Blockchains & Cryptocurrencies

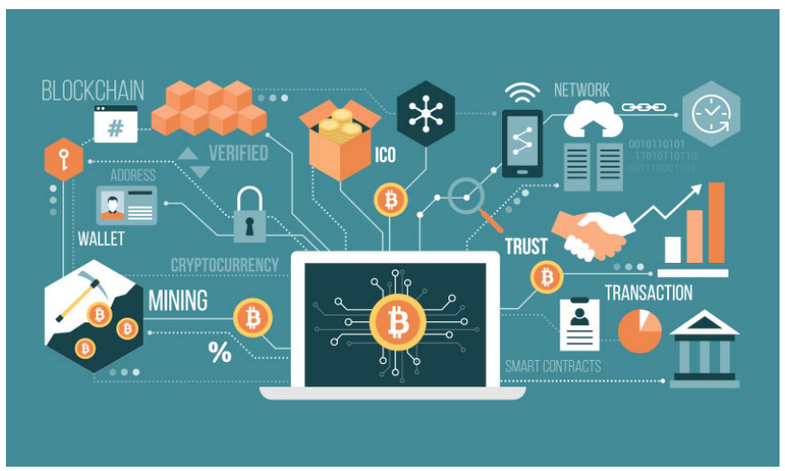
Proofs of Work & Bitcoin Consensus

Instructor: Abhishek Jain
Johns Hopkins University - Spring 2021

*Some slides based on NBFMG

Today

- Brief recap of Distributed Consensus
- Consensus mechanism in Bitcoin
- Proof of Work puzzles



