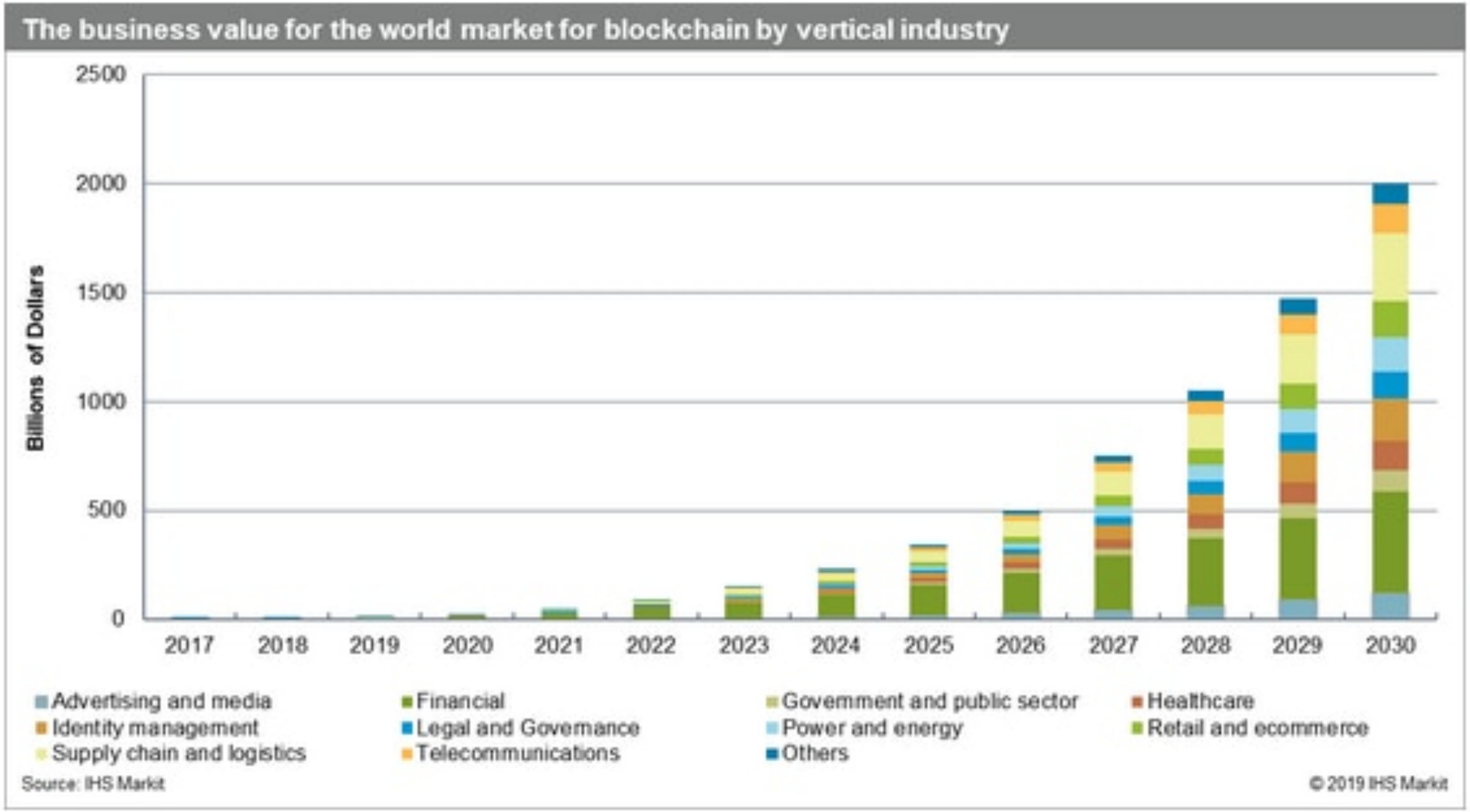
ORDERRR!

A Tale of Money, Intrigue, and Specifications

Lorenzo Alvisi
Cornell University
(joint work with Yunhao Zhang, Lidong Zhou, Qi Chen, Srinath Setty)

BLOCKCHAINS

BLOCKCHAINS




M·AGRIPPA·L·F·COS·TERTIVM·FECIT


570.

THE LEDGER

An append-only log for storing data

DECENTRALIZED

THE LEDGER

An append-only log for storing data

E R
D A N
T C Z L
E E I D

PERMISSIONLESS

No control over who can update the ledger

Unknown number of users who can update

Users can join and leave at any time

PERMISSIONED

Tight control over who can update the ledger

Known number of users who can update

Users need permission to join

PERMISSIONED

Tight control over who can update the ledger

Known number of users who can update

Users need permission to join

PERMISSIONED

Tight control over who can update the ledger

HYPERLEDGER

Known number of users who can update

Users need permission to join

PERMISSIONED

Tight control over who can update the ledger

Known number of users who can update

Users need permission to join

STATE MACHINE REPLICATION

Lamport, 1972

Coordinate replicas of a
deterministic service
to produce the abstraction of
a single, correct node

STATE MACHINE REPLICATION

Lamport, 1972

Ingredients: a service

1. Implement service as a deterministic state machine
2. Replicate
3. Build a total order of client requests, and execute them in that order
4. Vote on replica outputs

Safety: The ledger of correct replicas hold the same immutable sequence of commands

Liveness: Commands from correct clients eventually appear in the ledger of all correct replicas

+BFT: S&L hold if fewer than 1/3 of replicas are Byzantine

THERE'S THE RUB…

Ingredients: a service

1. Implement service as a deterministic state machine
2. Replicate
3. Build a total order of client requests, and execute them in that order
4. Vote on replica outputs

When it's about fault tolerance, order does not matter

When it's about financial transactions, order matters!

DUH!

DUH!

$ 0

FRONTRUNNING

Client

Issues an order
to buy a million shares
of ACME Co.

Broker

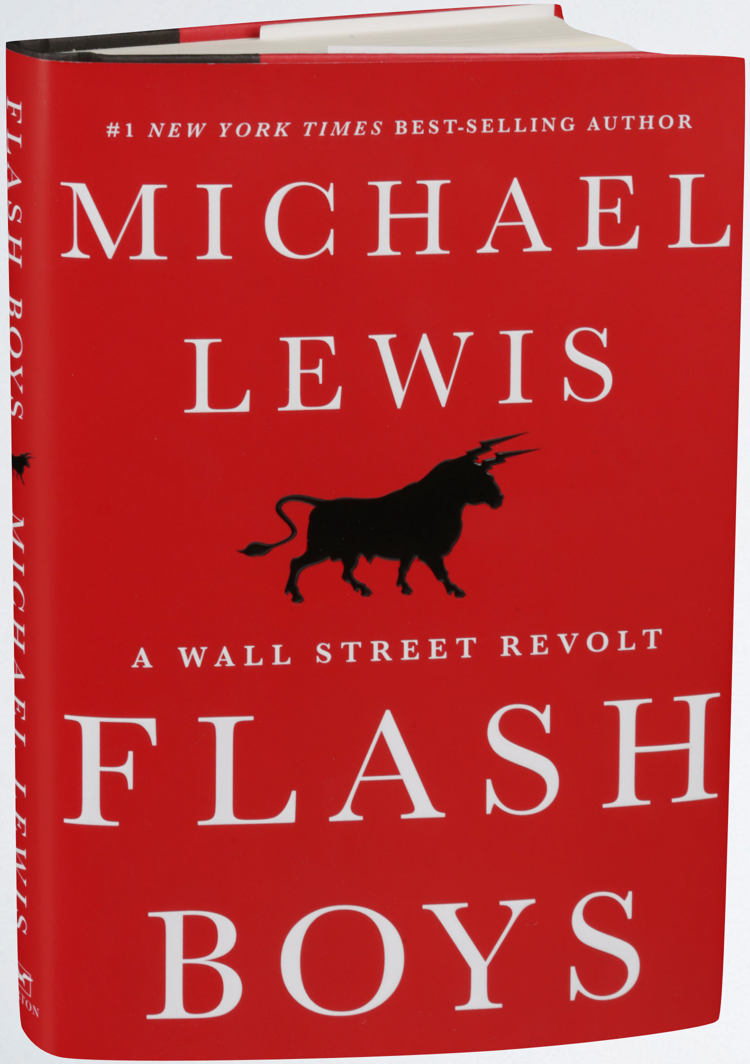
Places order for same
stock on his account

