Blockchains & Cryptocurrencies

Bitcoin

Instructors: Matthew Green & Abhishek Jain
Johns Hopkins University - Spring 2019

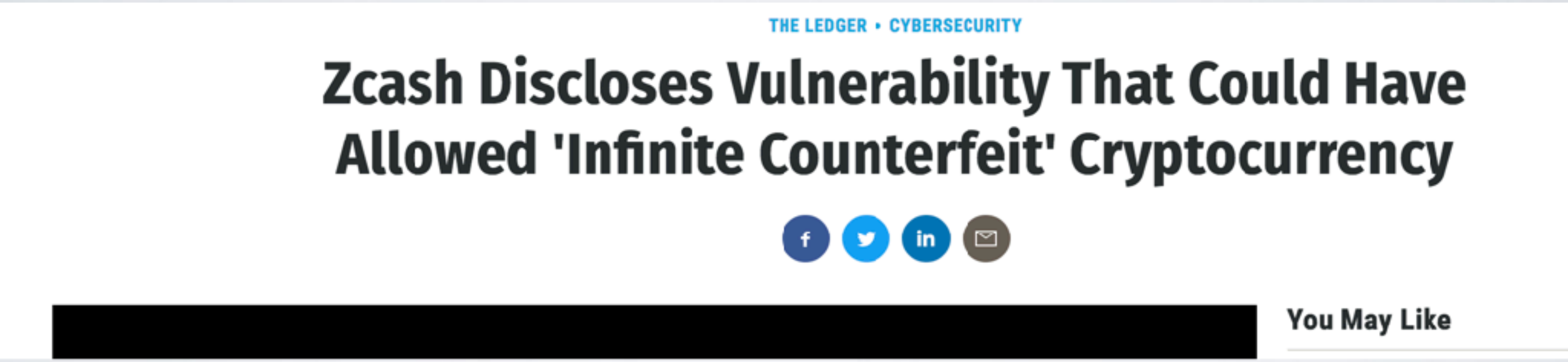
Many slides based on NBFMG

Housekeeping

- Readings: for today Bitcoin and Hashcash
  for Mon: Check Piazza

News?

News?



THE LEDGER • CYBERSECURITY

Zcash Discloses Vulnerability That Could Have Allowed 'Infinite Counterfeit' Cryptocurrency