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

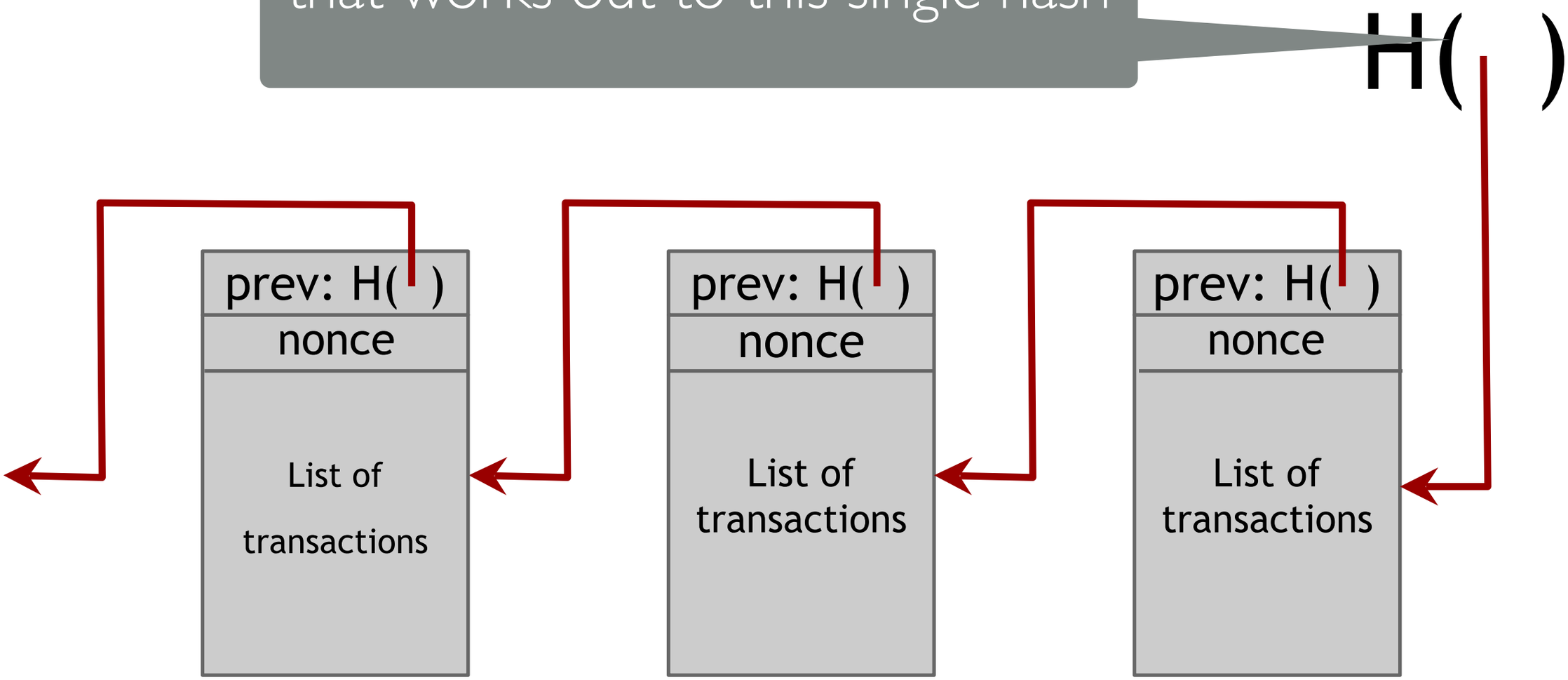
Defining distributed consensus

Q: What is this "value" in Bitcoin?

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

This value must have been proposed by some honest node

A: In Bitcoin, the value we want to agree on is the current state of the ledger. If we use a blockchain, that works out to this single hash

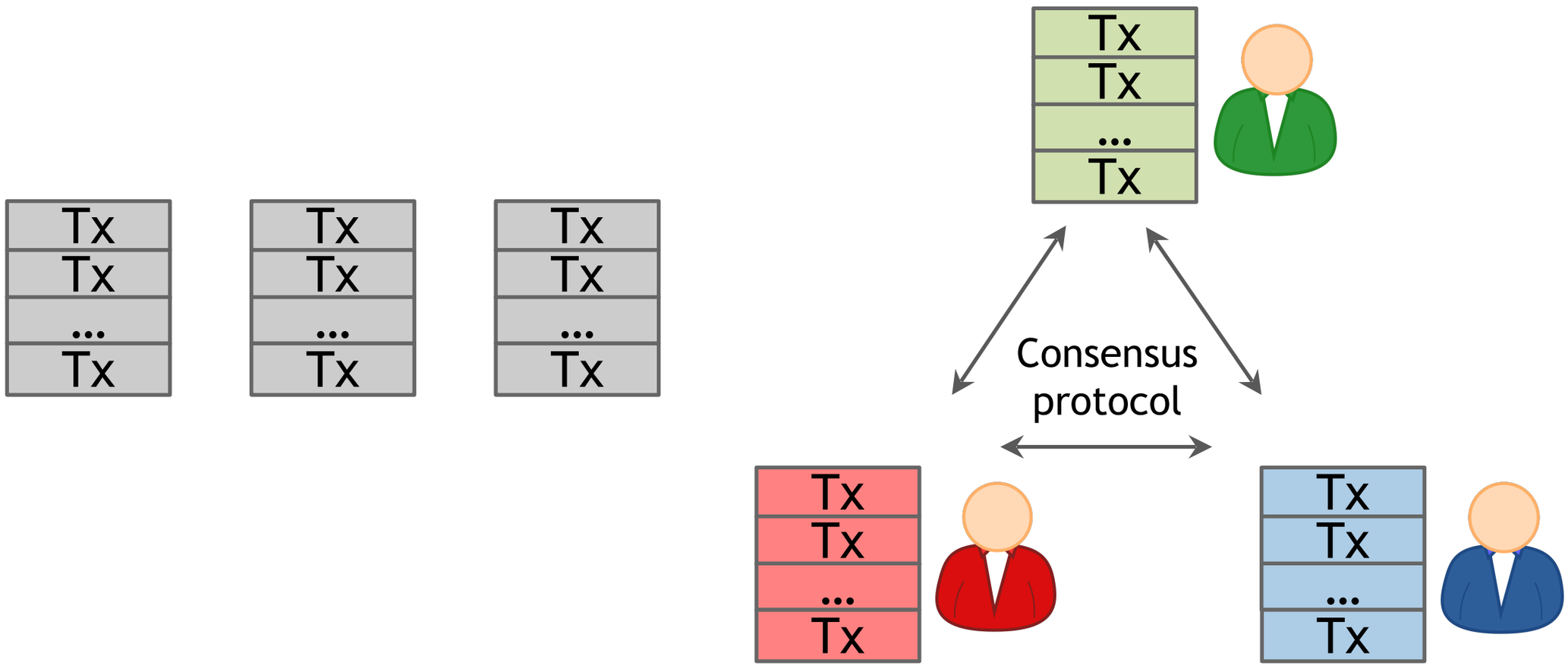


How consensus could work in Bitcoin

At any given time:

