BFT + Blockchain

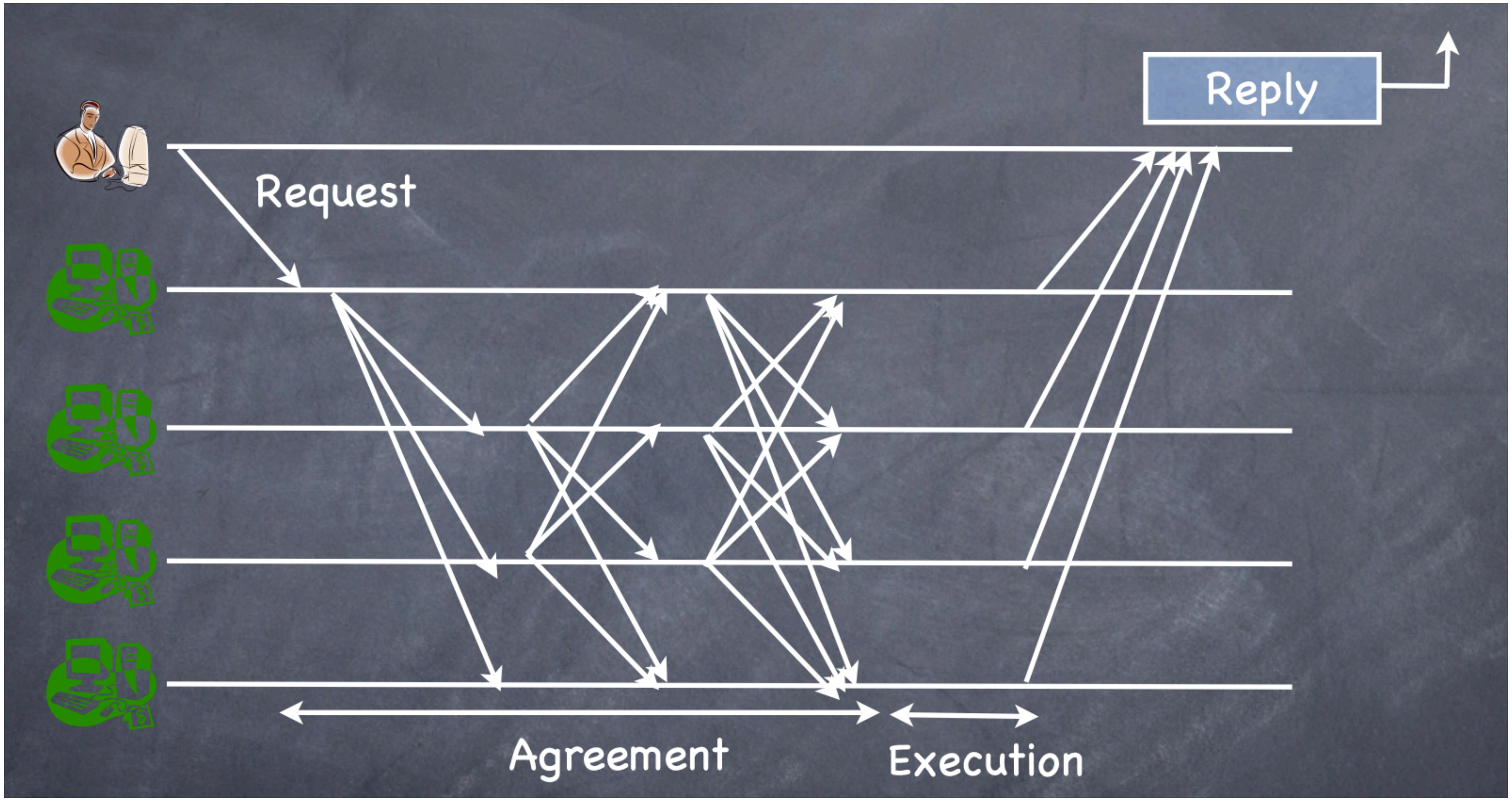
Arvind Krishnamurthy

University of Washington

Why another BFT protocol?

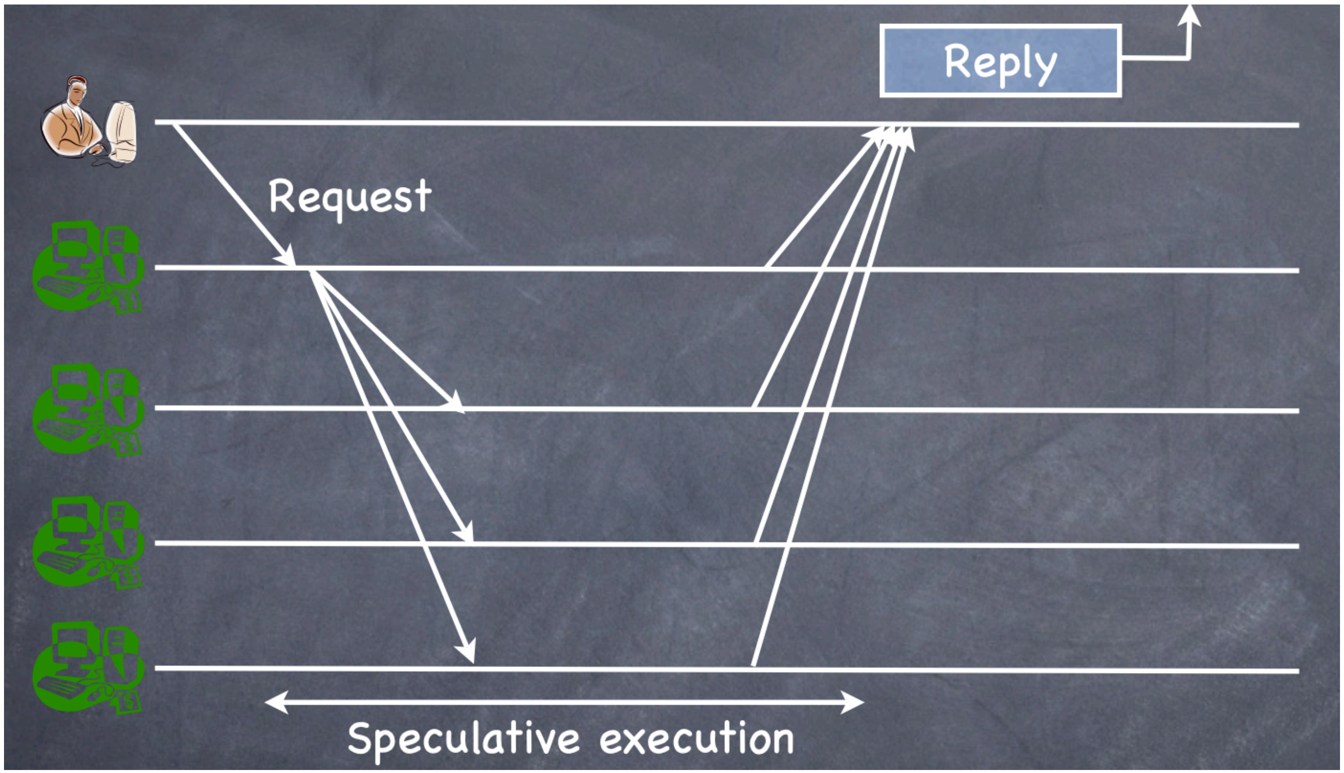
- Many BFT protocols: PBFT, HQ, Q/U, etc.
- Different protocols for different regimes
  - Number of failures tolerated
  - High request contention
  - Desire low latency
  - Replication overhead
- Zyzzyva: approach lower bounds in almost every metric

Traditional BFT Protocols



- Replicas agree on the request order before executing
  - Cost: Agreement protocol overhead

Zyzzyva: Speculative execution



- Replicas execute requests without agreement
  - Cost: No explicit replica agreement

Avoid explicit replica agreement

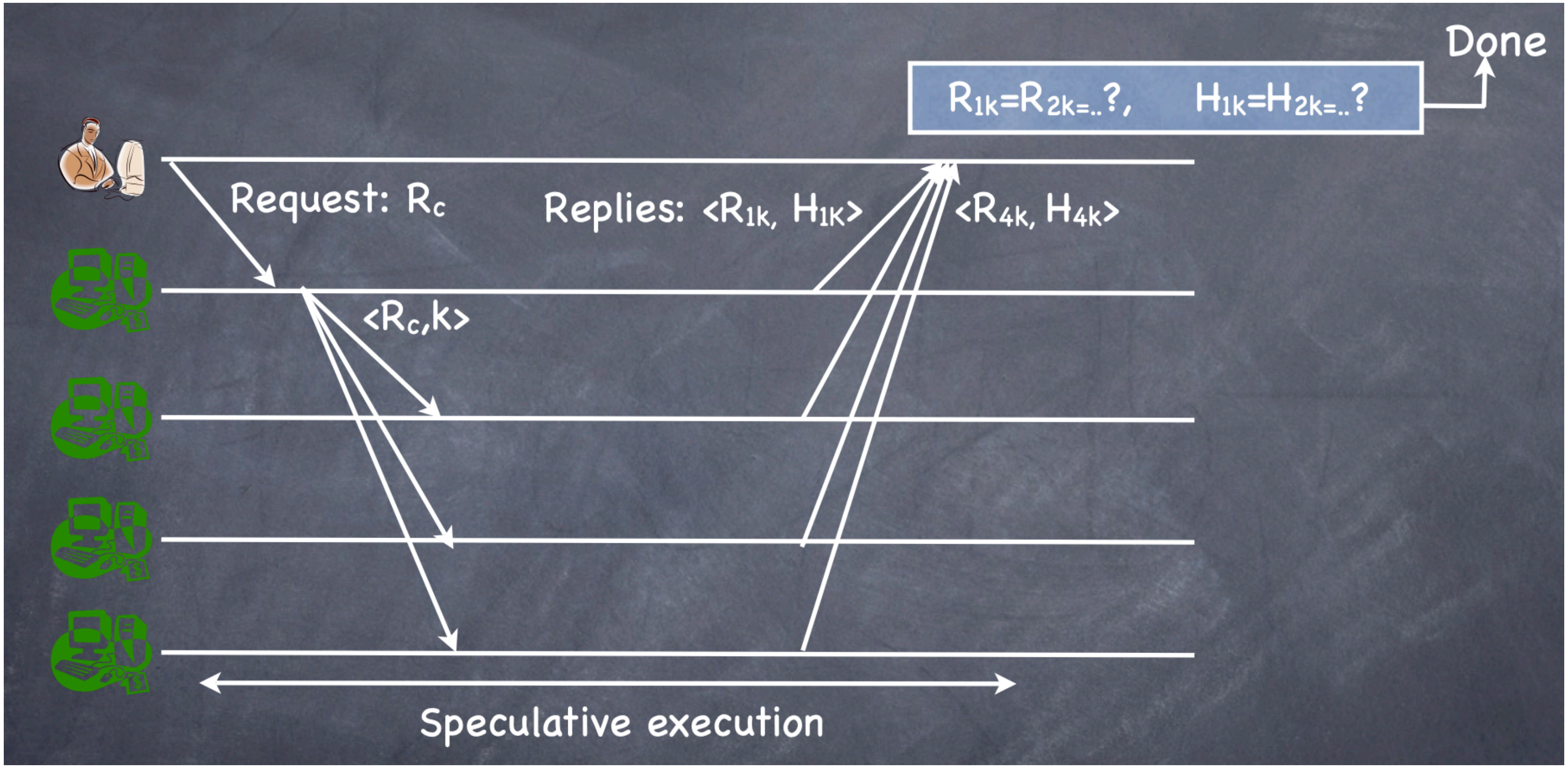
- Idea: leverage clients to avoid explicit agreement