Then places
client's order

Exchange

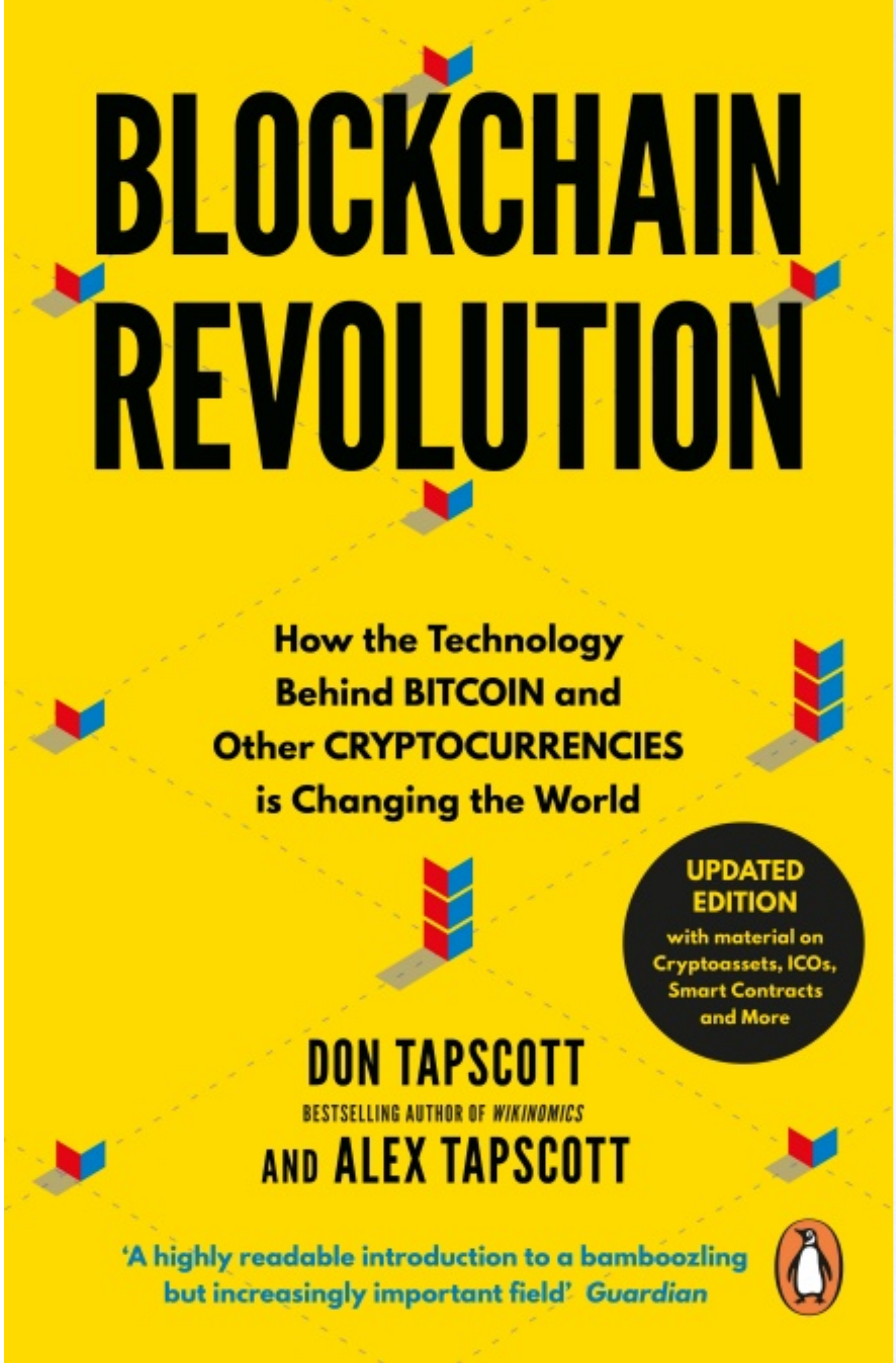
Broker sells for a huge profit,
possibly at client's expense

HIGH FREQUENCY TRADING



* Bots, algorithms, specialized hardware and fiber optic cables
* Front running through latency arbitrage
  * brokers (for a fee) let HFT firms see big upcoming orders
  * exchanges (for a fee) make it possible for HFT to frontrun these order

"The market is rigged"
Michel Lewis



"Blockchains can help build integrity into all our institutions and create a more secure and trustworthy world"

FAIR EXCHANGE

Alice

Bob

FAIR EXCHANGE

Alice

Bob

FAIR EXCHANGE

Alice

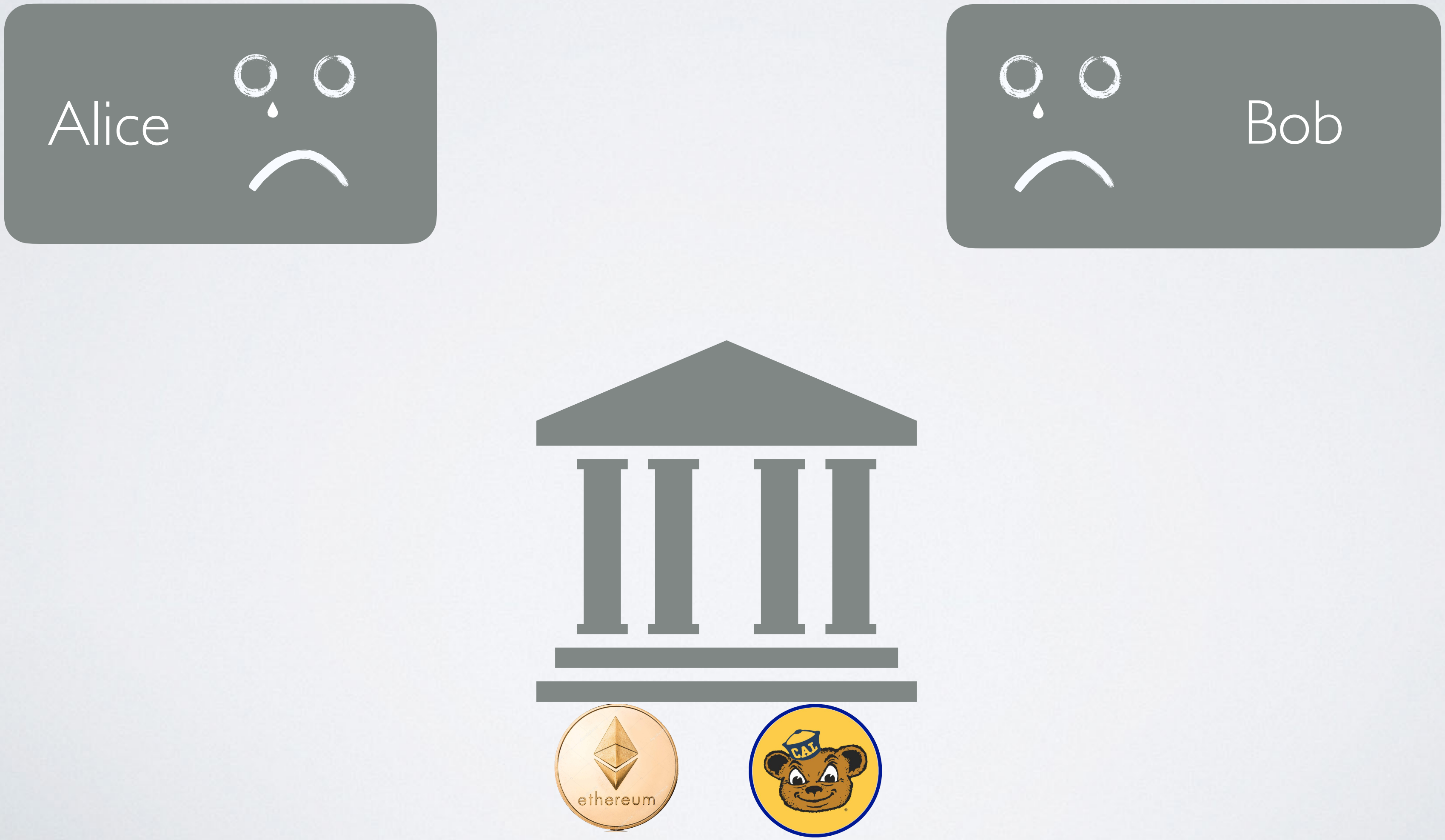
Bob

FAIR EXCHANGE

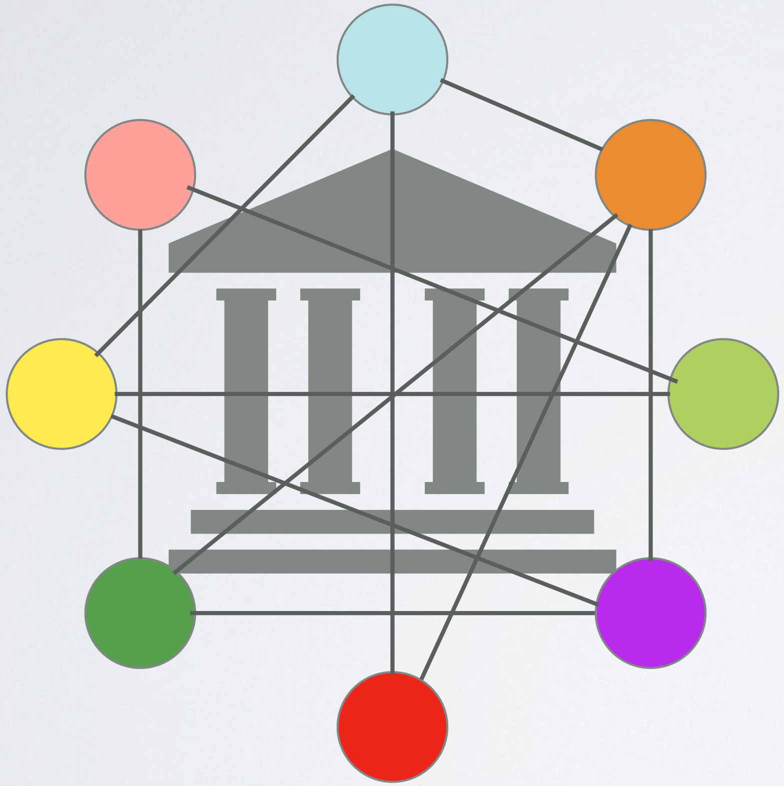
WHAT CAN POSSIBLY GO WRONG?

WHAT CAN POSSIBLY GO WRONG?

WHAT CAN POSSIBLY GO WRONG?