- All nodes have a sequence of blocks of transactions they've reached consensus on
- (Blocks are also distributed via p2p network)
- Each node has a set of outstanding transactions it's heard about

How consensus could work in Bitcoin



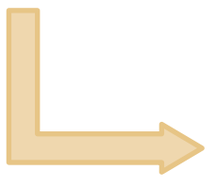
OK to select any valid block, even if proposed by only one node

Why consensus is hard

Nodes may crash
Nodes may be malicious

Network is imperfect

- Not all pairs of nodes connected
- Faults in network ("partitioning")
- Latency



No notion of global time

Many impossibility results

- **Corruption Threshold:** Impossible without 2/3 honest majority [Pease, Shostak, Lamport'80]

- **Asynchronicity and Determinism:** Impossible with a single faulty node, in the *fully asynchronous* setting, with *deterministic* nodes [Fischer-Lynch-Paterson'85]

Why do these results matter?

- Achieving 2/3 honest majority is hard in P2P systems
- Because synchronicity is hard

Understanding impossibility results

These results say more about the model than about the problem

The models were developed to study reliability in systems like distributed databases

Some positive results

Example: Paxos [Lamport]

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

Bitcoin consensus: theory & practice

- Bitcoin consensus mechanism 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,…]

- Most of these results assume partial synchrony, and make some other assumptions about the network to establish proofs.

- Theory is important, can help predict unforeseen attacks

Some things Bitcoin does differently

No node identities

- Uses a different notion of "honest majority"

Introduces incentives

- Possible because it's a currency!
- Gives a sort of "rational-adversary" guarantee against formation of dishonest majority

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?

Some protocols need node IDs

Allows for honest-majority assumptions: assume less than 50% nodes malicious

Why don’t Bitcoin nodes have identities?

Identity is hard in P2P system (Sybil attack)

Pseudonymity is a goal of Bitcoin

Consensus Idea (simplified): select *random* ID

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

Every block contains hash of the block it extends

Summary (so far)

1. New transactions are broadcast to all nodes
2. Each node collects new transactions into a block
3. In each round a random node gets to broadcast its block
4. Other nodes accept the block only if all transactions in it are valid (unspent, valid signatures) and it builds on a chain they already accept
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?

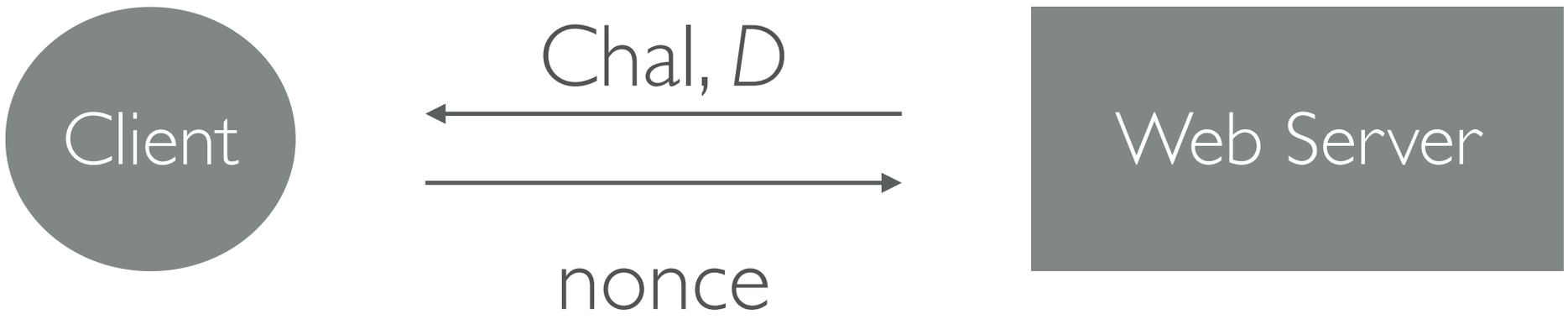
Resources & Consensus

- One computer can easily pretend to be many "nodes", so simple random voting != 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(Chal | nonce) < *T*

Spam/Hashcash

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

Bitcoin consensus (revised)