- Intuition: output commit at the client
  - Sufficient: client knows that the system is consistent
  - Not required: replicas know that they are consistent

Client Verification

- Client verify if reply is stable before committing operation
- Request history allows clients to verify stable reply
- Replicas include request history in the replies
  - Replies include application response and request history
  - Request history: ordered set of requests executed
  - $<R_{ik}, H_{ik}>$: Reply from a replica i after executing request k

Stable: Unanimous reply

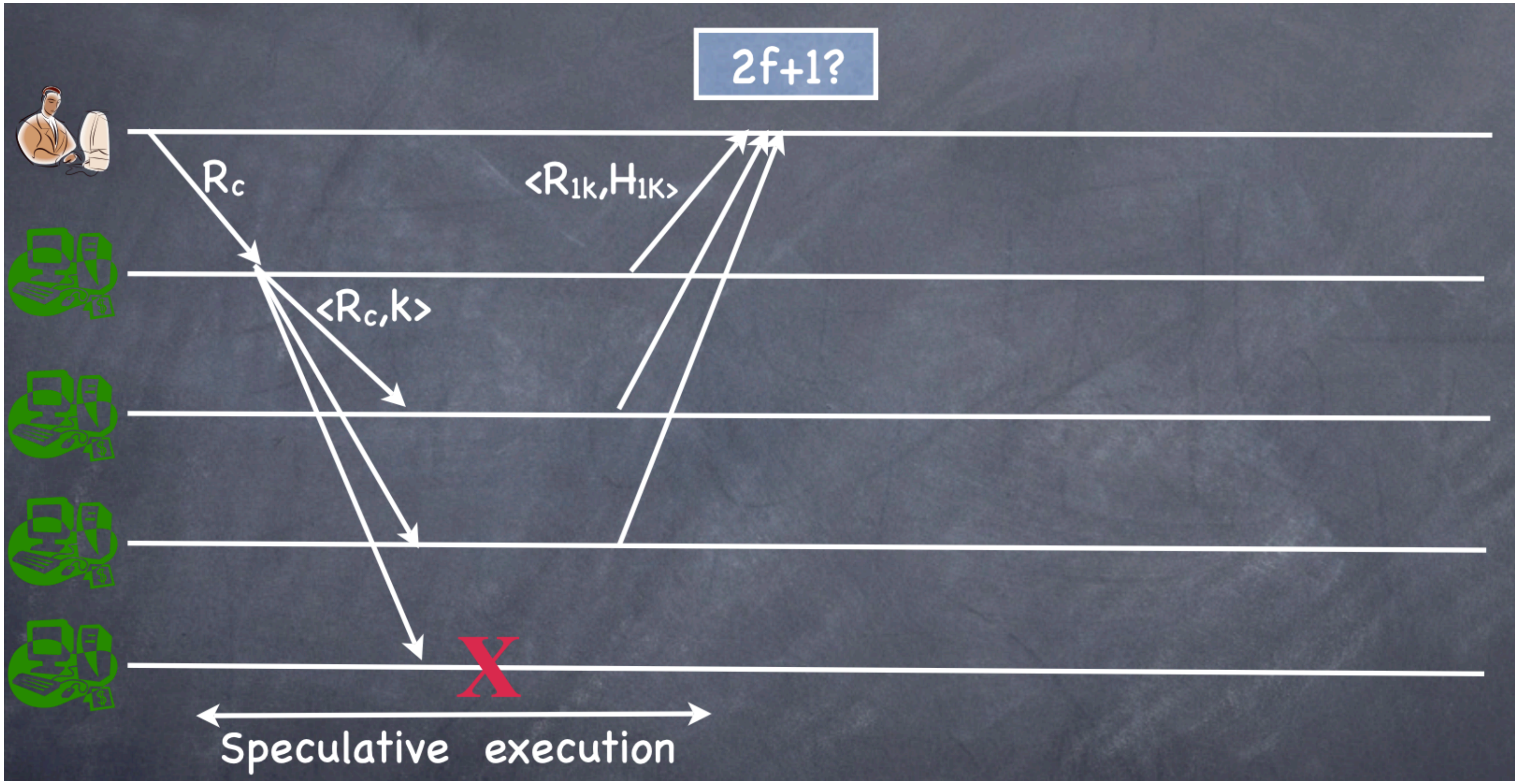


- Client commits the output when all replies match
  - All correct replicas are in consistent state

What if fast path is not successful?

- What if less than 3f+1 responses are received?
  - What if 2f+1 to 3f responses are received?
  - What if less than 2f+1 responses are received?
  - What if responses don't match?

Replies: Only majority match

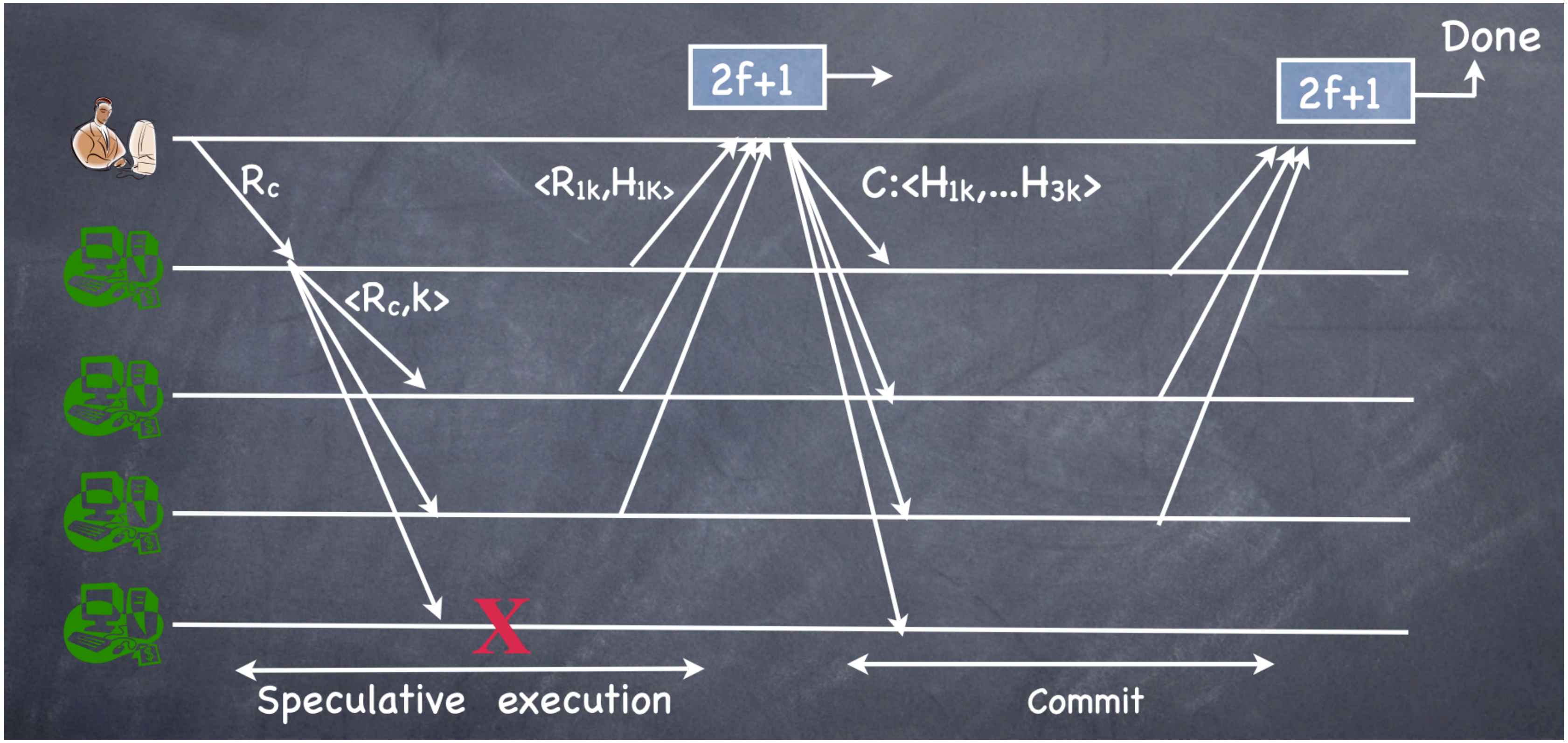


- Majority of correct replicas share the same history
  - Client receives at least 2f+1 matching replies

Stable replies with failures

- Client can make progress with additional work
- Sufficient: majority of correct replicas can prove that they share request history to other replicas
- Commit phase: client deposits commit certificate
  - Commit certificate consists of 2f+1 matching histories
  - Client commits after 2f+1 replicas respond with acks to the commit certificate