DECENTRALIZED EXCHANGES

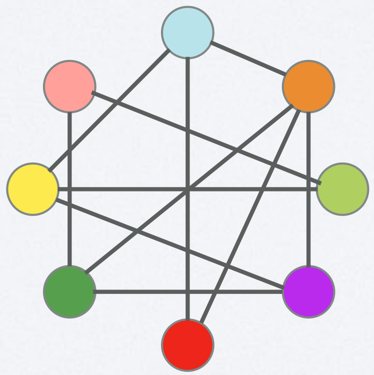


* ✳ Exchange operator holds an order book, but not the assets
* ✳ Assets held in custody in a smart contract
    * ‣ can't be stolen or lost by exchange operator
* ✳ Accessible to anyone
* ✳ Transparent

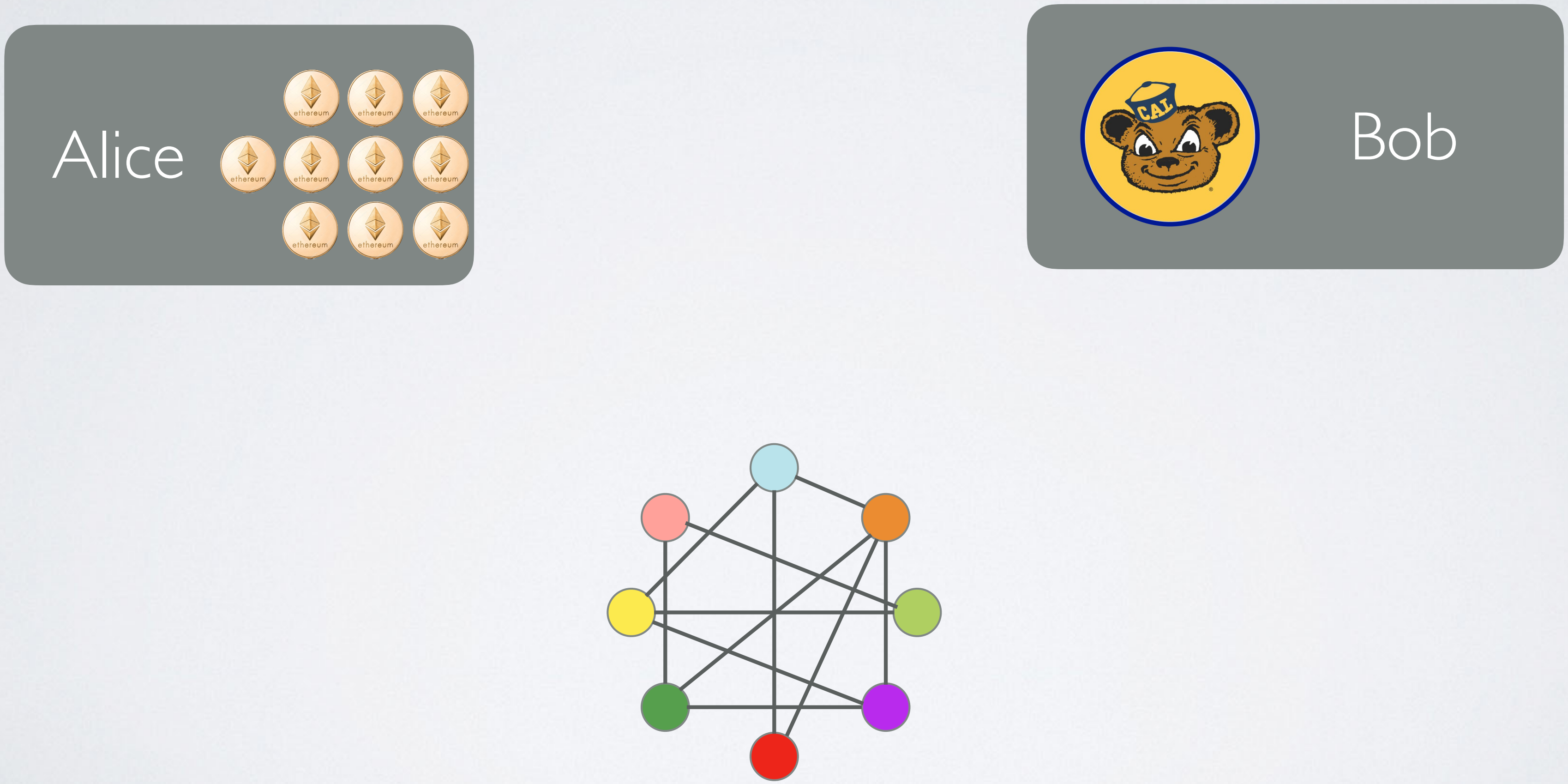
FAIR EXCHANGE

Alice

Bob



OOPS…



Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges

Philip Daian, *Cornell Tech*, phil@cs.cornell.edu
Steven Goldfeder, *Cornell Tech*, goldfeder@cornell.edu
Tyler Kell, *Cornell Tech*, sk3259@cornell.edu
Yunqi Li, *UIUC*, yunqil3@illinois.edu
Xueyuan Zhao, *CMU*, xyzhao@cmu.edu
Iddo Bentov, *Cornell Tech*, ib327@cornell.edu
Lorenz Breidenbach, *ETH Zürich*, lorenz.breidenbach@inf.ethz.ch
Ari Juels, *Cornell Tech*, juels@cornell.edu

arXiv:1904.05234v1 [cs.CR] 10 Apr 2019

***Abstract*—Blockchains, and specifically smart contracts, have promised to create fair and transparent trading ecosystems.**

Unfortunately, we show that this promise has not been met. We document and quantify the widespread and rising deployment of *arbitrage bots* in blockchain systems, specifically in *decentralized exchanges* (or "DEXes"). Like high-frequency traders on Wall Street, these bots exploit inefficiencies in DEXes, paying high transaction fees and optimizing network latency to frontrun, i.e., anticipate and exploit, ordinary users' DEX trades.

We study the breadth of DEX arbitrage bots in a subset of transactions that yield quantifiable revenue to these bots. We also study bots' profit-making strategies, with a focus on blockchain-specific elements. We observe bots engage in what we call *priority gas auctions* (PGAs), competitively bidding up transaction fees in order to obtain priority ordering, i.e., early block position and execution, for their transactions. PGAs present an interesting and complex new continuous-time, partial-information, game-theoretic model that we formalize and study. We release an interactive web portal, frontrun.me, to provide the community with real-time data on PGAs.

We additionally show that high fees paid for priority transaction ordering poses a systemic risk to *consensus-layer* security. We explain that such fees are just one form of a general phenomenon in DEXes and beyond—what we call *miner extractable value* (MEV)—that poses concrete, measurable, consensus-layer security risks. We show empirically that MEV poses a realistic threat to Ethereum today.

Our work highlights the large, complex risks created by transaction-ordering dependencies in smart contracts and the ways in which traditional forms of financial-market exploitation are adapting to and penetrating blockchain economies.

I. Introduction

Cryptocurrency exchanges today handle more than $10 billion in trade volume per day. The vast majority of this volume occurs in *centralized* exchanges, which hold custody of customer assets and settle trades. At best loosely regulated, centralized exchanges have experienced scandals ranging from high-profile thefts [38] to malfeasance such as price manipulation [22]. One popular alternative is what is called a *decentralized exchange* (or "DEXes").[1] In a DEX, a smart contract (a program executing on a blockchain) or other form of peer-to-peer network executes exchange functionality.

At first glance, decentralized exchanges seem ideally designed. They appear to provide effective price discovery and fair trading, while doing away with the drawbacks of centralized exchanges. Trades are atomically executed by a smart contract and visible on the Ethereum blockchain, providing the appearance of transparency. Funds cannot be stolen by the exchange operator, because their custody and exchange logic is processed and guaranteed by the smart contract.

Despite their clear benefits, however, many DEXes come with a serious and fundamental weakness: on-chain, smart-contract-mediated trades are slow.[2] Traders thus may attempt to take orders that have already been taken or canceled but appear active due to their views of messages sent on the network. Worse still, adversaries can *frontrun* orders, observing them and placing their own orders with higher fees to ensure they are mined first.

Past work has acknowledged "transaction ordering dependence" as an anti-pattern and vector for potential frontrunning [30, 34]. Unfortunately, these analyses have previously proved overly broad: virtually every smart contract can be said to have *some* potential dependence on transaction order, the majority of which is benign. As a result, effective practical mitigations for these issues have failed to materialize, and few deployed smart contracts feature ordering protections. Other work has focused on systematizing knowledge around smart contract frontrunning [18], including citing early public versions of this work, but has not measured the size of this economy or formalized its connection to protocol attacks.

[1] "Decentralized" exchange is something of a misnomer, as many such systems have centralized components; most systems we call "decentralized" exchanges could more accurately be classified as non-custodial: users trade without surrendering control of their funds to a third party in the process.

[2] The average Ethereum block time is roughly 15s at the date of writing [19].