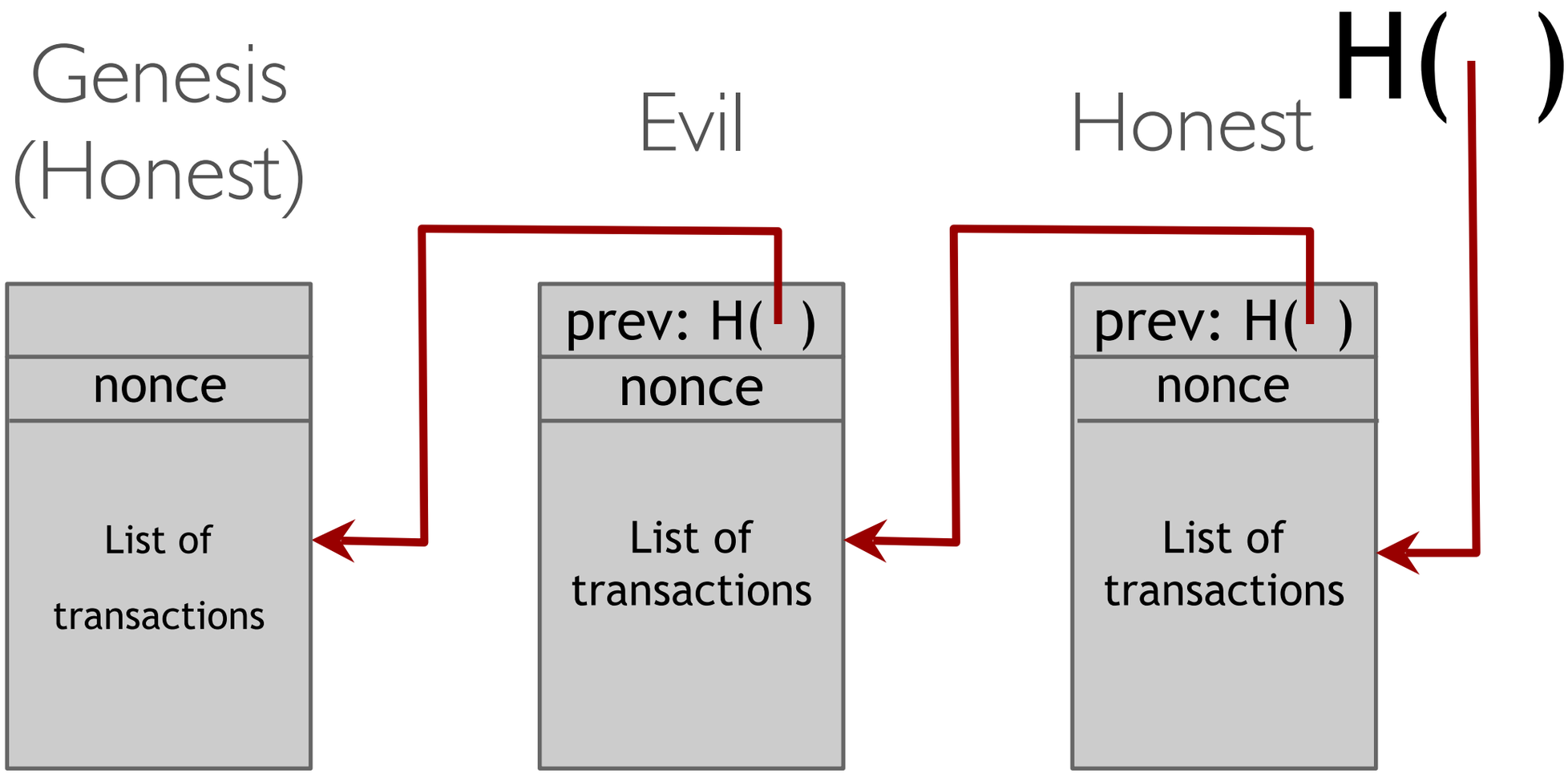
In each round, all nodes compete to solve a puzzle

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


Genesis
(Honest)
Evil
Honest
H( )
prev: H( )
nonce
List of
transactions
prev: H( )
nonce
List of
transactions
prev: H( )
nonce
List of
transactions

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 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*?

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?

Selecting T (Bitcoin)

Bitcoin's difficulty changes every 2016 blocks

Goals:

- Bitcoin block time should <u>average</u> 10 minutes
- Everyone in the network agrees on *T*
- *Hence must be a function of chain data*

(This brings back a notion of time)

Selecting T (Bitcoin)

Every block contains a (packed) encoding of T (target) which corresponds to "difficulty" **(d)**

d = ((2^{16} - 1) * 2^{8*26}) / T : relationship between d, T

Goals:

- Bitcoin block time should <u>average</u> 10 minutes
- 2016 blocks * 10 minutes == 2 weeks
- Everyone in the network agrees on *T*
- Hence must be a function of chain data

Bitcoin blocks include timestamps

Every block contains a timestamp that alleges when it was found

Remember that nodes can be adversarial! So some timestamps may be lies! This requires heuristics:

- Each timestamp must be no sooner than the median of the past 11 blocks
- Honest nodes must reject timestamps that are > 2hrs in the future

What if there's a collision?

