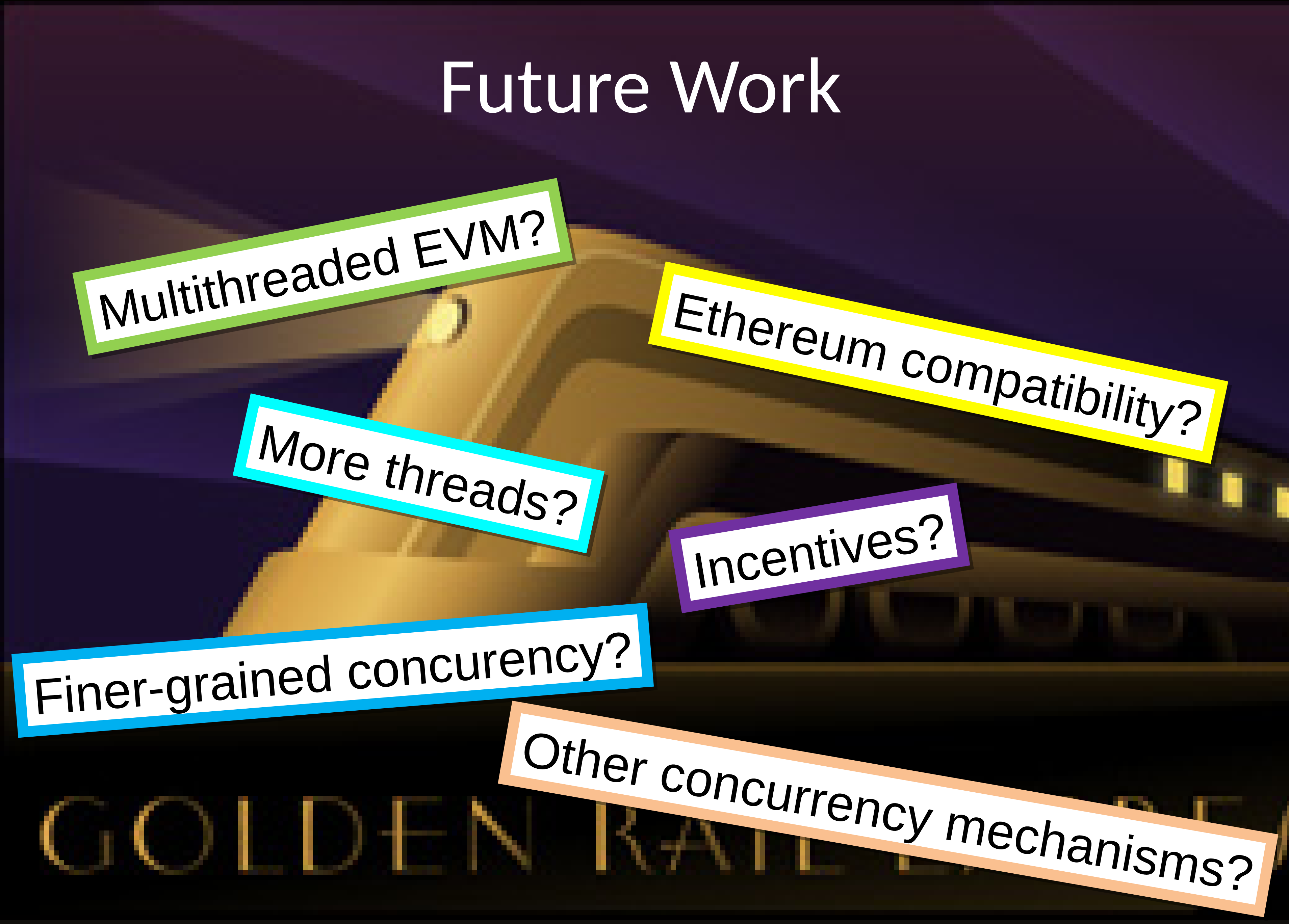
Threads kept busy

Conflict rate moderate

Improvements with only 3 threads



Future Work



Multithreaded EVM?

Ethereum compatibility?

More threads?

Incentives?

Finer-grained concurency?

Other concurrency mechanisms?