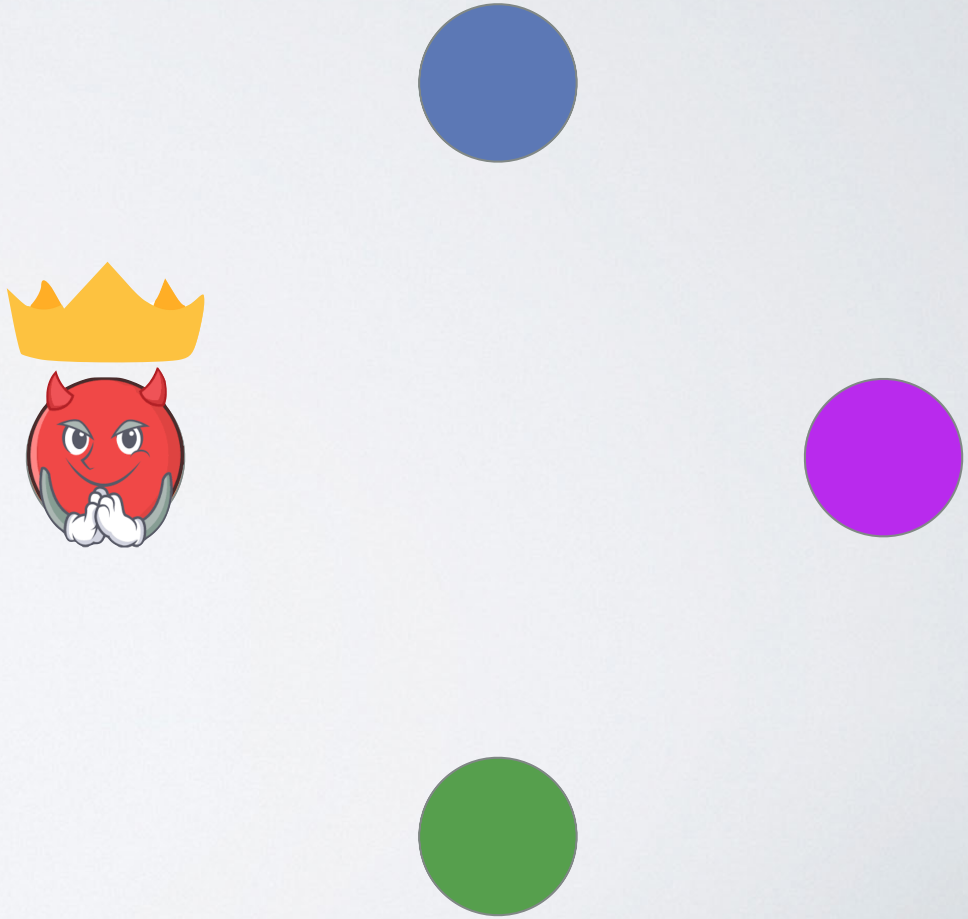
THE RISE OF ARBITRAGE BOTS

Bots routinely "bribe" miners on the Ethereum blockchain to frontrun other users

and viciously compete with one another for the privilege to do so!

WHAT ABOUT PERMISSIONED BLOCKCHAINS?

- Most protocols are leader-based
- Leader has full control over the ledger's order
- Bad if the leader has an agenda…



ROTATING LEADERS

* Yet…
* Each leader still has full control over the order of commands in its batch
* Mistakes mechanism for policy

THE CRUX

The spec has no way to express, never mind enforce, "good" orders

ORDERING PREFERENCES

- Pair each command c with an ordering indicator o
  - sequence number; timestamp; dependency graph…
- For any pair of proposals $\langle o_1, c_1 \rangle$ and $\langle o_2, c_2 \rangle$

$$o_1 \prec_o o_2$$

  indicates a preference to order c_1 before c_2

PROFILES AND TRACES

- Profile: $\mathscr{P}^i$ is the set of proposals of node i
  - $\mathscr{P}^C$: set of profiles of correct nodes $c \in C$
- Trace: the result of a single run of a consensus protocol augmented by ordering preferences
  - same $\mathscr{P}^C$ may yield different traces, different ledgers
    - influenced by Byzantine nodes and network

BYZANTINE OLIGARCHY

* $\forall$ profiles $\mathscr{P}^C$, $\forall c_1, c_2$ in $\mathscr{P}^C$, there are two traces

T_1

Correct ledger

1		13	
2		14	
3		15	
4	c_1	16	
5		17	
6		18	c_2
7		19	
8		20	
9		21	
10		22	
11		23	
12		24	

and

T_2

Correct ledger

1		13	
2		14	
3		15	
4		16	
5		17	
6		18	
7		19	
8	c_2	20	
9		21	
10		22	
11	c_1	23	
12		24	

NO CAPRICIOUSNESS

* Commands by correct nodes can't be ignored
* Ordering preferences matter
    * $\exists\ \mathscr{P}_\alpha$ and $\mathscr{P}_\beta$ s.t. for all c_1, c_2 in both $\mathscr{P}_\alpha$ and $\mathscr{P}_\beta$

$T_{\mathscr{P}_\alpha}$

Correct ledger

1		13	
2	c_1	14	
3		15	
4		16	c_2
5		17	
6		18	
7		19	
8		20	

$T_{\mathscr{P}_\beta}$

Correct ledger

1		13	
2	c_2	14	
3		15	
4		16	
5	c_1	17	
6		18	
7		19	
8		20	

BYZANTINE POLITICS

No Capriciousness
implies
Byzantine Democracy

BYZANTINE OLIGARCHY

- $\forall$ profiles $\mathscr{P}^C$, $\forall c_1, c_2$, there are two traces

T_1

Correct ledger

1		13	
2		14	
3		15	
4	c_1	16	
5		17	
6		18	c_2
7		19	
8		20	
9		21	
10		22	
11		23	
12		24	

and

T_2

Correct ledger

1		13	
2		14	
3		15	
4		16	
5		17	
6		18	
7		19	
8	c_2	20	
9		21	
10		22	
11	c_1	23	
12		24	

Are there profiles insensitive to Byzantine influence?

★★★

Can we design protocols that enforce ordering guarantees that specify such profiles?

★★★

How would these guarantees look like?

ORDERING UNANIMITY

If all correct nodes
order c_1 before c_2

then

$c_1 < c_2$ in the ledger
of all correct nodes

ORDERING UNANIMITY?

Condorcet Cycle

$$\mathscr{P}^1 = \{\langle 1, c_1\rangle, \langle 2, c_2\rangle, \langle 3, c_3\rangle, \langle 4, c_4\rangle\}$$
$$\mathscr{P}^2 = \{\langle 1, c_2\rangle, \langle 2, c_3\rangle, \langle 3, c_4\rangle, \langle 4, c_1\rangle\}$$
$$\mathscr{P}^3 = \{\langle 1, c_3\rangle, \langle 2, c_4\rangle, \langle 3, c_1\rangle, \langle 4, c_2\rangle\}$$
$$\mathscr{P}^4 = \{\langle 1, c_4\rangle, \langle 2, c_1\rangle, \langle 3, c_2\rangle, \langle 4, c_3\rangle\}$$

ORDERING UNANIMITY?

Condorcet Cycle

$$\mathscr{P}^1 = \{\langle 1, c_1\rangle, \langle 2, c_2\rangle, \langle 3, c_3\rangle, \langle 4, c_4\rangle\}$$
$$\mathscr{P}^2 = \{\langle 1, c_2\rangle, \langle 2, c_3\rangle, \langle 3, c_4\rangle, \langle 4, c_1\rangle\}$$
$$\mathscr{P}^3 = \{\langle 1, c_3\rangle, \langle 2, c_4\rangle, \langle 3, c_1\rangle, \langle 4, c_2\rangle\}$$
$$\mathscr{P}^4 = \{\langle 1, c_4\rangle, \langle 2, c_1\rangle, \langle 3, c_2\rangle, \langle 4, c_3\rangle\}$$

ORDERING UNANIMITY?

Condorcet Cycle

$$\mathscr{P}^1 = \{\langle 1, c_1\rangle, \langle 2, c_2\rangle, \langle 3, c_3\rangle, \langle 4, c_4\rangle\}$$
$$\mathscr{P}^2 = \{\langle 1, c_2\rangle, \langle 2, c_3\rangle, \langle 3, c_4\rangle, \langle 4, c_1\rangle\}$$
$$\mathscr{P}^3 = \{\langle 1, c_3\rangle, \langle 2, c_4\rangle, \langle 3, c_1\rangle, \langle 4, c_2\rangle\}$$
$$\mathscr{P}^4 = \{\langle 1, c_4\rangle, \langle 2, c_1\rangle, \langle 3, c_2\rangle, \langle 4, c_3\rangle\}$$

ORDERING UNANIMITY?

Condorcet Cycle

$$\mathscr{P}^1 = \{\langle 1, c_1\rangle, \langle 2, c_2\rangle, \langle 3, c_3\rangle, \langle 4, c_4\rangle\}$$
$$\mathscr{P}^2 = \{\langle 1, c_2\rangle, \langle 2, c_3\rangle, \langle 3, c_4\rangle, \langle 4, c_1\rangle\}$$
$$\mathscr{P}^3 = \{\langle 1, c_3\rangle, \langle 2, c_4\rangle, \langle 3, c_1\rangle, \langle 4, c_2\rangle\}$$
$$\mathscr{P}^4 = \{\langle 1, c_4\rangle, \langle 2, c_1\rangle, \langle 3, c_2\rangle, \langle 4, c_3\rangle\}$$

ORDERING UNANIMITY?