Sometimes two separate nodes find a valid solution simultaneously

This can result in network partition

What if there's a collision?

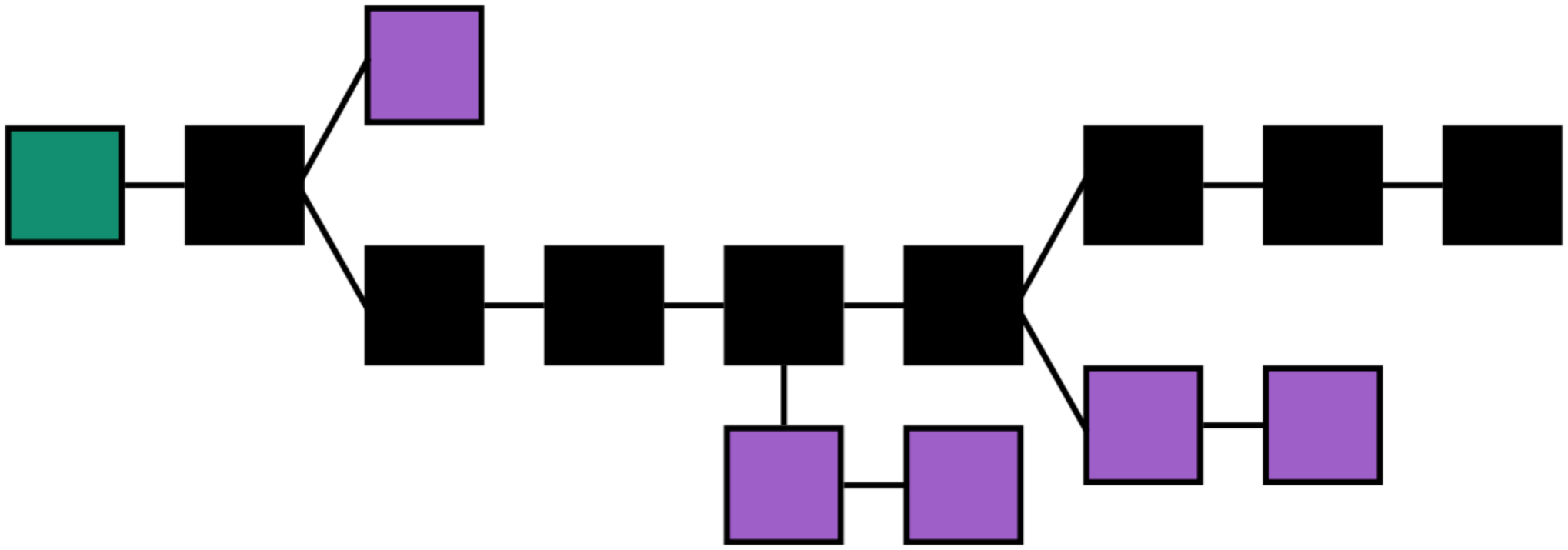


Image CC-BY-3 Theymos taken from the Bitcoin wiki

“Longest chain rule”