Stable reply: majority



Failures: primary or network

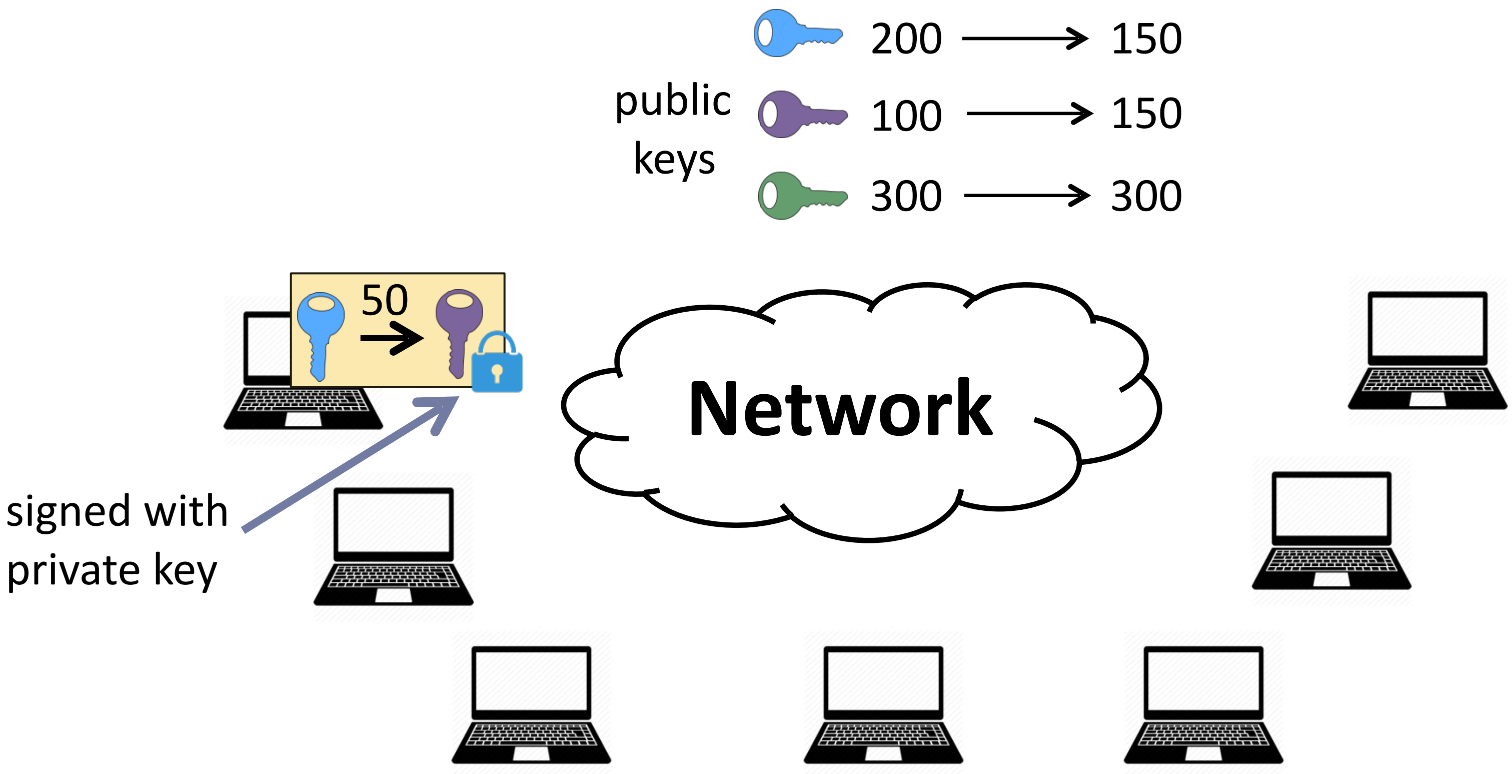
- If client receives fewer than 2f+1 responses
  - Client resends its request to all replicas
  - Replicas forward the request to the primary to ensure that the request is assigned a sequence number
    - If this results in a successful operation, then fine
    - Else, initiate a view change
- If client receives responses indicating inconsistent ordering
  - Sends a proof of misbehavior to the replicas, which initiate a view change

View Change

1. Replica initiates it by sending an accusation against the primary to all replicas ("I hate primary")
2. Replica receives f+1 accusations that the primary is faulty and commits to the view change
3. Replica receives 2f+1 view change messages
4. Replica receives a valid new view message and sends a view confirmation message to all other replicas
5. Replica receives 2f+1 matching view-confirm messages and begins accepting requests

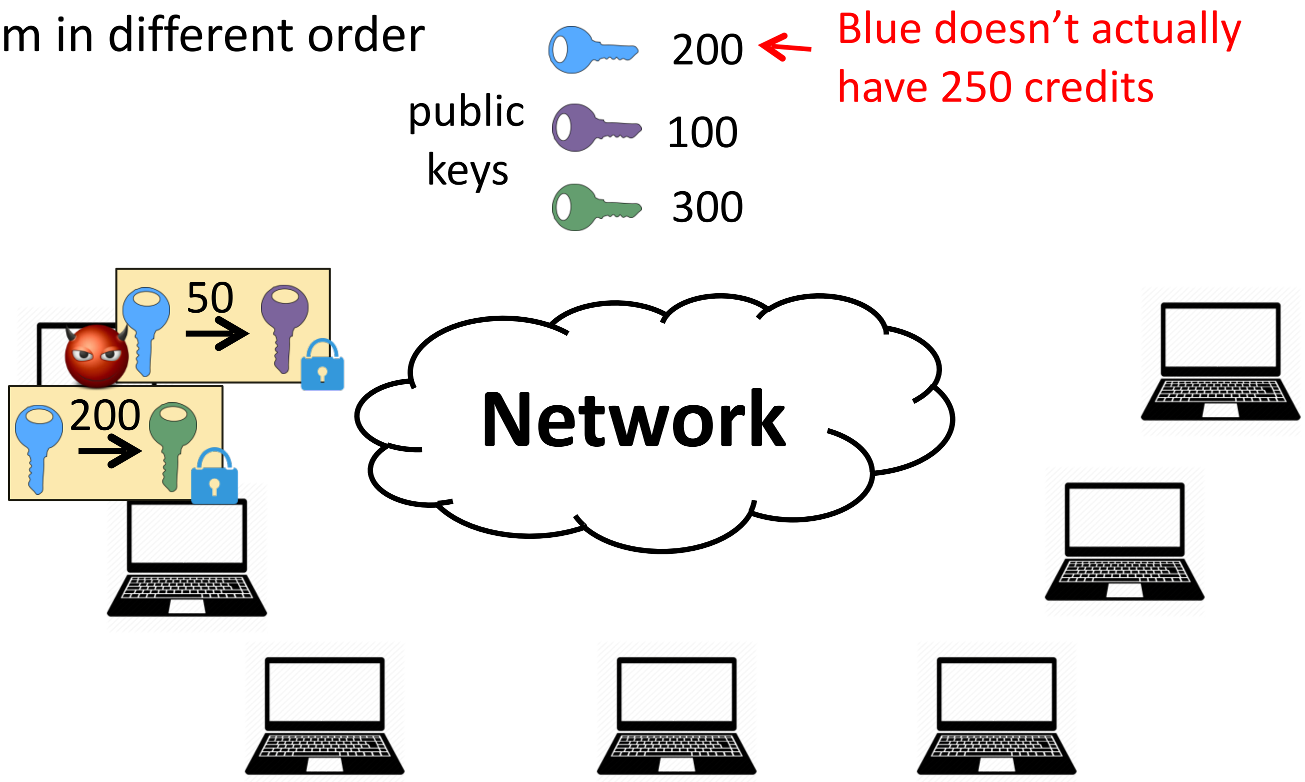
Algorand: BFT meets Blockchain

Cryptocurrencies at a high level



Double Spending Challenge

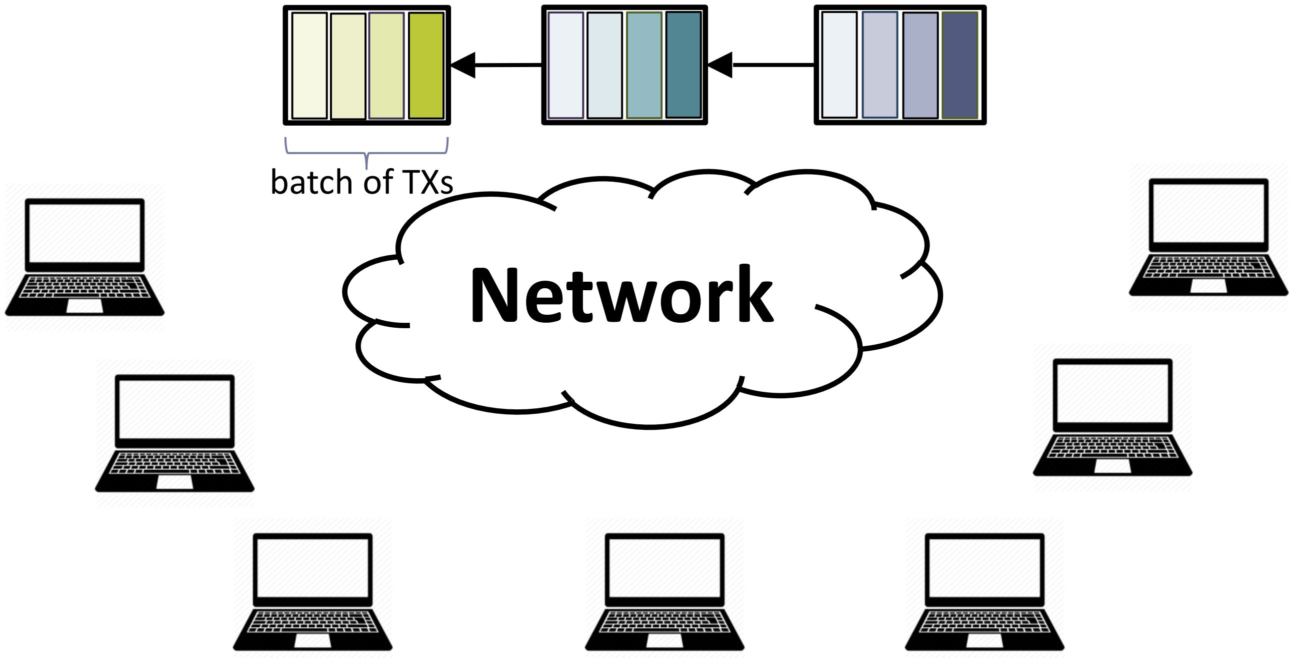
Users might not see both transactions,
or see them in different order