Condorcet Cycle

$$\mathscr{P}^1 = \{\langle 1, c_1\rangle, \langle 2, c_2\rangle, \langle 3, c_3\rangle, \langle 4, c_4\rangle\}$$

$$\mathscr{P}^2 = \{\langle 1, c_2\rangle, \langle 2, c_3\rangle, \langle 3, c_4\rangle, \langle 4, c_1\rangle\}$$

$$\mathscr{P}^3 = \{\langle 1, c_3\rangle, \langle 2, c_4\rangle, \langle 3, c_1\rangle, \langle 4, c_2\rangle\}$$

$$\mathscr{P}^4 = \{\langle 1, c_4\rangle, \langle 2, c_1\rangle, \langle 3, c_2\rangle, \langle 4, c_3\rangle\}$$

LINEARIZABILITY

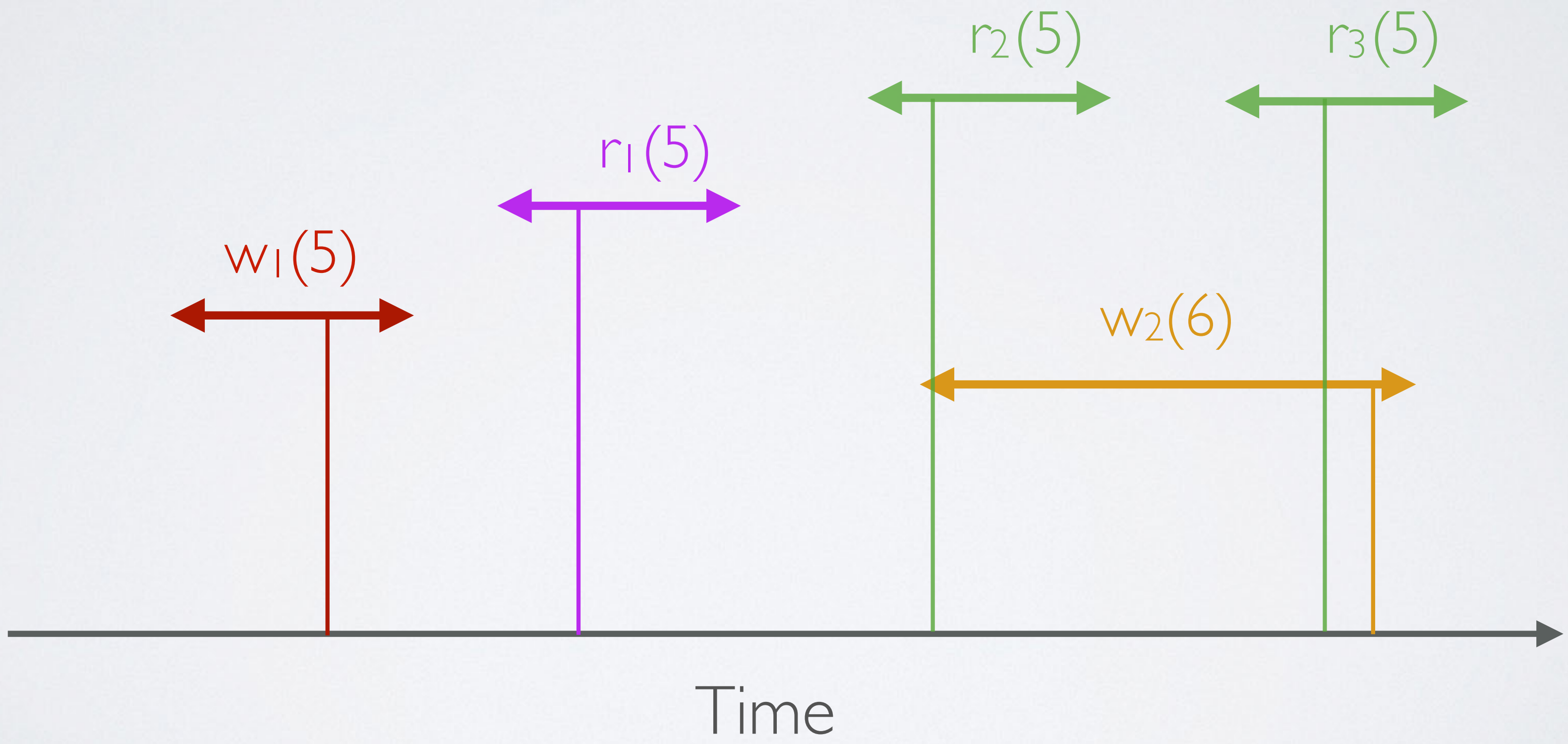
Herlihy & Wing, 1987

A correctness condition for concurrent objects

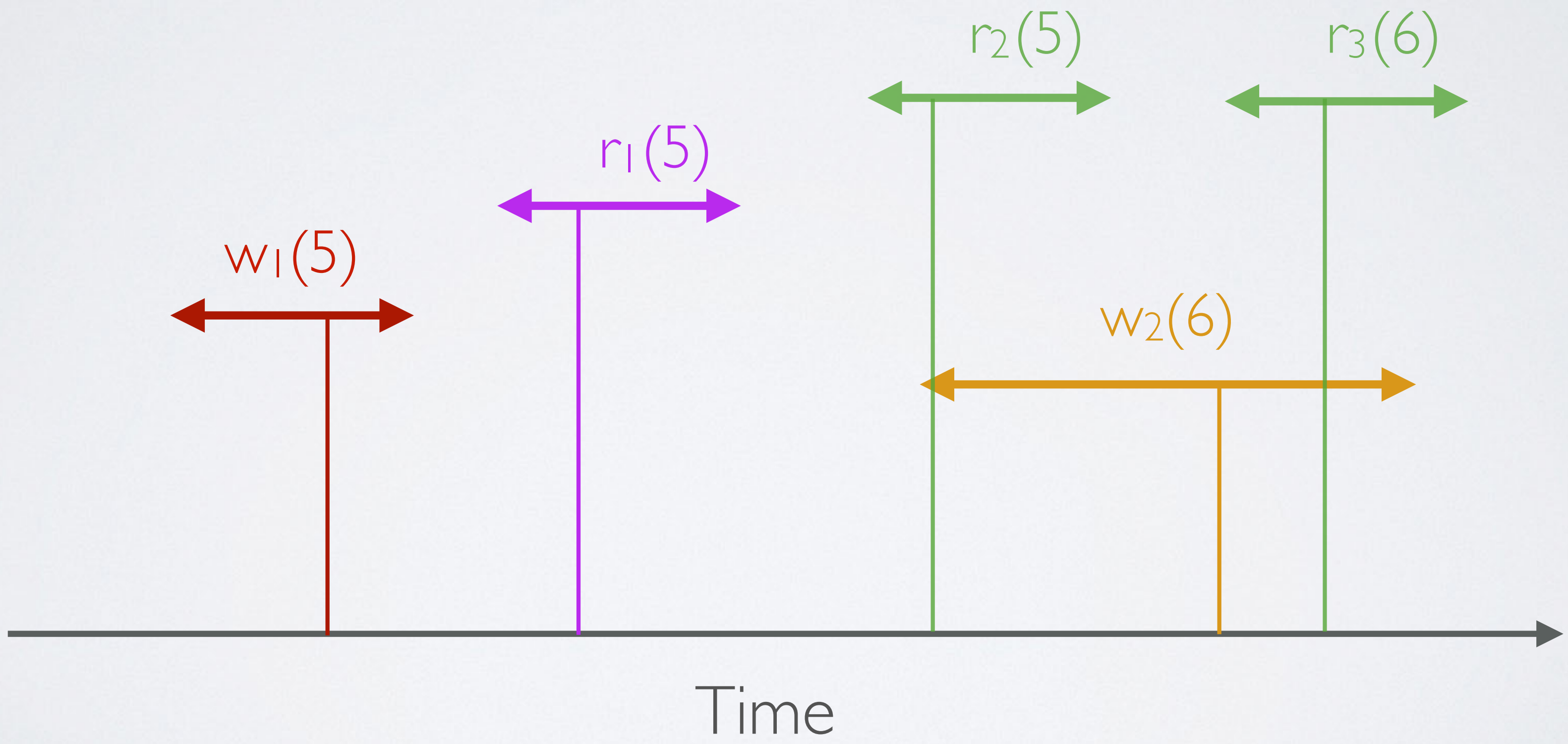
Assign each method a invocation time i and an response time r

Object behaves as if operation happened in an instant (the linearization point) in the interval [i, r]