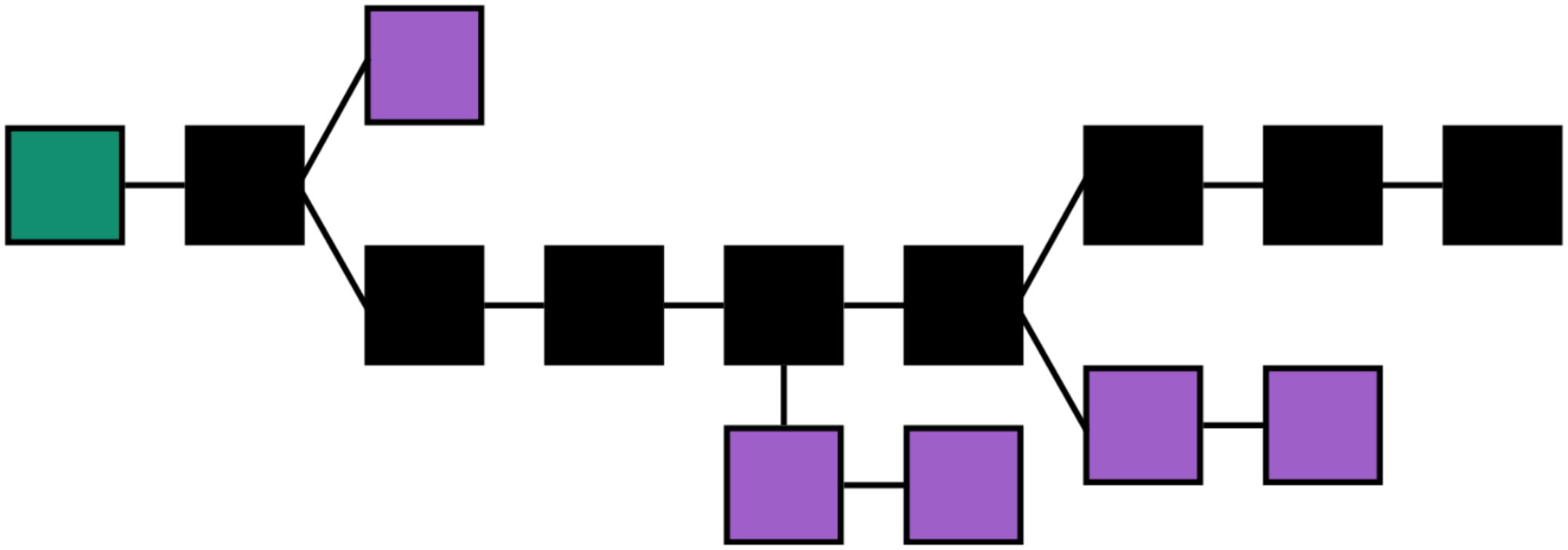


Image CC-BY-3 Theymos taken from the Bitcoin wiki

“Longest chain rule”

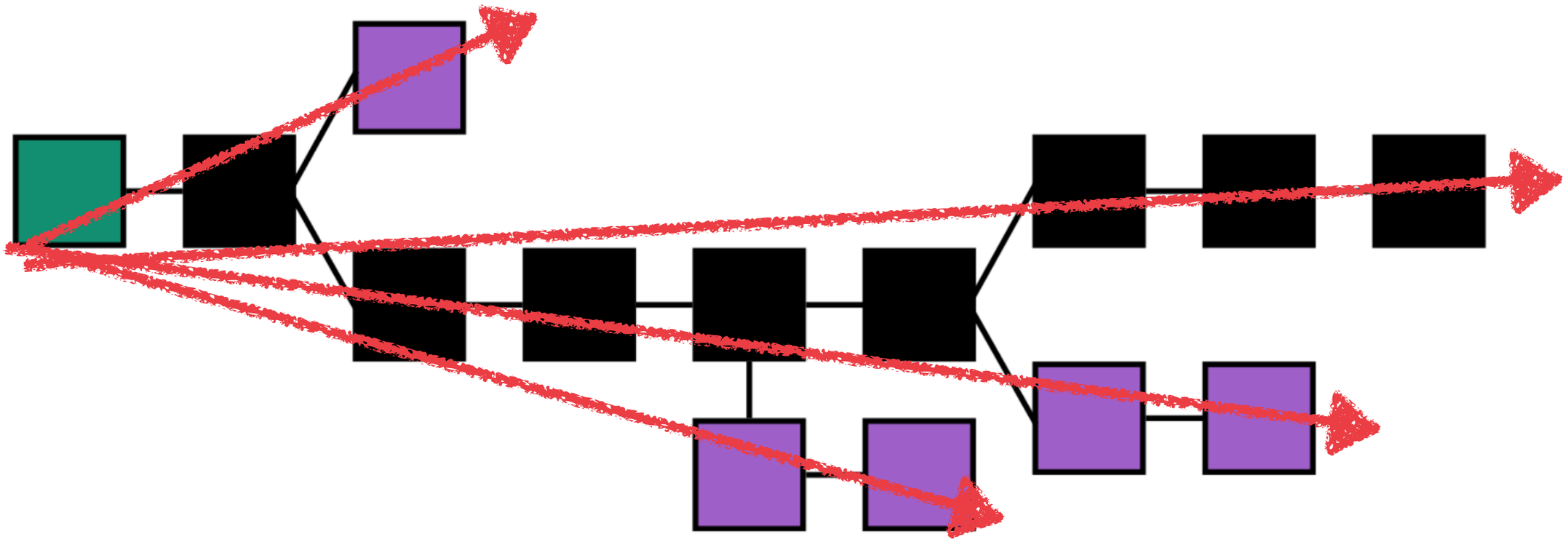


Image CC-BY-3 Theymos taken from the Bitcoin wiki

This is good and bad

Good: if we experience a "chain fork" and the network is connected (I.e., not totally partitioned), then eventually we will learn about both forks