You May Like

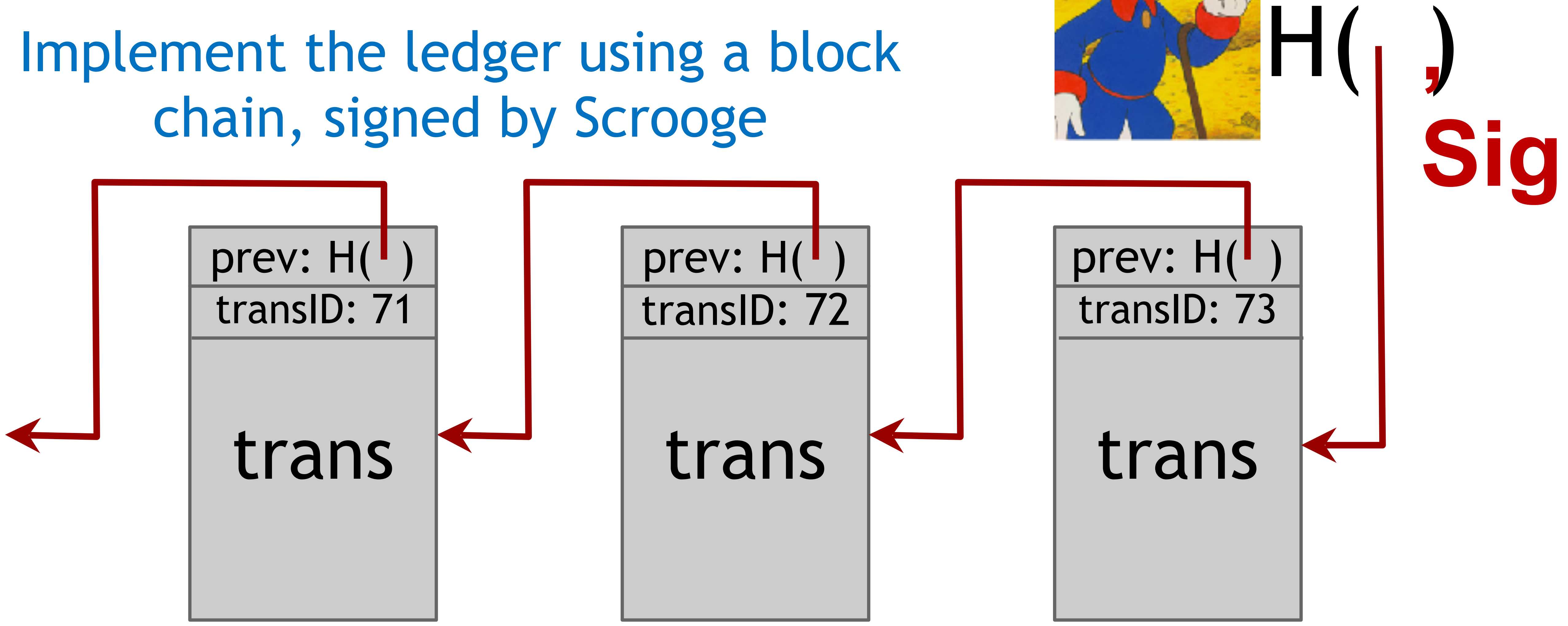
Last time & Today

- We started talking about "consensus"
- Gave some basic examples of centralized currencies
- Started to talk about why these problems were hard
- Today: finish this + Bitcoin

Review

Scrooge publishes a history of all transactions in an “append-only” ledger

Implement the ledger using a block chain, signed by Scrooge



optimization: put multiple transactions in the same block

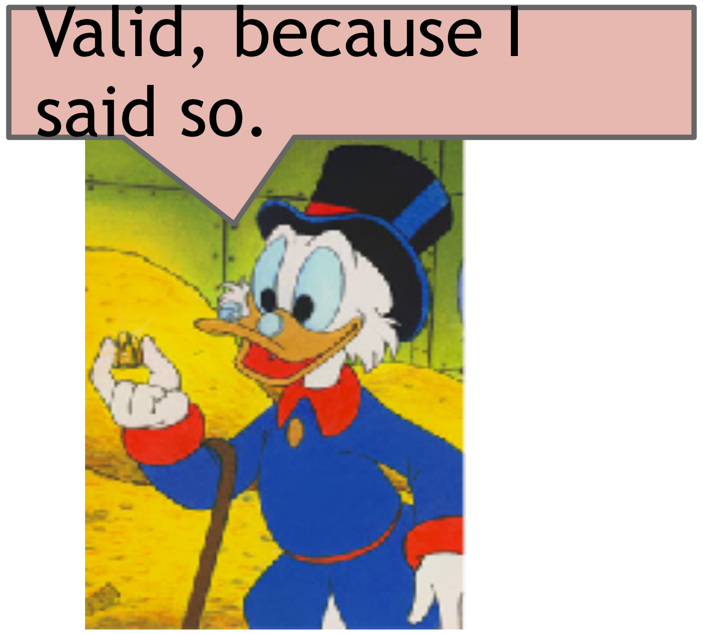
CreateCoins transaction creates new coins

transID: 73 type:CreateCoins

coins created

num	*value*	*recipient*	
0	3.2	0x...	coinID 73(0)
1	1.4	0x...	coinID 73(1)
2	7.1	0x...	coinID 73(2)

signature



PayCoins transaction consumes (and destroys) some coins, and creates new coins of the same total value

transID: 73 type:PayCoins

consumed coinIDs:
68(1), 42(0), 72(3)

coins created

num	*value*	*recipient*
0	3.2	0x...
1	1.4	0x...
2	7.1	0x...

signatures

Valid if:
- -- consumed coins valid,
- -- not already consumed,
- -- total value out = total value in, and
- -- signed by owners of all consumed coins