Solved by a public ledger

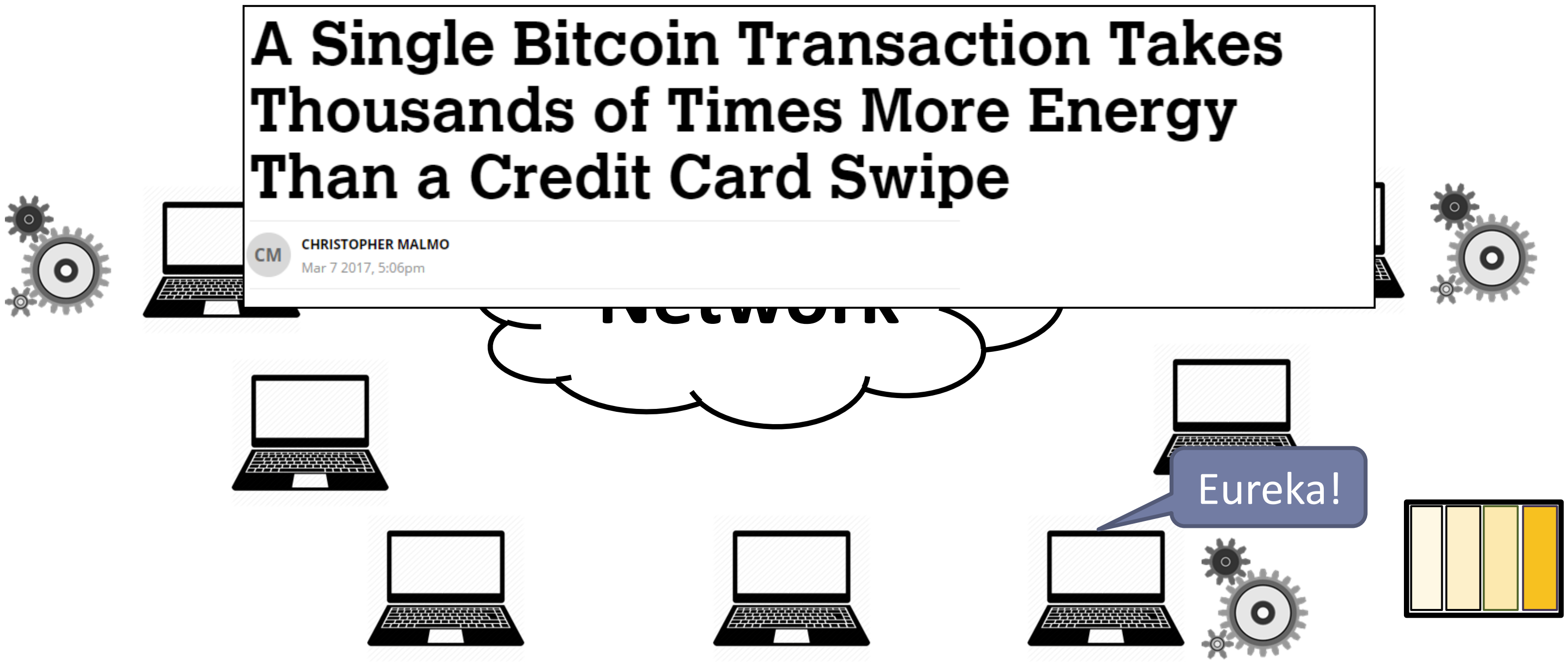
The blockchain is a public log of agreed-upon transactions

- Permissionless: anyone can join and help maintain the log



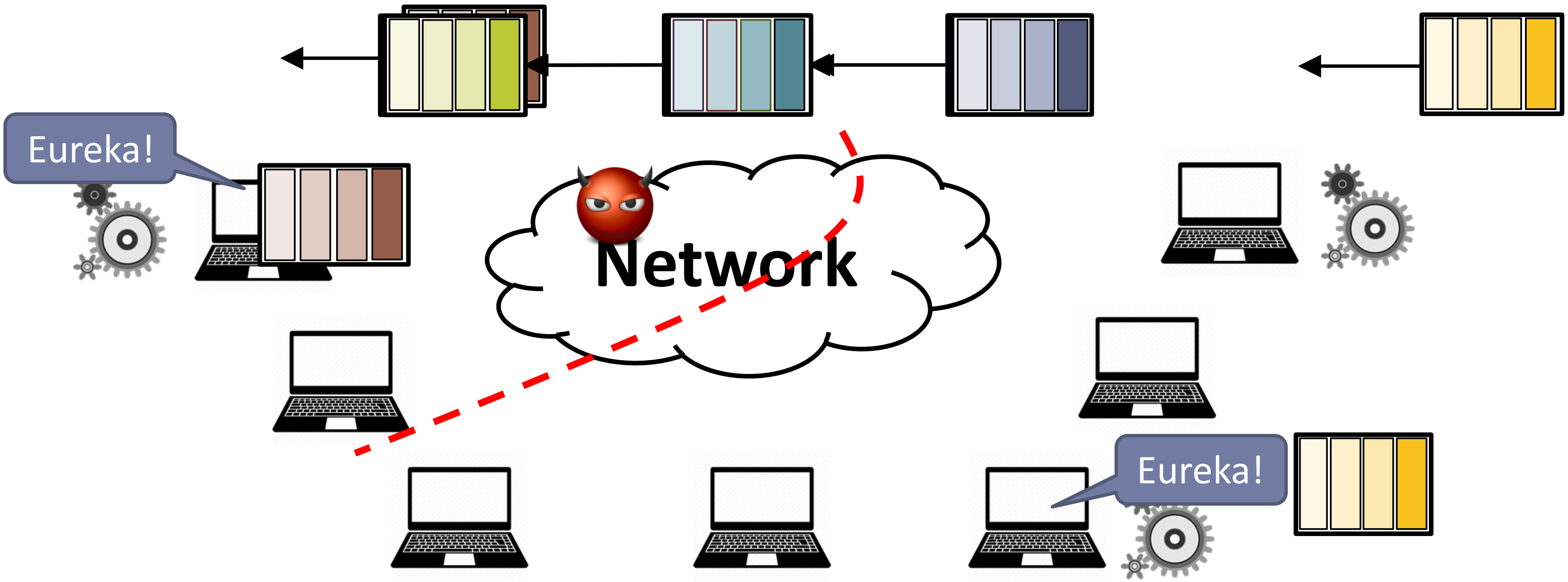
Today’s predominant cryptocurrency: Bitcoin

- Proof of Work: assume honest fraction of compute power



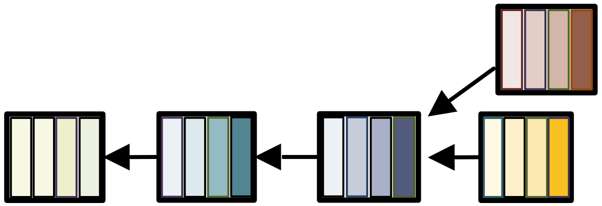
Problem with PoW based agreement: partitions

- Eclipse attacks [Heilman et al., Usenix Security15']
- Routing hijacks [Apostolaki et al., IEEE S&P 17']



Problem with PoW based agreement: forks

- Two users grow the block chain
  - transient divergent views

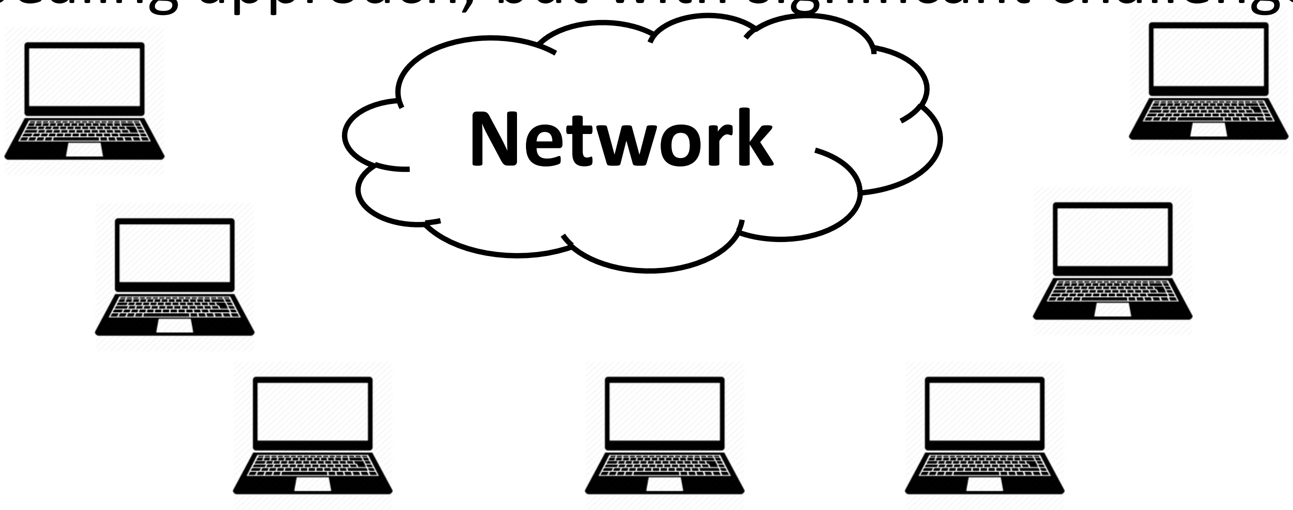


- To contend with forks, Bitcoin makes two sacrifices:
  - long time to produce a new block (10 minutes)
  - must wait for to be sure a TX not "reverted" (60 minutes)

	Energy efficient?	Throughput (MB/hour)	Latency (sec)	Confirm. time
Bitcoin	no (uses PoW)	6	600	~hour