Good: if the "hash power" behind the two chains is unequal, we will probably end up with one chain getting longer

Even if the hash power is equal, the inherent randomness of the puzzle (PoW) will likely cause an advantage

As one chain grows longer, other nodes will adopt it, and start adding to it

What’s the bad?

What's the bad?

When a chain becomes longer than the "current chain" a node thinks is the longest chain, they must abandon that older chain

Finality

"Finality is the assurance or guarantee that cryptocurrency transactions cannot be altered, reversed, or canceled after they are completed."

https://academy.binance.com/glossary/finality

Finality

"Finality is the assurance or guarantee that cryptocurrency transactions cannot be altered, reversed, or canceled after they are completed."

Bitcoin's finality is probabilistic (and computational)

Reorganizations become less probable (and more expensive) over time, but they never become impossible*

Q: What if difficulty changes?

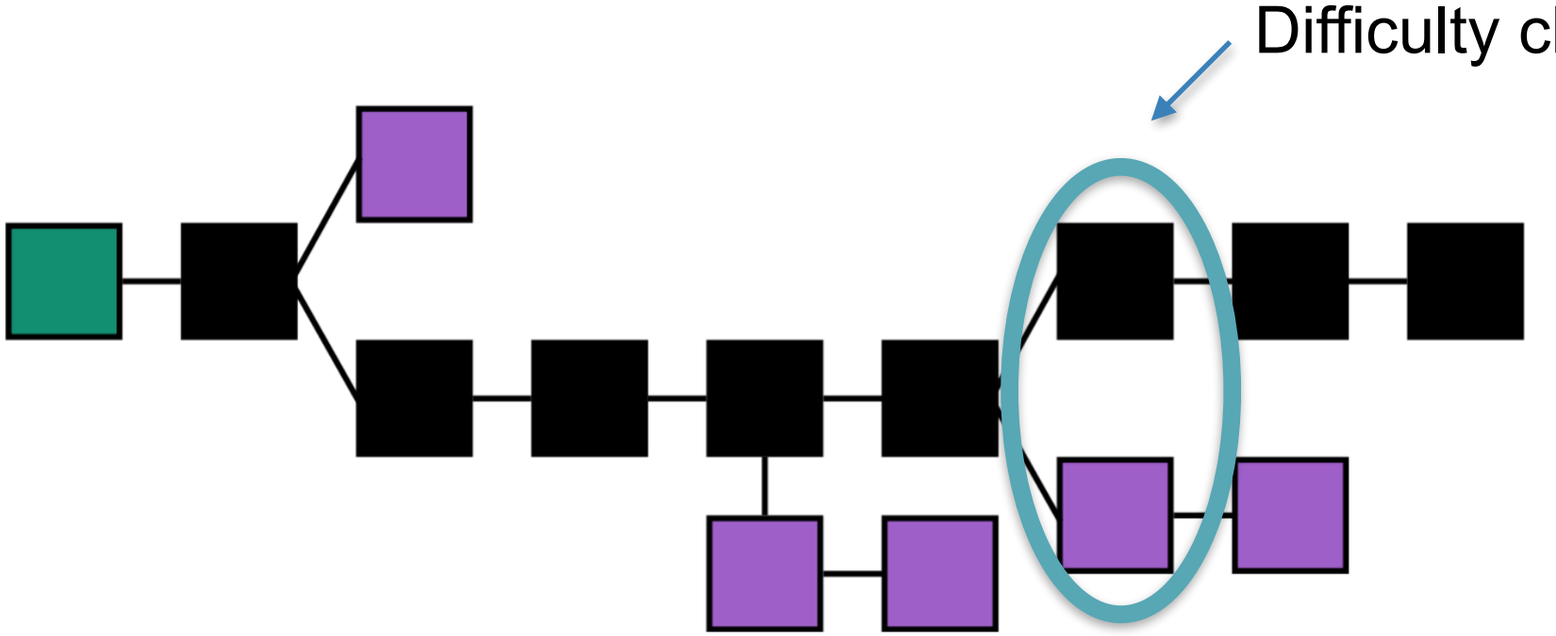


Image CC-BY-3 Theymos taken from the Bitcoin wiki

Q: What if difficulty changes?