Distributed consensus

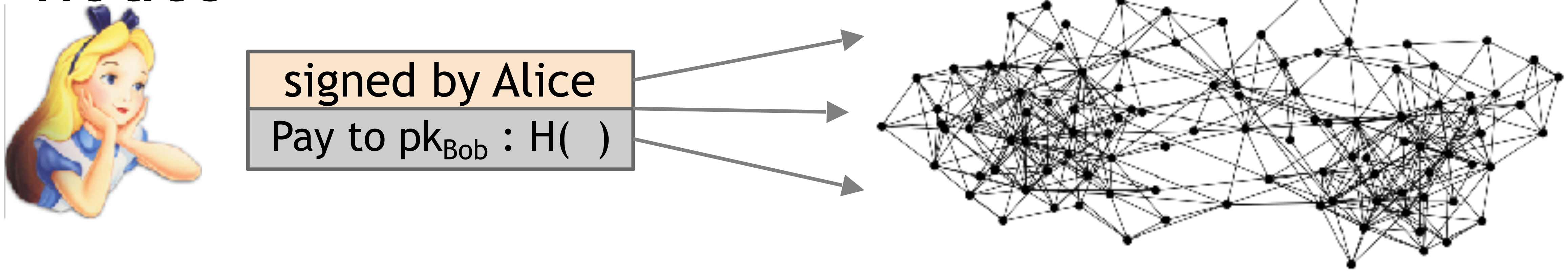
Defining distributed consensus

The protocol terminates and all honest nodes decide on the same value

This value must have been proposed by some honest node

Bitcoin is a peer-to-peer system

When Alice wants to pay Bob:
she <u>broadcasts the transaction</u> to all Bitcoin nodes



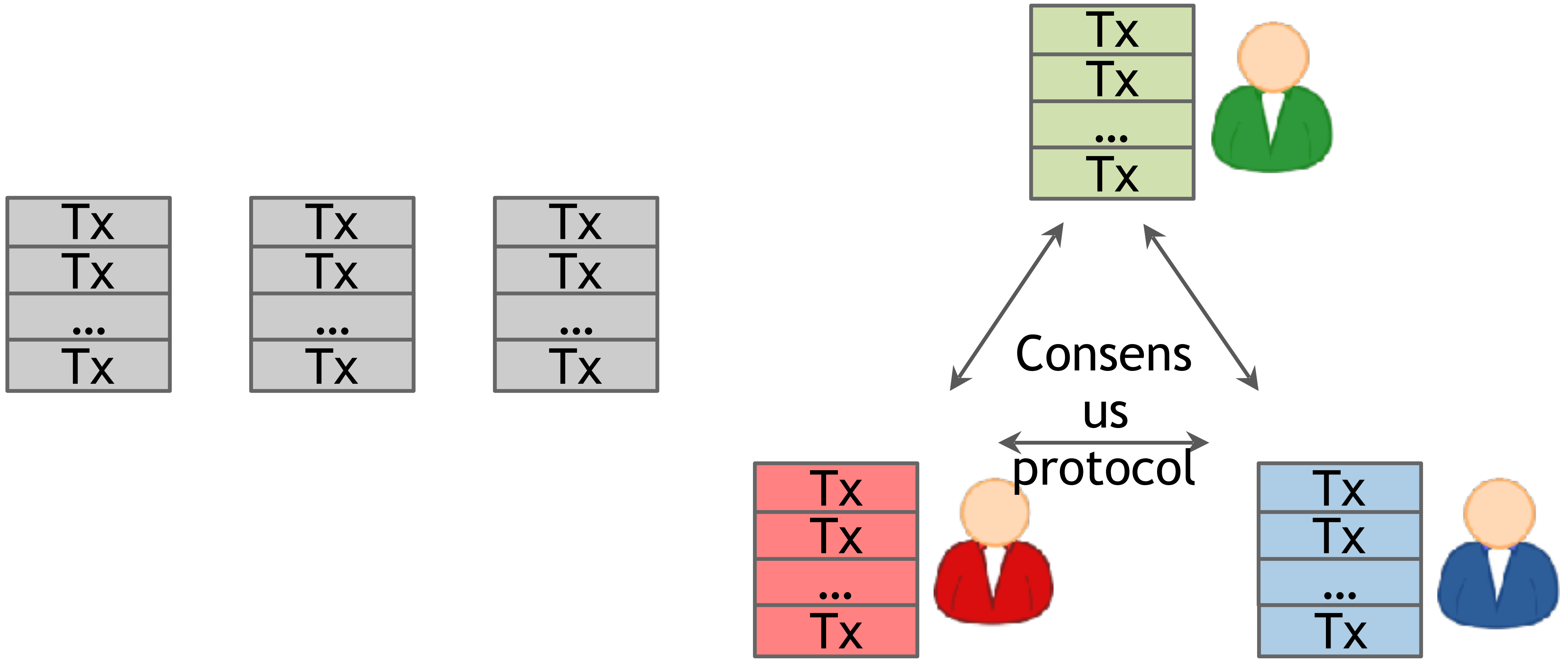
Note: Bob's computer is not in the picture