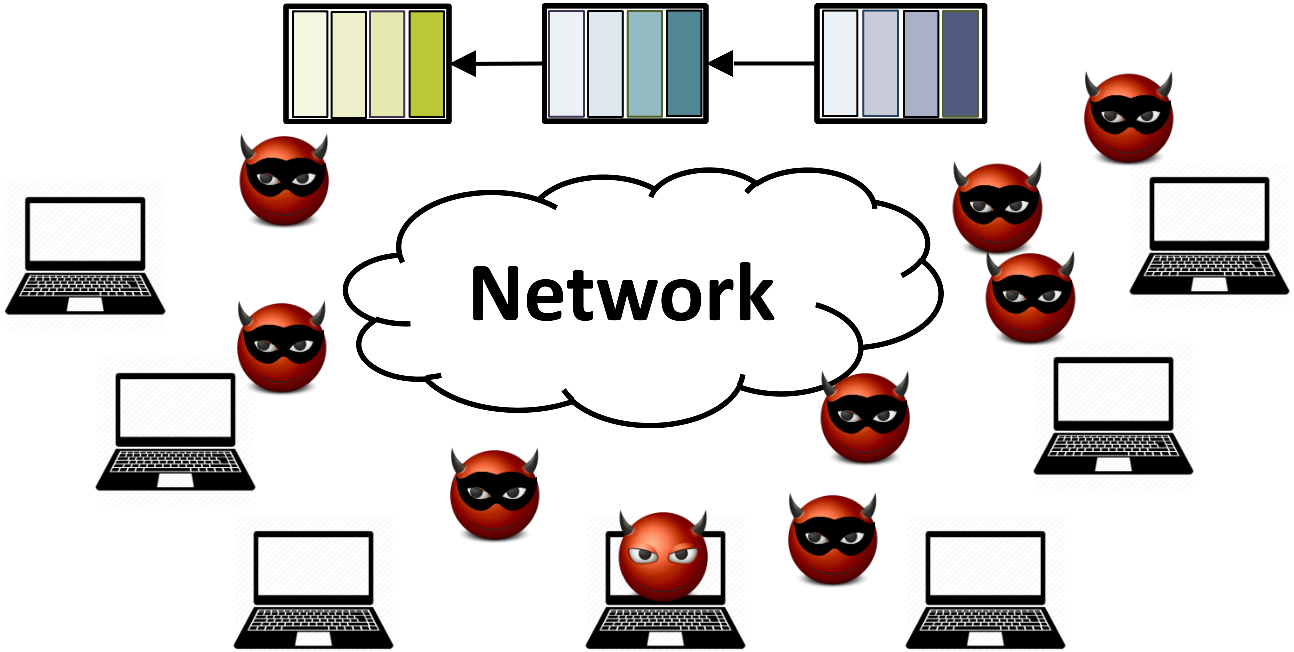
What about Byzantine Agreement (BA)?

- Allows to establish agreement on each block despite malicious participants
- There is a long line of BA research
- Appealing approach, but with significant challenges…



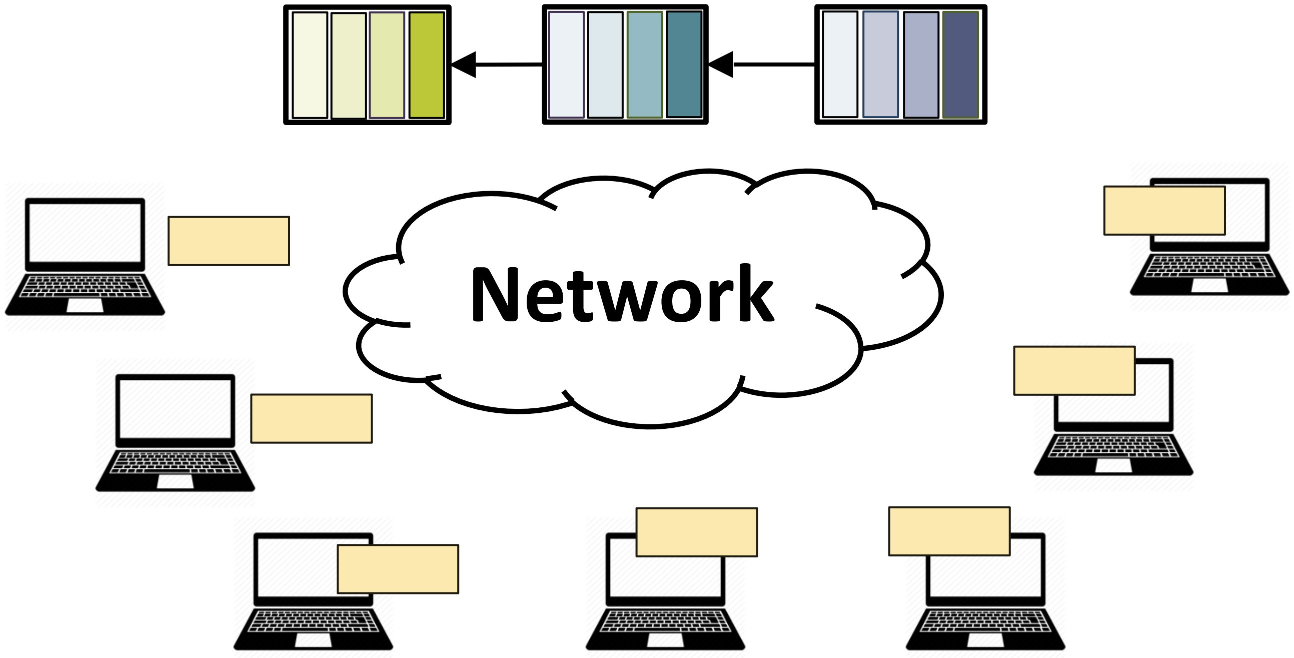
Security challenge

- Need more than const fraction of honest users
- Cryptocurrency setting is open: pseudonyms are a problem



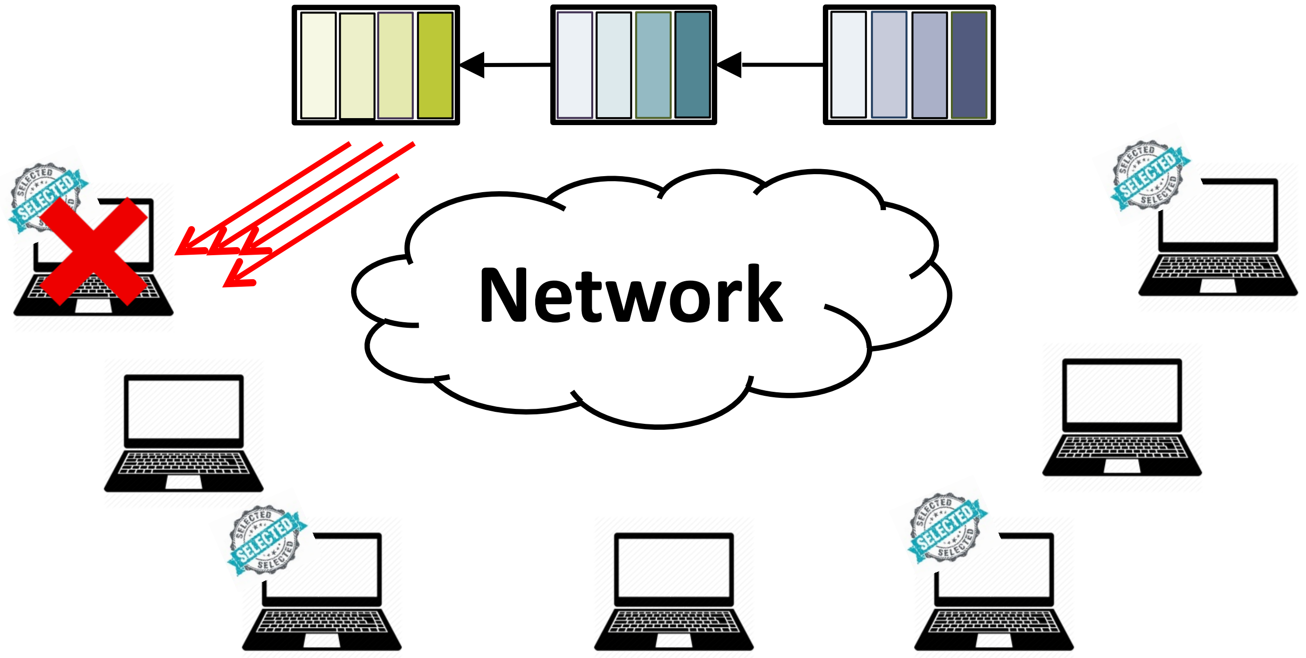
Scale challenge

- Byzantine agreement participants broadcast
- We need to support millions of users: doesn't scale



Availability challenge

- Could *sample committee* to scale Byzantine agreement
  - but, committee members can be targeted and taken offline

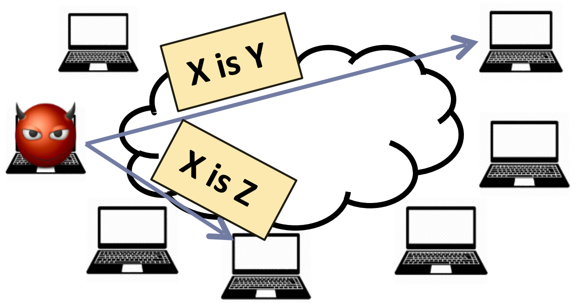


Algorand

- Algorand: scalable permissionless cryptocurrency using BA
  - sybil-resilience: users weighted by money (i.e. proof-of-stake)
  - scalability: non-interactive committee members sampling
  - availability: replace committee members after they speak
- Evaluation:
  - commit block in under 1 min, achieve 750MB/hour throughput

