Lecture 9. Scaling Latency

Module 1: Bitcoin. (lectures 2-7).

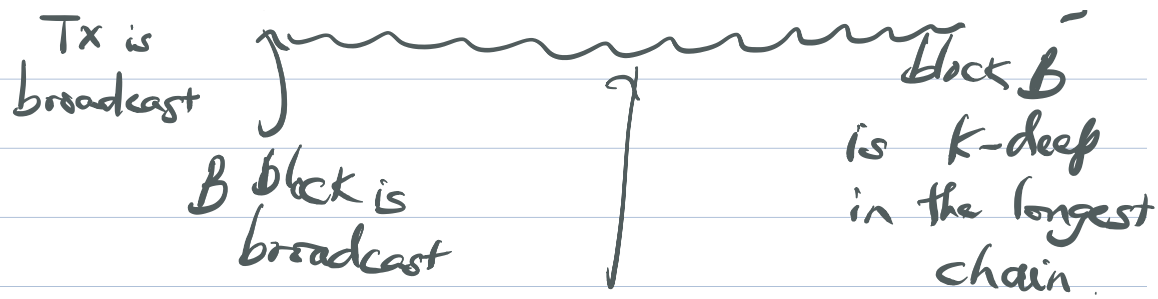
Module 2: Scaling Bitcoin (lectures 8 - 15)

- improving performance while still retaining longest chain protocol
  - throughput (#8)
  - latency (#9)
  - storage & compute (#11)
  - trust taken out externally (#10)
  - energy #(12)
- more or less the software stack is unchanged.

(1) Bitcoin latency.

↓

time from when a tx was broadcast until the tx is confirmed in the ledger



- real bottleneck
- depends on how large k is.

$$\text{Latency} \approx k \cdot \frac{1}{\lambda}.$$

↑ rate of blocks.

But due to forking, not all blocks may be on a chain.

$$\text{Latency} \approx k \cdot \frac{1}{\frac{(1-\beta)\lambda}{1+(1-\beta)\lambda\Delta}}$$

↑ network delay (Δ).

$$\Delta = D + \frac{B}{C}$$

B ← block size

C ← capacity.

speed of light

Propagation delay | processing delay

$$\text{Latency} \approx k \cdot \Delta \cdot \frac{1}{\frac{(1-\beta)\lambda\Delta}{1+(1-\beta)\lambda\Delta}}.$$

for low forking: $\lambda\Delta \ll 1$.

how about k?

From lecture 6; $P_e \approx e^{-ck}$

$$k \approx \frac{1}{c} \log\left(\frac{1}{P_e}\right).$$

↑ c: Constant depends on β.

$$\text{Latency} \approx \frac{1}{c} \cdot \underbrace{\log\left(\frac{1}{P_e}\right)}_{\text{security}} \cdot \frac{1}{(1-\beta)\lambda}.$$

$$\lambda \ll \frac{1}{\Delta}.$$

Bitcoin: $\frac{1}{\lambda} \simeq 10$ minutes

$k = 200$

then latency $= 16$ hours.

Only way to improve latency is

* reduce k ; but this reduces security

* increase λ ; but this reduces security.

Ethereum: $\frac{1}{\lambda} = 15s$

$k = 100$

latency $= 25$ minutes.

Way better than Bitcoin performance.

improvement simply by picking better

parameters.

Question: Can one make relatively small changes to the longest chain protocol & PoW mining while scaling latency?

⇒ do not want latency to depend on security level.

→ decouple security from latency.

Hybrid Consensus.

BFT Consensus algorithms provide fast Consensus among a fixed set of participating nodes.

1982 ; ← began

40 years later: Very good state of the art.

We will cover 2 such protocols in

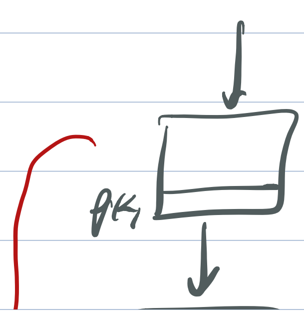
Module 3.

For now blackbox BFT Consensus

Q: How to bring this blackbox to Bitcoin to speed up latency?