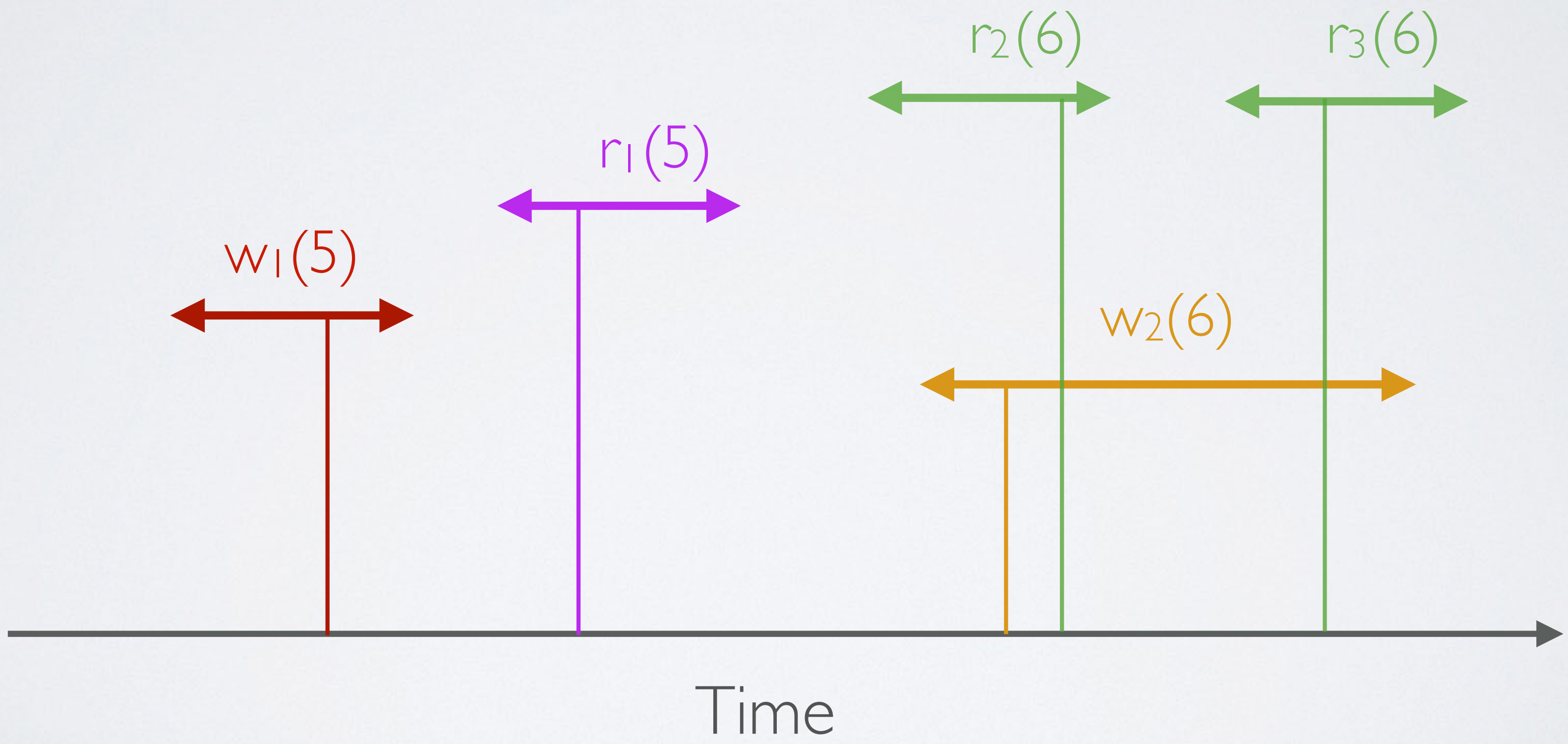
LINEARIZABLE REGISTERS



LINEARIZABLE REGISTERS

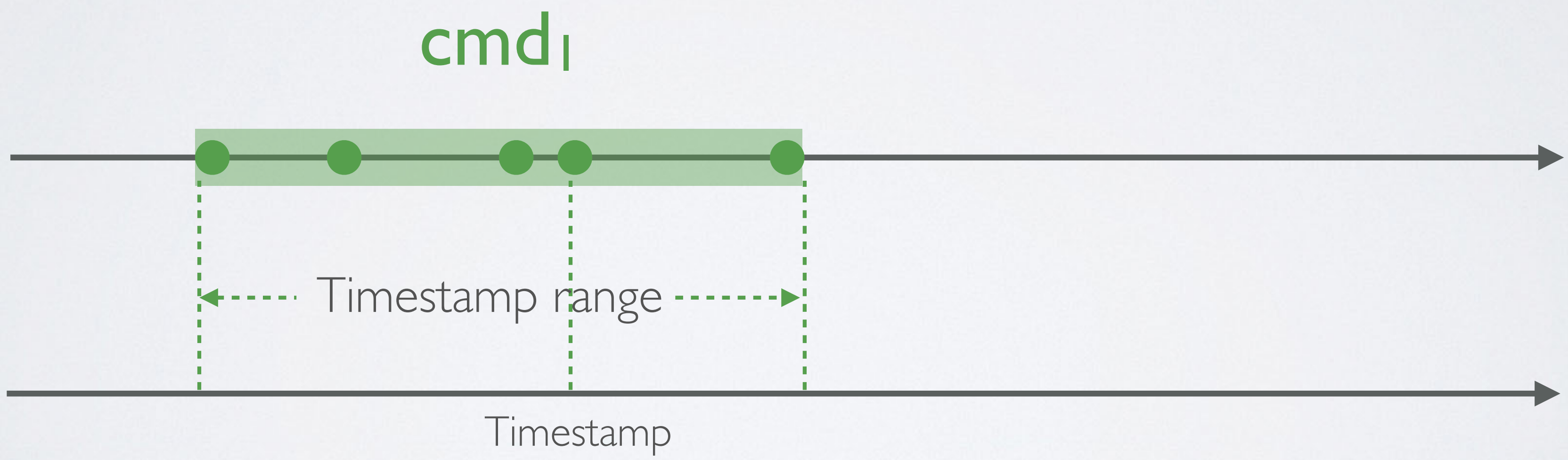


LINEARIZABLE REGISTERS



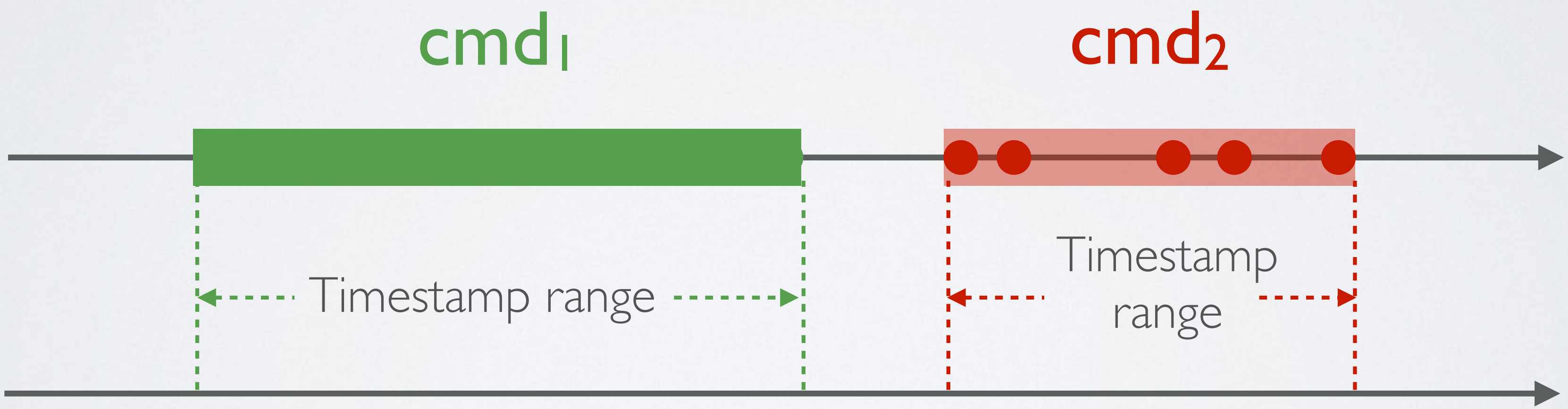
ORDERING LINEARIZABILITY

* Express ordering preferences as timestamps
  * no circularity



ORDERING LINEARIZABILITY

- Express ordering preferences as timestamps
  - no circularity



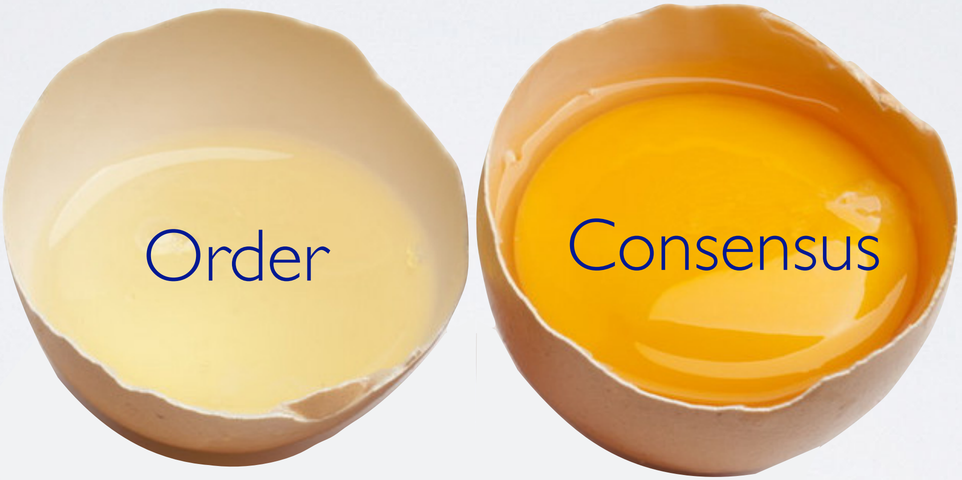
Order commands according to their "linearization" point