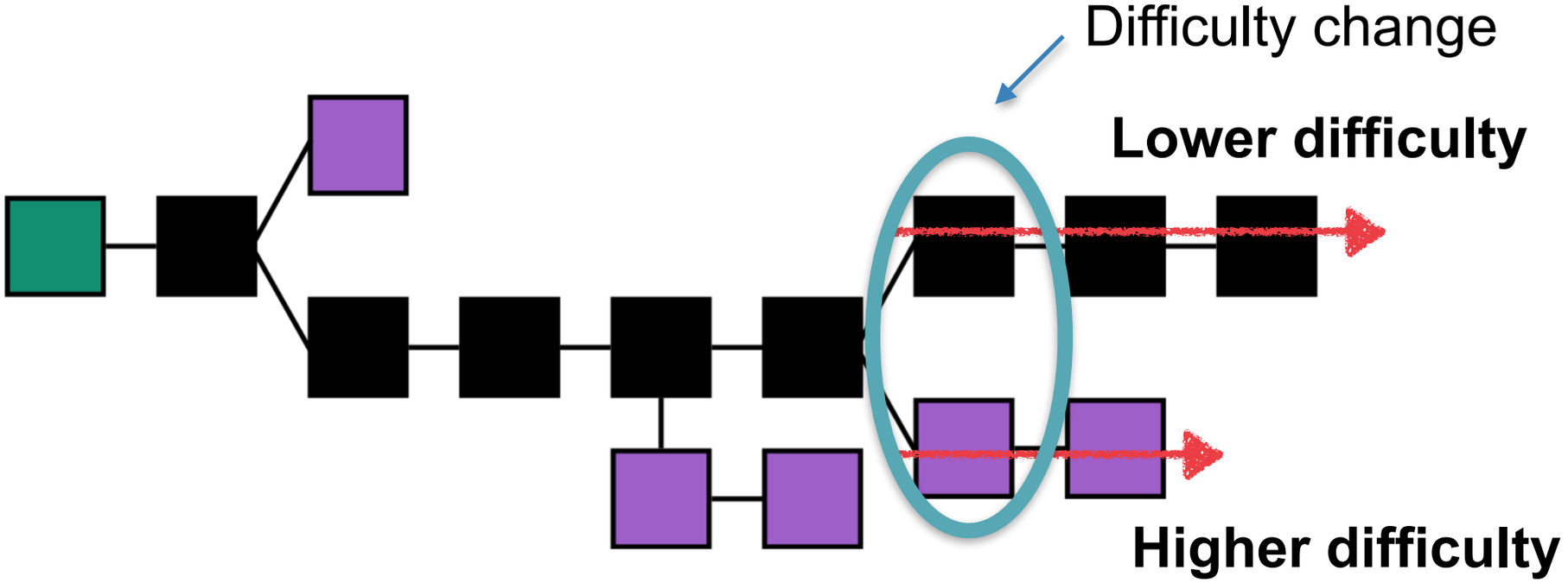


Image CC-BY-3 Theymos taken from the Bitcoin wiki

A: "Chain with most hashwork"

Bitcoin doesn't exactly use the longest chain rule

Instead, it employs a calculation that takes block difficulty into account

Each block has a difficulty. Convert to expected # of hashes to find block. Total these values. Chain with largest total is "longest".

Most of the time, this is equivalent to longest chain

How many blocks can the adversary make?

Consider an adversary that controls a r-fraction of the hash power

How many blocks in expectation can they build in a t-block window?

What is the probability that they dominate that t-block window?

Look at papers (reading list) for detailed analysis

We will see some attacks that leverage these simplified calculations later….

How do we incentivize mining?

How do we incentivize mining?

Two answers to this question in Bitcoin:

1. "Transaction fees" Each transaction has a "tip" that can be collected by the node who mines it into a block (incentivizes inclusion of transactions)

2. "Block reward" 50/25/12.5/… BTC made from scratch (in a special Coinbase transaction) and given to the miner