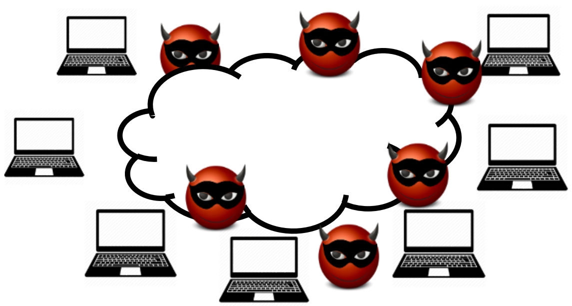
Threat model: the attacker can…

send conflicting messages to users

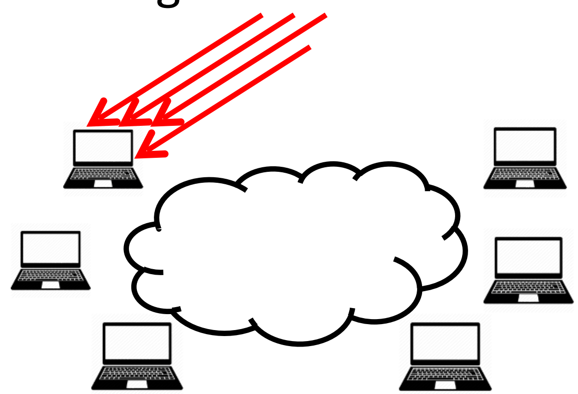


partition the network for bounded time

have many pseudonyms



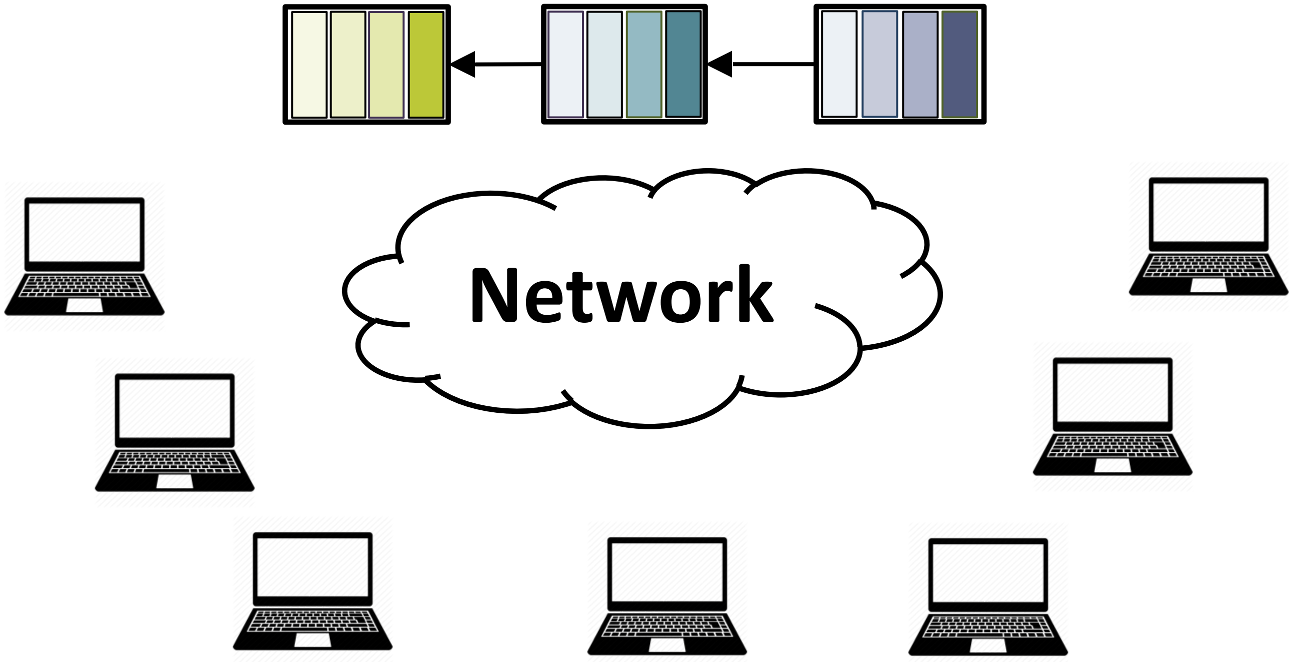
target some users



hold up to 1/3 of the wealth

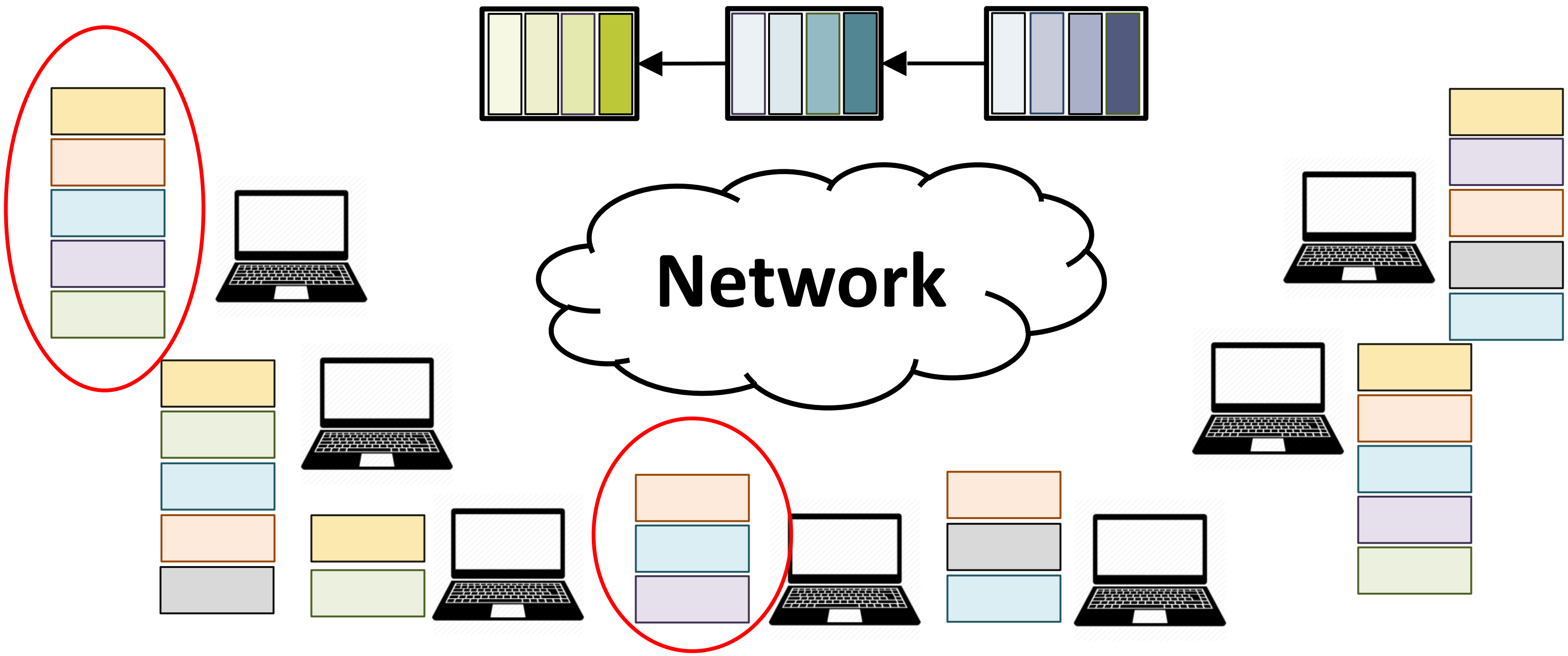
Algorand's gossip network

- Node relays msgs to a few peers, who relay to their peers…
  - All messages are signed by the origin



What is the block to agree on?

- Users have different views of pending TX

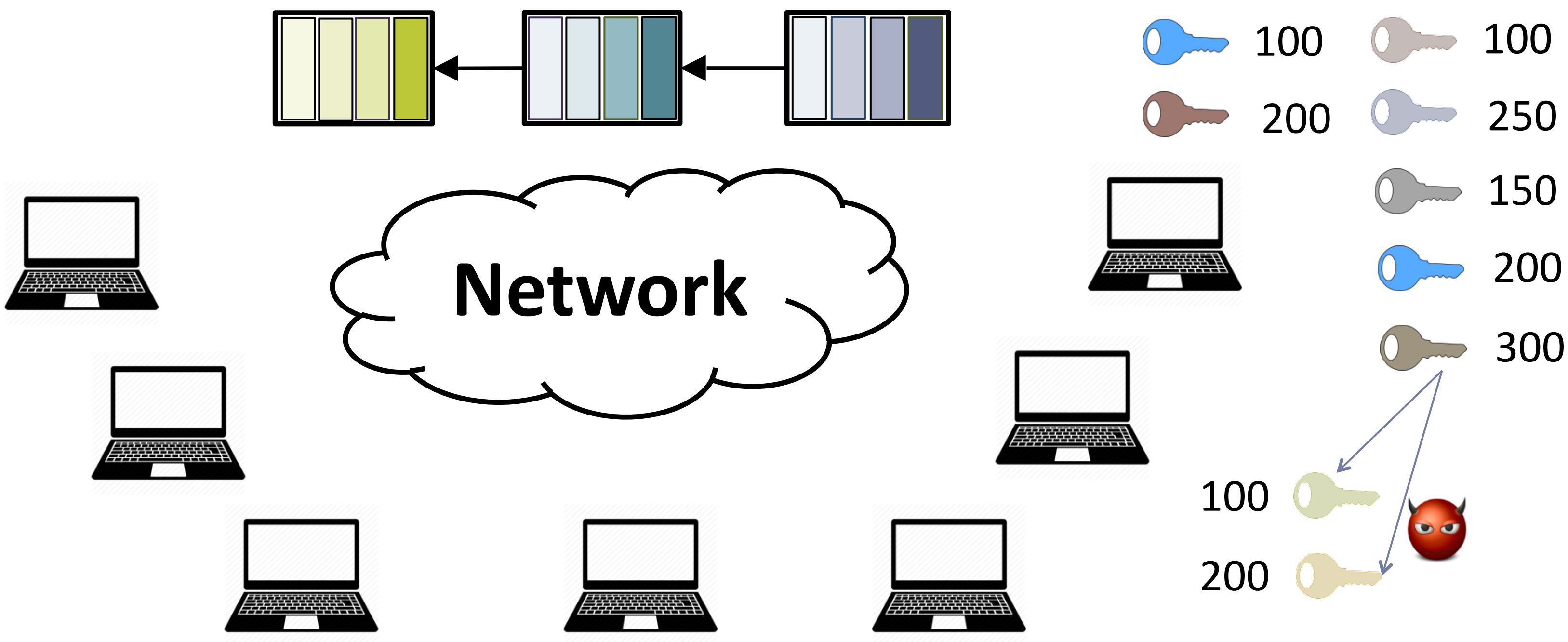


Someone proposes a block. Who?