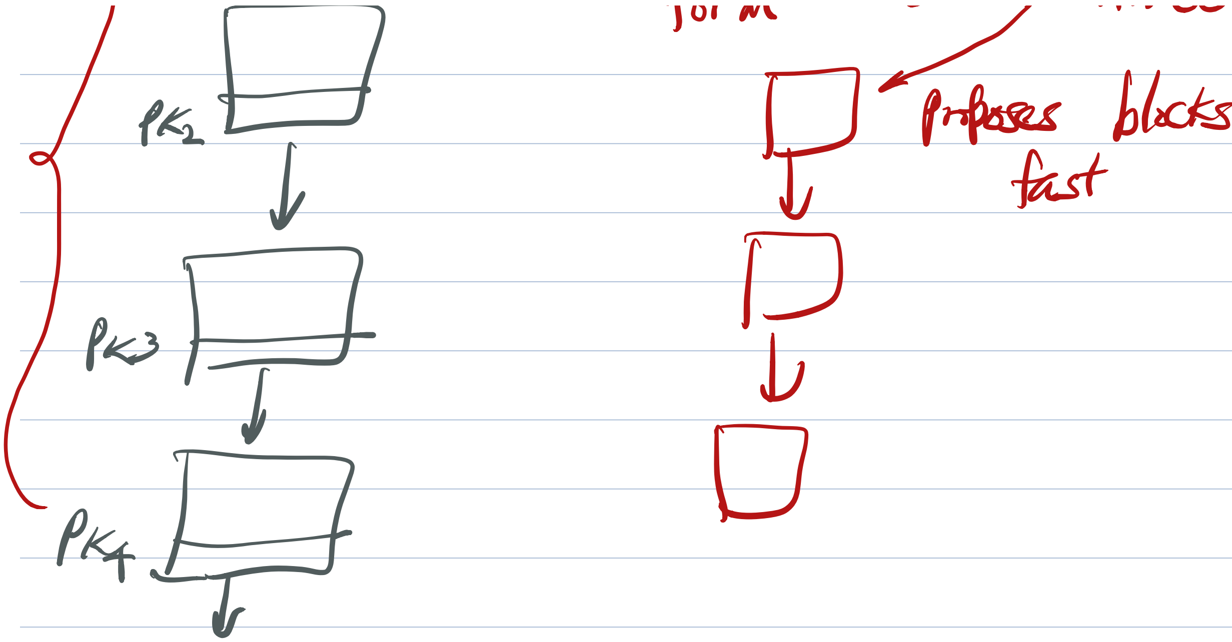
<u>challenge</u>: to use the blackbox, one needs to have a <u>fixed</u> set of participants.

<u>Idea</u>: elect the BFT participants (Committee) from the longest chain itself.



(pk_1, pk_2, pk_3, pk_4)

form a committee.



Slow longest chain protocol.

- The committe evolves over time as the longest chain progresses.
- This keeps <u>refreshing</u> the committe for BFT protocol.
- Vulnerable to an <u>adaptive</u> adversary

Can pick which miner is adversarial *dynamically* as a function of blockchain state

- In practice: bribing attack. works because miners can have very small hash power but got lucky to be a proposer.

Prism

↓

optimal latency.