How consensus could work in Bitcoin

At any given time:

- All nodes have a sequence of blocks of transactions they’ve reached consensus on
- Each node has a set of outstanding transactions it’s heard about

How consensus could work in Bitcoin



OK to select any valid block, even if proposed by only one node

Why consensus is hard

Nodes may crash
Nodes may be malicious

Network is imperfect

- Not all pairs of nodes connected
- Faults in network
- Latency

No notion of global time

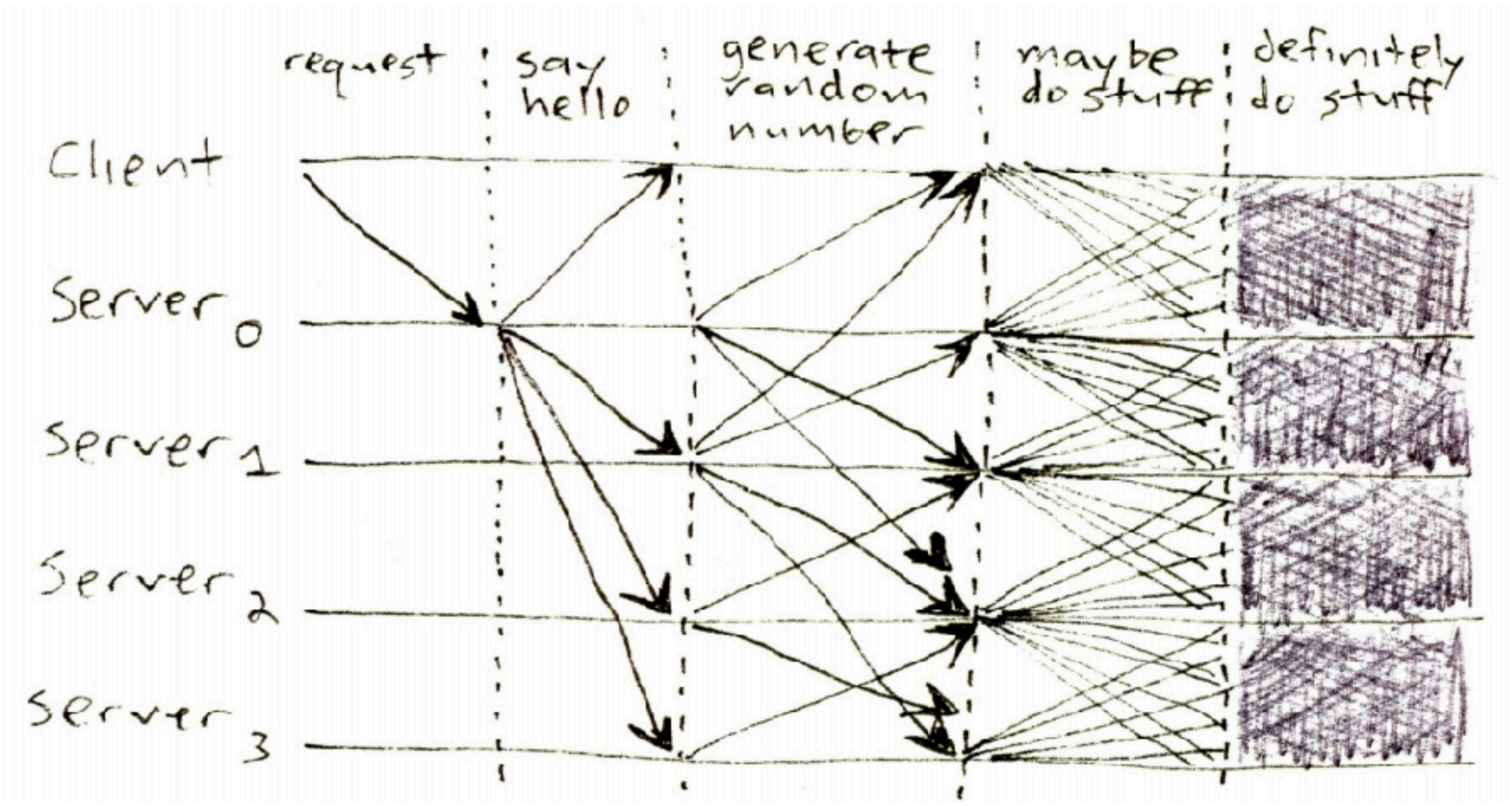
Many impossibility results

- Impossible without 2/3 honest majority [Pease, Shostak, Lamport'80]
- Impossible with a <u>single</u> faulty node, in the fully asynchronous setting, with deterministic nodes [Fischer-Lynch-Paterson'85]

Why do these results matter?

- Because without node identities,
  an attacker could easily crash these networks by impersonating many nodes (“Sybil attack”)

- Because synchronicity is hard


request
say hello
generate random number
maybe do stuff
definitely do stuff
Client
Server 0
Server 1
Server 2
Server 3

Some positive results

Example: Paxos [Lamport]

Never produces inconsistent result, but can (rarely) get stuck

Understanding impossibility results

These results say more about the model than about the problem

The models were developed to study systems like distributed databases

Bitcoin consensus: theory & practice

- Bitcoin consensus: initially, seemed to work better in practice than in theory

- Theory has been steadily catching up to explain why Bitcoin consensus works [e.g., Garay-Kiayias-Leonardos’15,Pass-Shelat-Shi’17,Garay-Kiayias-Leonardos’17,…]