- Can’t have everyone propose
  - high overhead, doesn’t scale
- Can’t have one user in charge
  - single point of failure
- Solution: non-interactive verifiable sampling

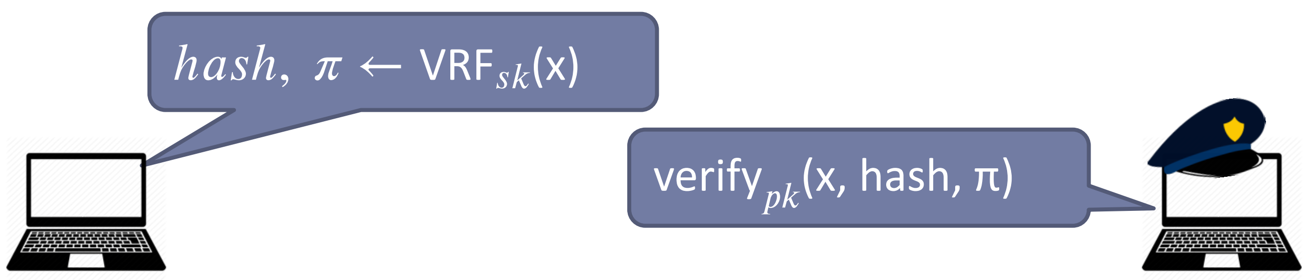
Money as weights

- PKs assigned to weights by relative fraction of money
  - attacker has to split wealth between pseudonyms



Non-interactive verifiable sampling

- Crypto tool: verifiable random functions
  - *hash:* pseudorandom value (unpredictable without sk)
  - π: proof that $hash$ was computed correctly
  - VRF is deterministic: a public key maps *x* to one *hash*

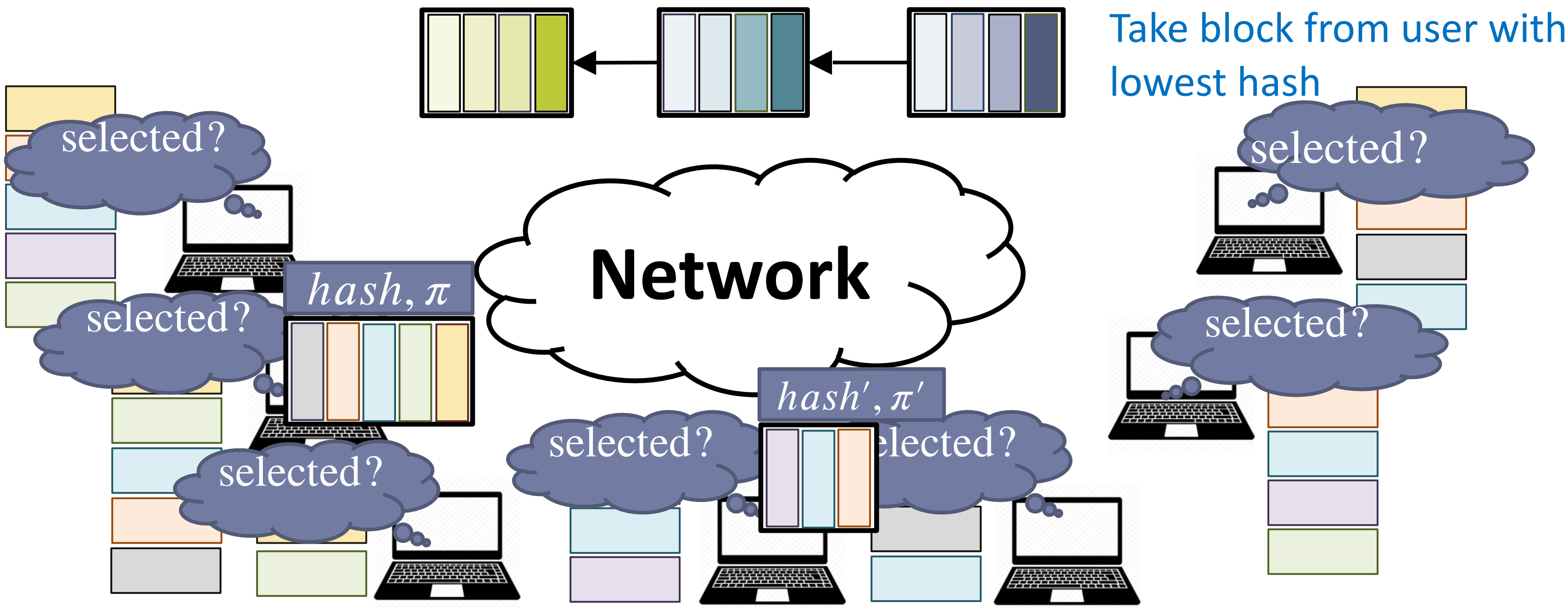


Prover, has secret key sk

Verifier, has public key pk

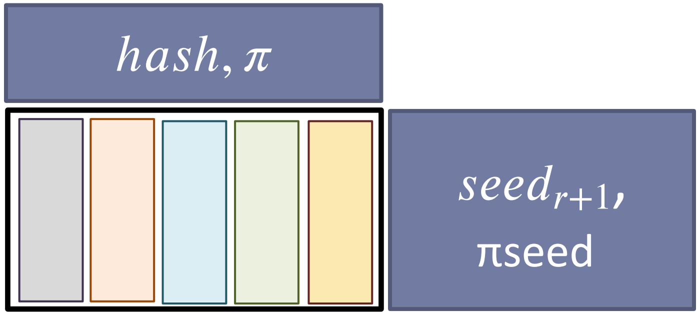
Block proposers

- Choose which transactions go in the next block
- We need: not too many, but at least one (at least often)



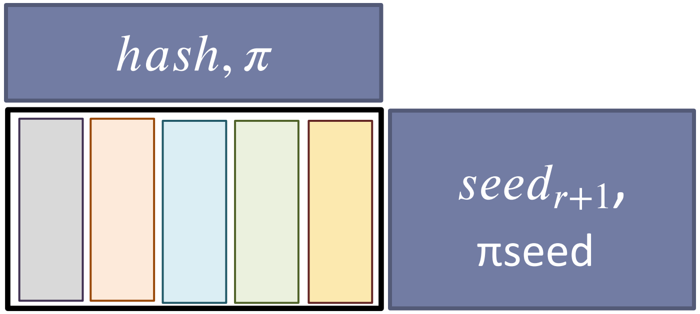
Algorand blocks contain...

- New transactions
- Proof that the proposer was selected
  - $hash, \pi$
- A seed for next round r+1:
  - $seed_{r+1}$, πseed $\leftarrow VRF_{sk}(seed_r \, || ``next\ seed'')$



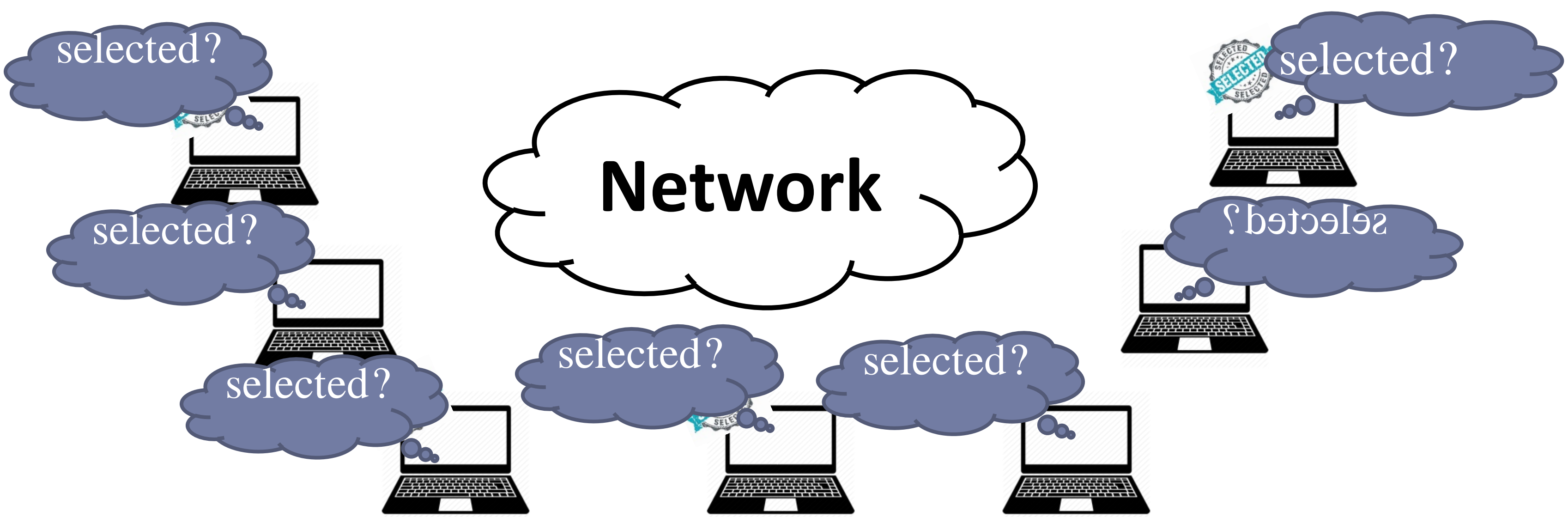
Can we take proposed block and be done?