POMPĒ

ORDER LINEARIZABLE SMR

A NEW ARCHITECTURE FOR BYZANTINE SMR

A NEW ARCHITECTURE FOR BYZANTINE SMR



A NEW ARCHITECTURE FOR BYZANTINE SMR

Ordering phase decides the ordering of commands

Prevents Byzantine nodes from controlling ordering

Consensus phase periodically freezes a prefix of the ledger

Can preserve benefits of leader-based consensus

A BYZANTINE-TOLERANT TIMESTAMP

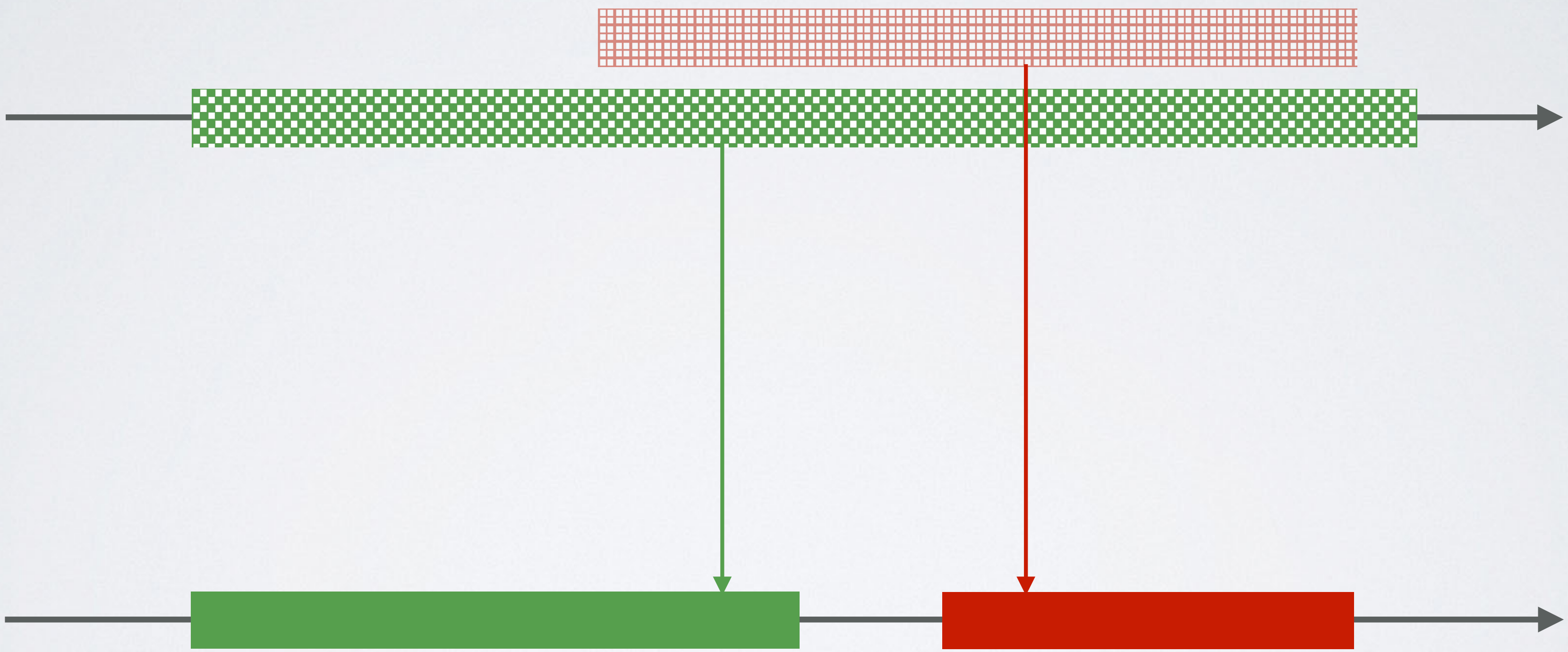
cmd_1 timestamps
by correct nodes

cmd_2 timestamps
by correct nodes

A BYZANTINE-TOLERANT TIMESTAMP

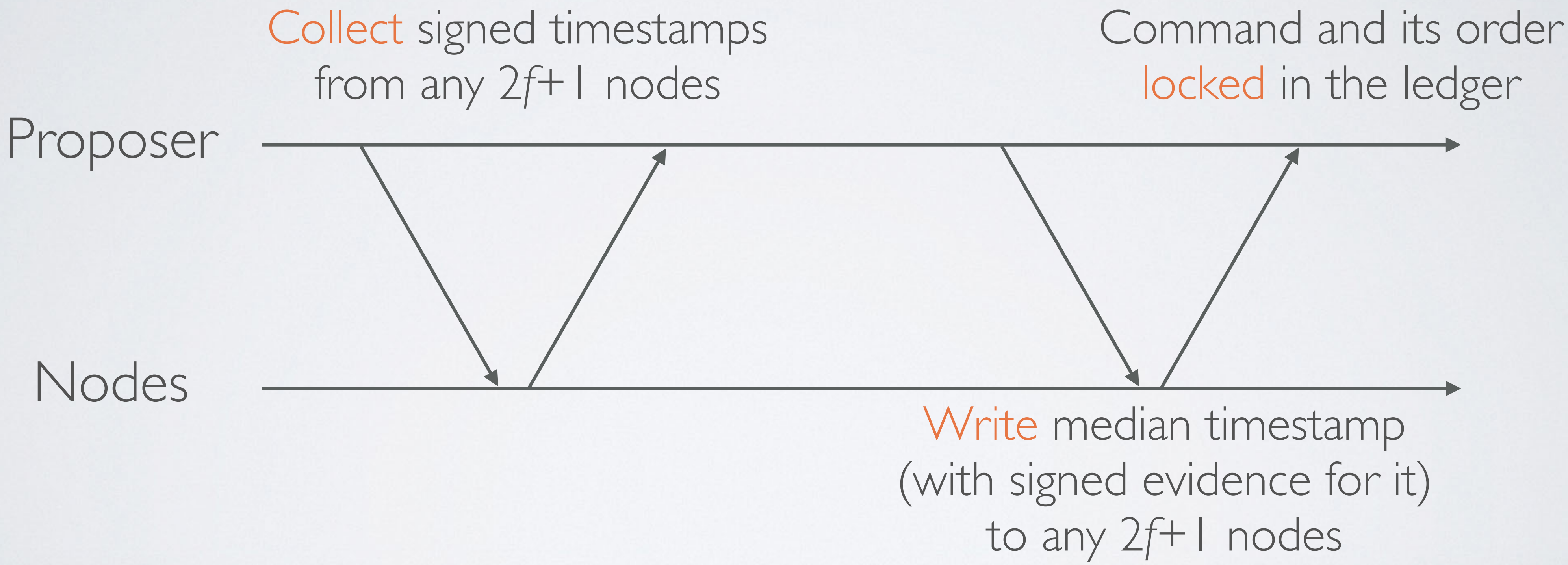
But if f out of $3f+1$ are Byzantine, then…

A BYZANTINE-TOLERANT TIMESTAMP



…the median of any $2f+1$ timestamps
falls within correct interval!

LOCKING
THE MEDIAN TIMESTAMP



CONSENSUS IN POMPĒ

Maps each consensus slot to a time interval

Waits until commands issued in current time interval are locked

Collects commands in current time interval and their timestamps

Uses any SMR protocol to add these commands to the ledger in timestamp order

HOW WELL DOES IT WORK?

How does Pompē's performance compare with state of the art BFT?

How does separating ordering from consensus impact performance?

Baselines

Concord (VMware) — SBFT consensus

Libra (Facebook) — Hotstuff consensus

BATCHING TURBOCHARGES CONSENSUS

Amortizes the cost of consensus across all commands in the batch

Essential for achieving high throughput

at the cost of higher latency

BATCHING IN POMPĒ

Batching in Pompē is safe

No tradeoff between batch size and Byzantine control

Consensus phase yields lower throughput for a given latency

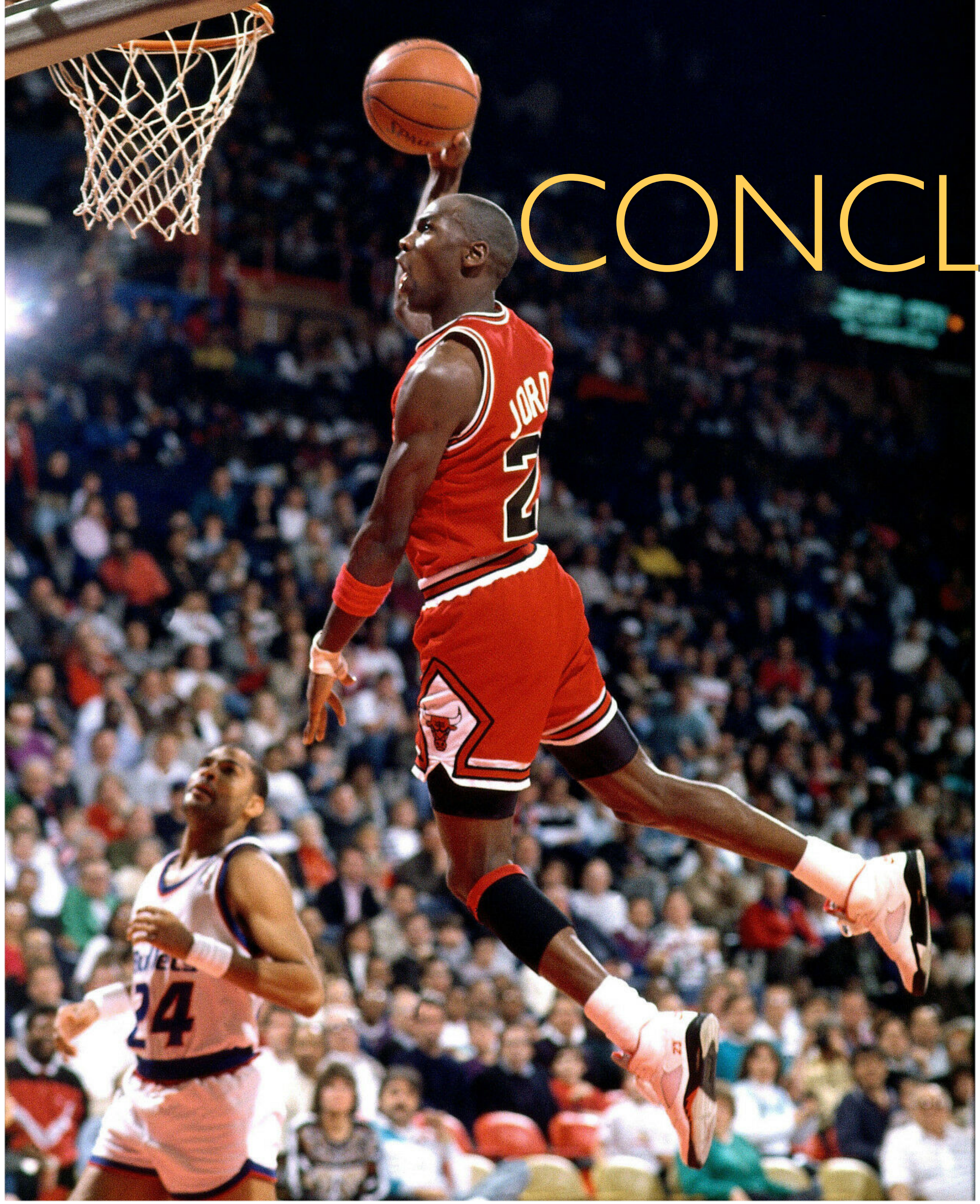
Nodes must produce & validate signed timestamps during ordering