- Theory is important, can help predict unforeseen attacks

Some things Bitcoin does differently

Introduces incentives

- Possible only because it’s a currency!

Embraces randomness

- Does away with the notion of a specific end-point
- Consensus happens over long time scales — about 1 hour

Consensus without identity: the blockchain

Why identity?

Pragmatic: some protocols need node IDs

Security: assume less than 50% malicious

Why don’t Bitcoin nodes have identities?

Identity is hard in a P2P system — Sybil attack

Pseudonymity is a goal of Bitcoin

Weaker assumption: select random node

Analogy: lottery or raffle

When tracking & verifying identities is hard, we give people tokens, tickets, etc.

Now we can pick a random ID & select that node

Key idea: implicit consensus

In each round, random node is picked

This node proposes the next block in the chain

Other nodes implicitly accept/reject this block

- by either extending it
- or ignoring it and extending chain from earlier block

<u>Every block contains hash of the block it extends</u>

Consensus algorithm (simplified)

1. New transactions are broadcast to all nodes
2. Each node collects new transactions into a block
3. In each round a <u>random</u> node gets to broadcast its block
4. Other nodes accept the block only if all transactions in it are valid (unspent, valid signatures)
5. Nodes express their acceptance of the block by including its hash in the next block they create

So how do we pick a random node?