- The block proposer may be malicious
  - proposer might send different blocks to different users
- Need a Byzantine agreement



Scale Byzantine agreement by sampling

- Recall: in traditional BA everyone broadcasts → doesn't scale
- Sample a random *committee* using weights to scale BA
  - computation using private key, produces non-interactive proof
  - *selected users originate messages, everyone gossips*

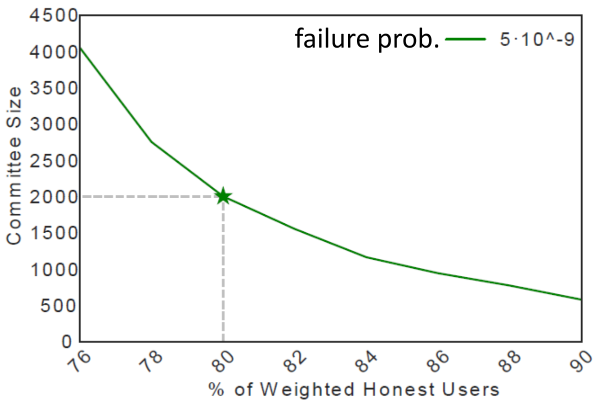


Scale Byzantine agreement by sampling

- How large should the committee be?
  - need $n \geq 3f + 1$ participants to deal with f bad users
  - but, selection is pseudorandom!
  - so we don't know n or have bound on f
- But BAs require constant decision thresholds
  - how can we set the threshold? (without knowing f and n)

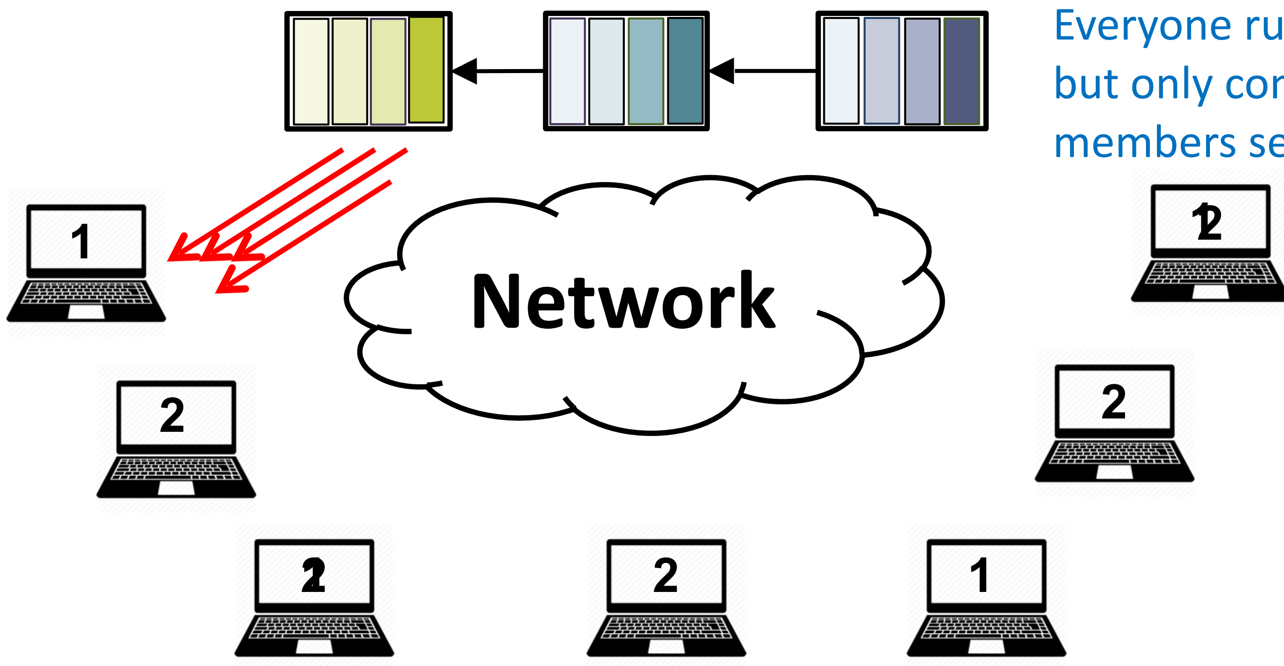
Scale Byzantine agreement by sampling

- We need to find a *thresh* that satisfies:
  - $\#good > thresh$ To reach agreement
  - $\frac{1}{2}\#good + \#bad \leq thresh$ To avoid forks
    - need more than ½ of good users to "vote for" the same value
    - therefore, cannot agree on two values



Resisting targeted attacks

- Replace committee members after they send a message
- Requirement: no private state (except static keys)



Everyone runs the protocol, but only committee members send messages

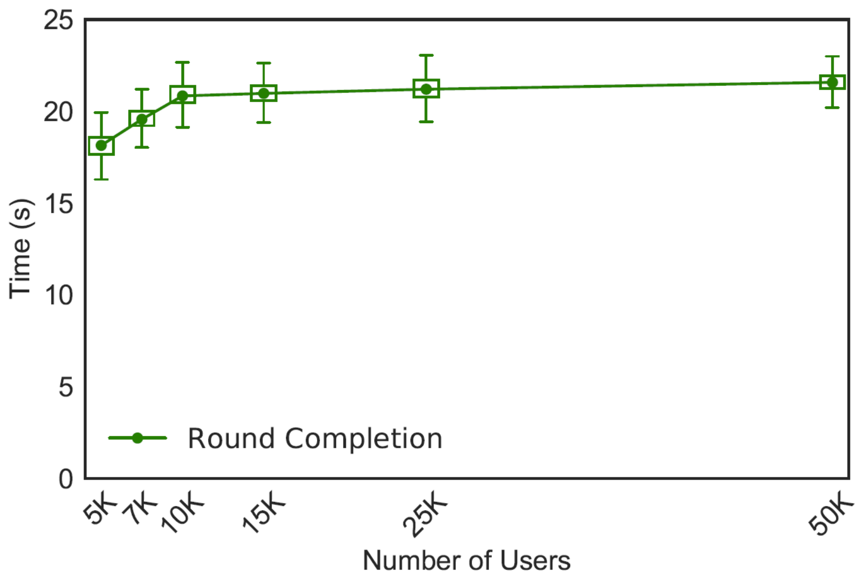
Design summary

- Weighing by money
- Sample committee based on weights using VRFs
- Replace committee at every step of Byzantine agreement

- More in the paper:
  - details of Byzantine agreement with participant replacement
  - selection procedure
  - theorems and analysis

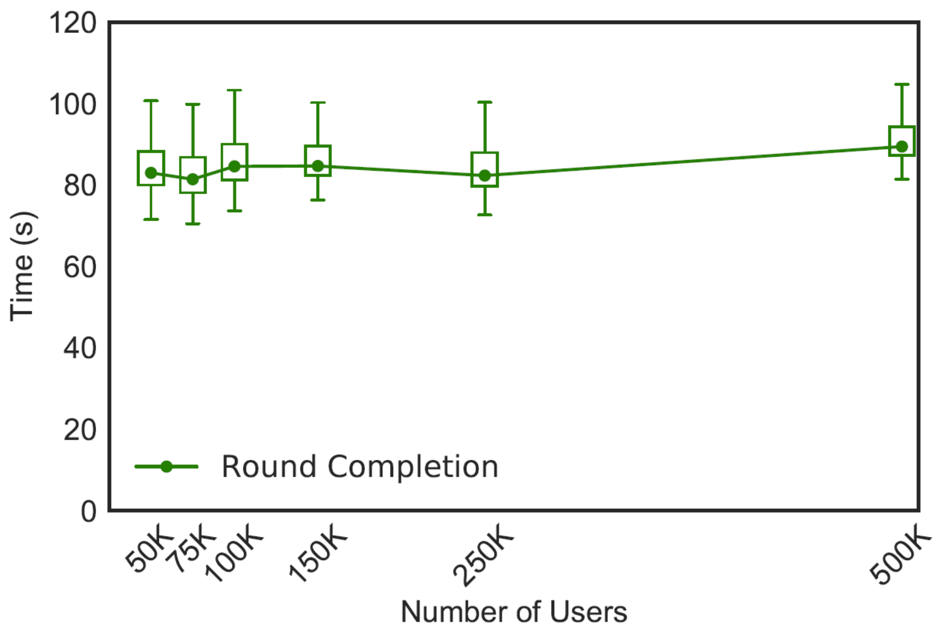
Algorand achieves low latency

- 50 users per virtual machine, 1MB block of transactions
  - average bandwidth use is 10mbps



Evaluation: scalability

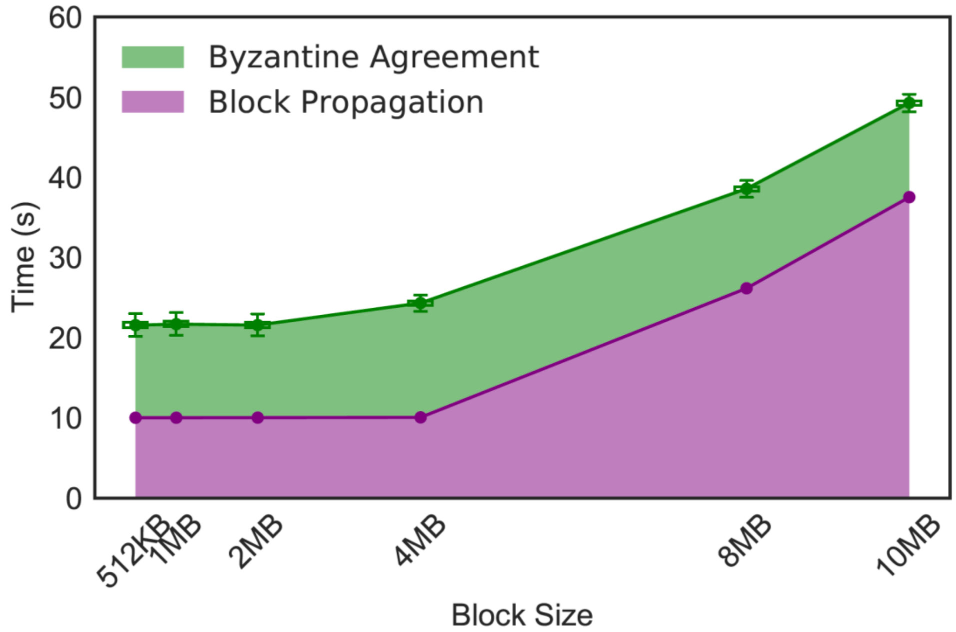
- 500 users per virtual machine, 1MB block



Algorand achieves high throughput

Algorand: up to 10MB/48sec → 750MB/hour

Bitcoin: 1MB/10min → 6MB/hour



50 users X 1,000 virtual machines

Algorand Takeaways

- Algorand doesn't utilize proof-of-work and instead weights users based on how much money they have in the system.
- Algorand is more communication efficient since it is committee based.
- However, it is not clear what incentives users have to participate in the protocol (their stake in the system notwithstanding).
- Algorand requires money holders to be online and broadcasting their address to the world.
- Algorand is really complicated.