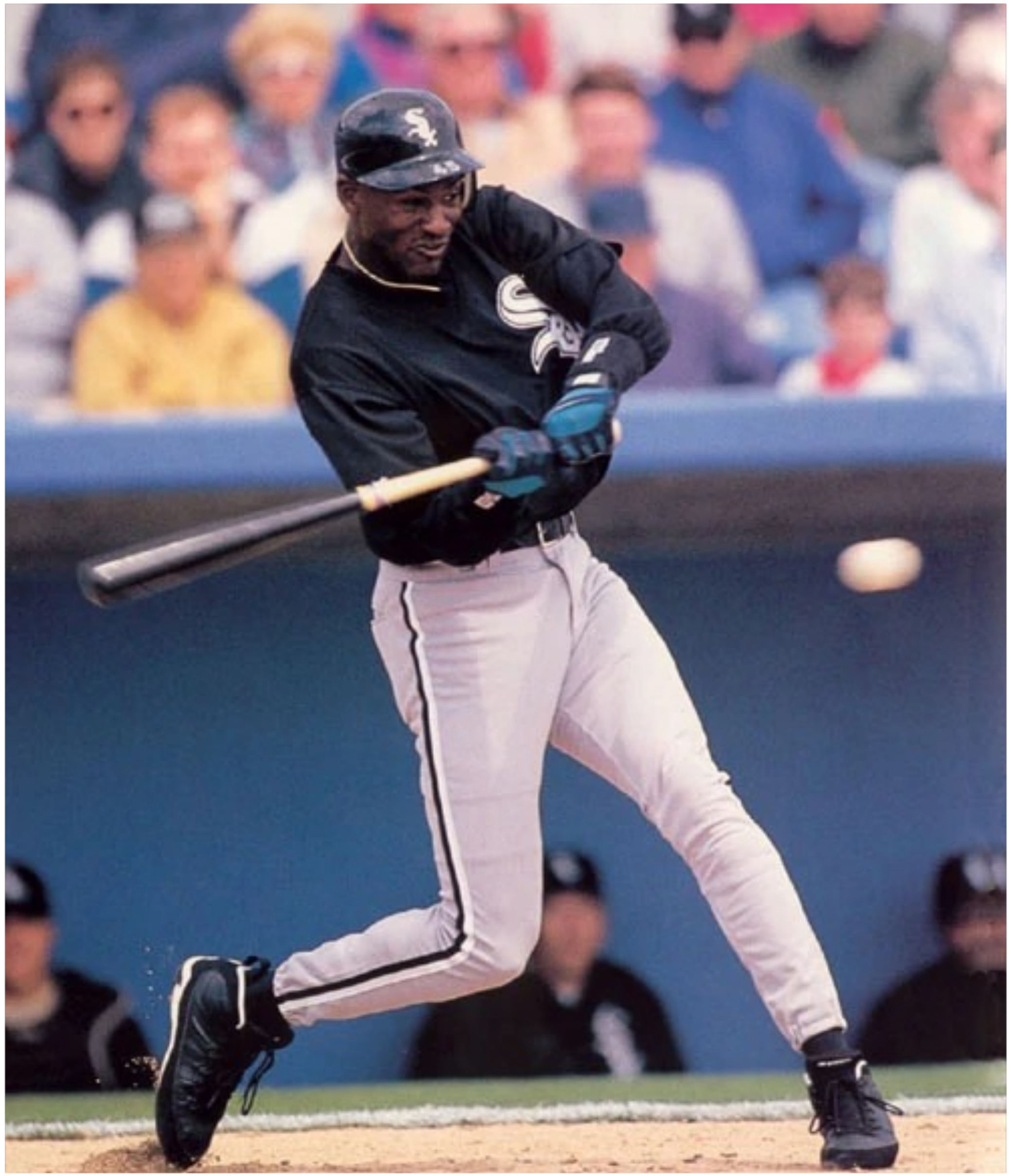
More batching opportunities!

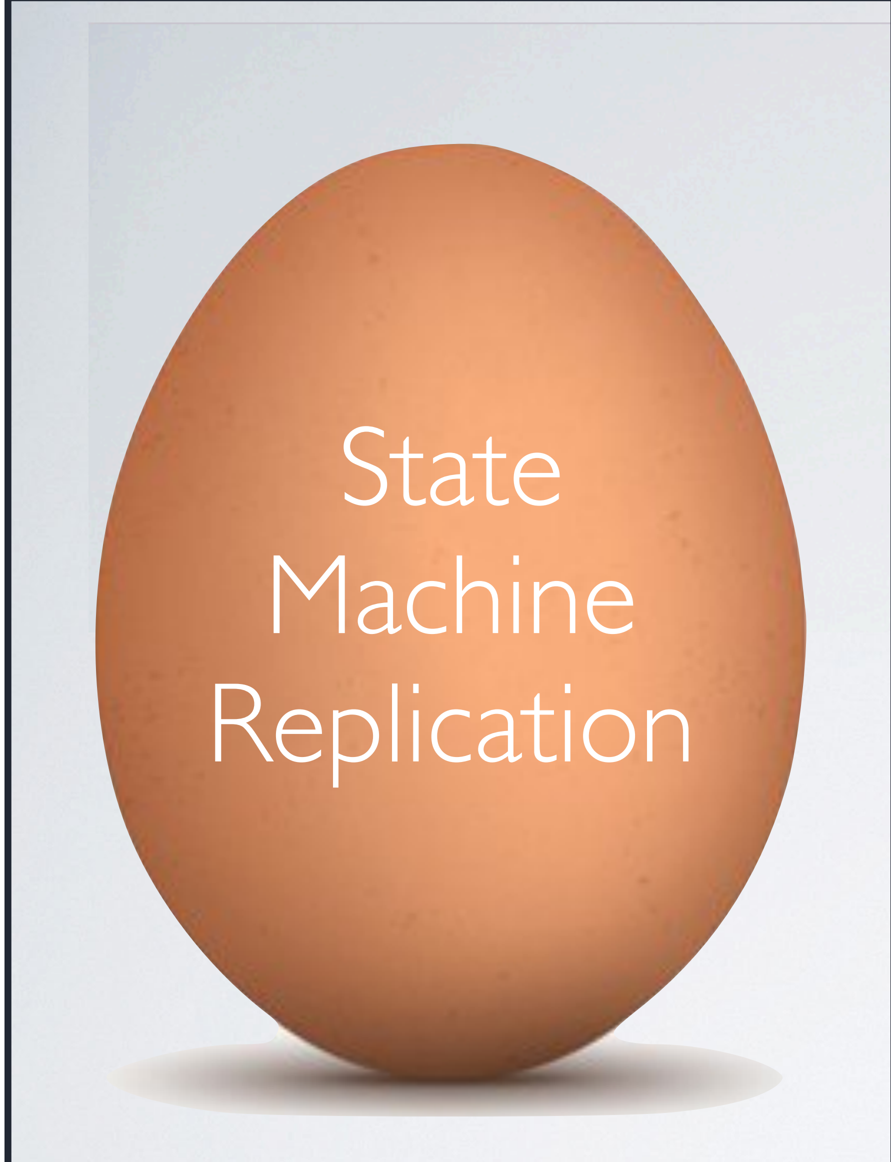
Amortize cost of ordering across commands from same node

CONCLUSIONS

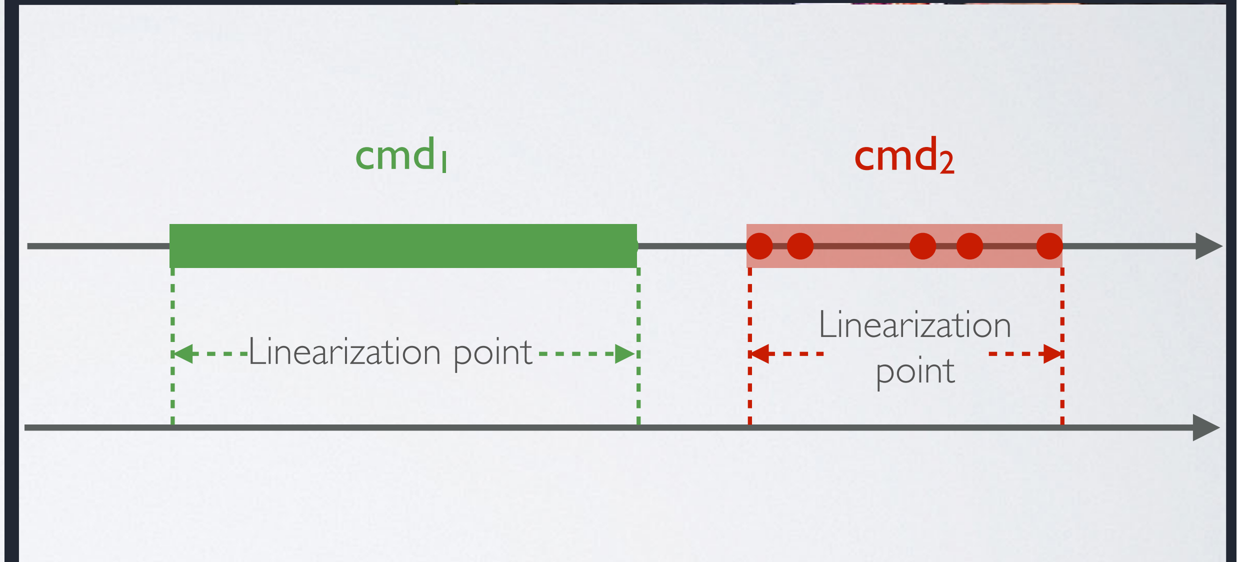
CONCLUSIONS






State
Machine
Replication


BITCOIN REVOLUTION


cmd1
cmd2
Linearization point
Linearization point


Consensus


MAXIMIANVS


Order