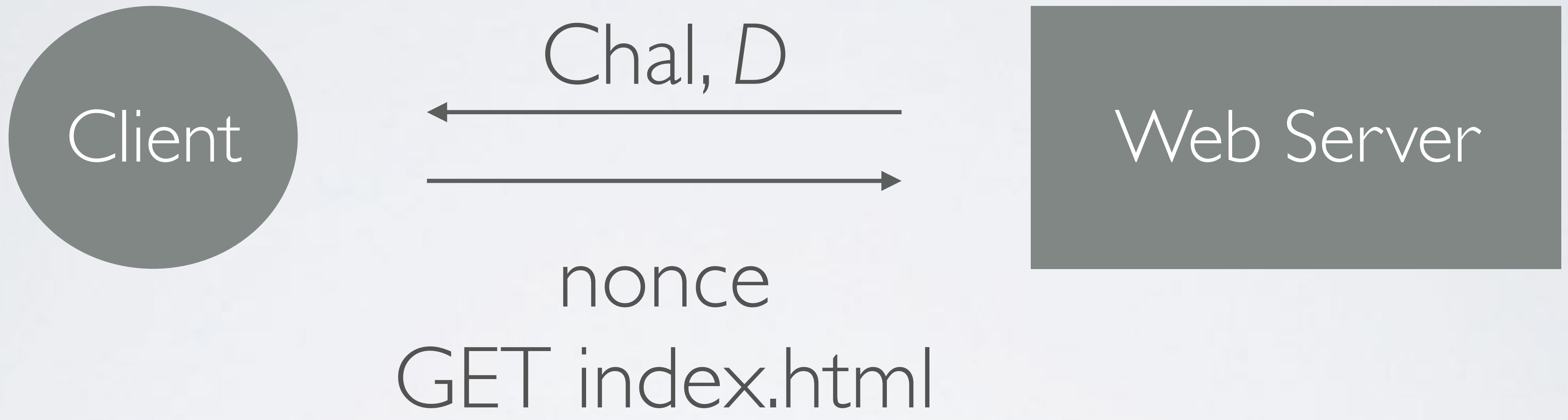
Resources & Consensus

- One computer can easily pretend to be many "nodes", so simple random selection != good
- But an observation: resources (e.g., hardware, storage, CPU, GPU, etc.) are much harder to fake
- Idea: make your probability of winning the vote proportional to your overall resources

Puzzles

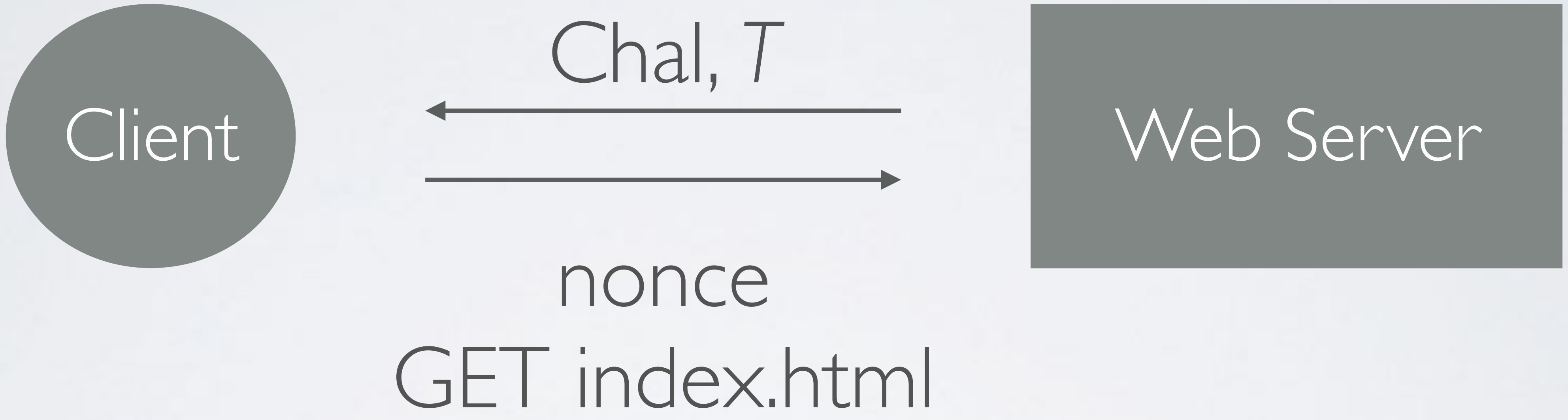
- Early idea, proposed in the 90s: solve computational puzzles to prove CPU power
- Dwork & Naor (1992): use them to make spam email more expensive
- Back (1997): Hashcash, spam emails again
- Juels & Brainard (1999): use them to prevent DoS on web servers

Simple interactive puzzle



The web server accepts the GET iff S = H(Chal | Nonce) begins with "*D*" 0 bits

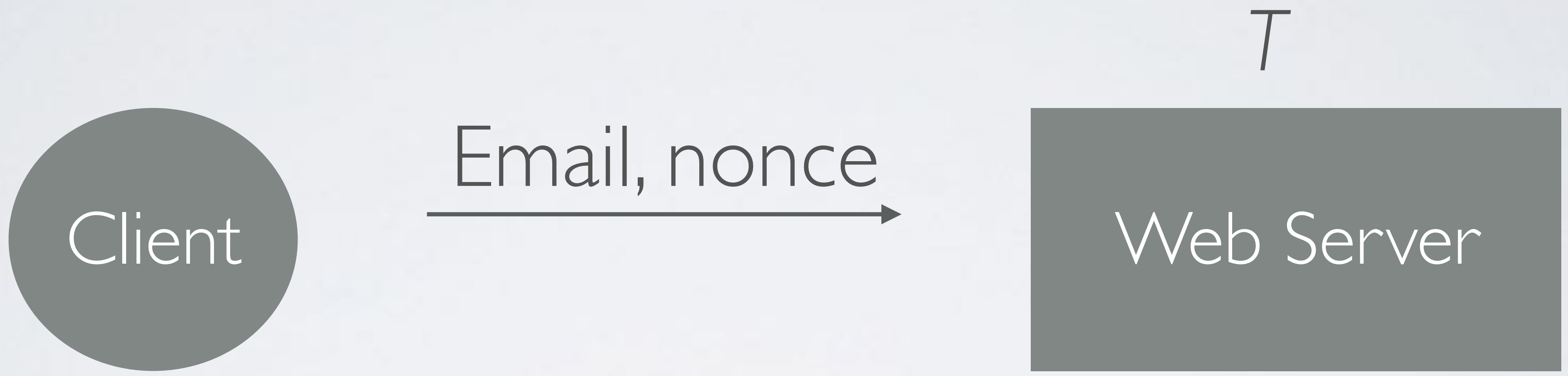
Simple interactive puzzle



Alternative formulation:

The web server accepts iff $H(\text{Chal} \mid \text{nonce}) < T$

Spam/Hashcash



The web server accepts iff H(Email | nonce) < *T*
and the email hasn't been seen before

Bitcoin PoW

The web server accepts iff H(Email | nonce) < *T*
and the email hasn't been seen before

Key idea: implicit consensus

In each round, all nodes compete to solve a puzzle

The winner proposes the next block in the chain
and sends out their solution along with it

Other nodes implicitly accept/reject this block
- by either extending it
- or ignoring it and extending chain from earlier block

Every block contains hash of the block it extends

What's the puzzle?

The puzzle is simply the hash of the new block, which must be chained off the previous block

I.e., find a nonce such that H(block | nonce) < T for some T

The winner proposes the next block in the chain
and sends out their nonce

Other nodes implicitly accept/reject this block
- by either extending it
- or ignoring it and extending chain from earlier block

What happens if nodes ignore the block?

Where do we get *T*?

The value T is called the “block difficulty”. It’s adjustable.

Ideas:

- Choose T once at the start, keep fixed
- Adjust T from time to time

  What are the impacts of these choices?

Where do we get *T*?

The value T is called the “block difficulty target”.
It’s adjustable.

Ideas:

- Choose T once at the start, keep fixed
- Change T from time to time

  What are the impacts of these choices?

Where do we get *T*?

The value T is called the “block difficulty target”. It’s adjustable.

Ideas:

- Choose T once at the start, keep fixed
- Change T from time to time

  What are the impacts of these choices?