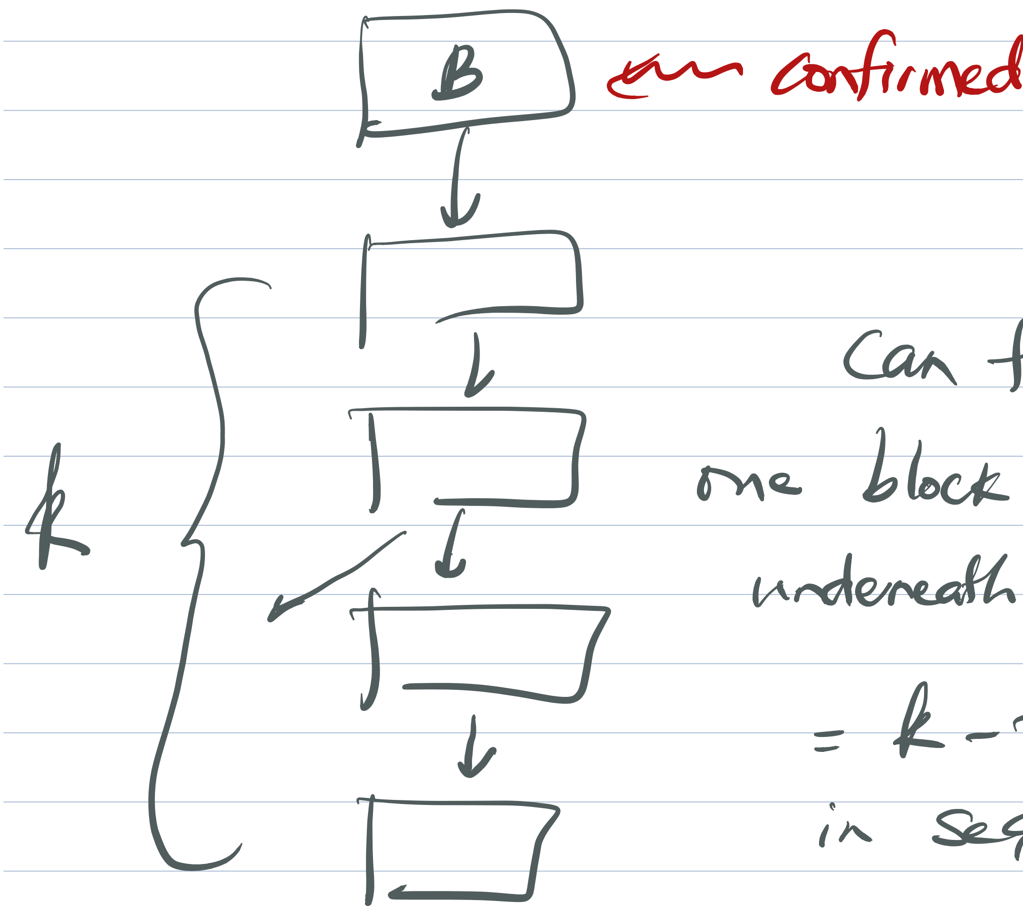
- Prism 1.0 which scaled throughput; last lecture.

Decoupling Principle:

Security

Voting

k-deep confirmation rule is a form of voting.

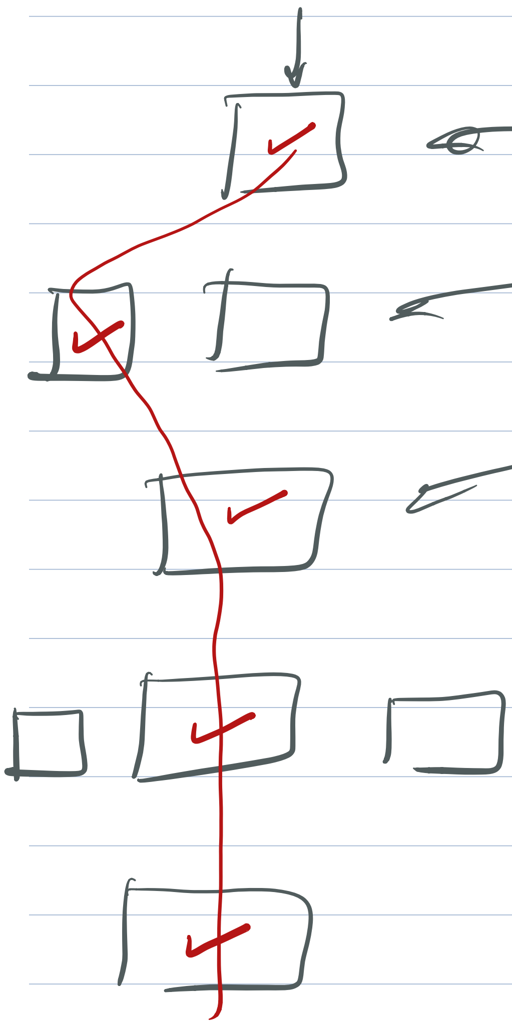


Can think of one block = 1 vote underneath B.

= k-votes in sequence

really need k large ~ sample the miners.

Idea: vote in parallel.

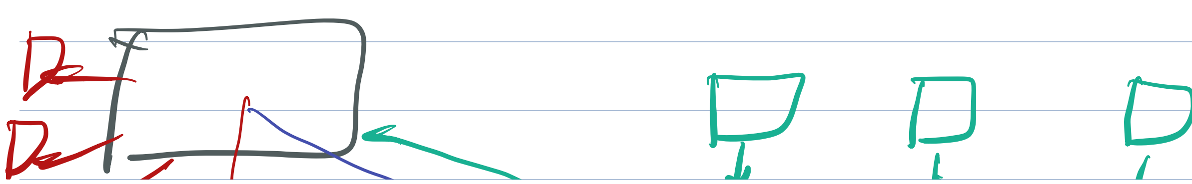


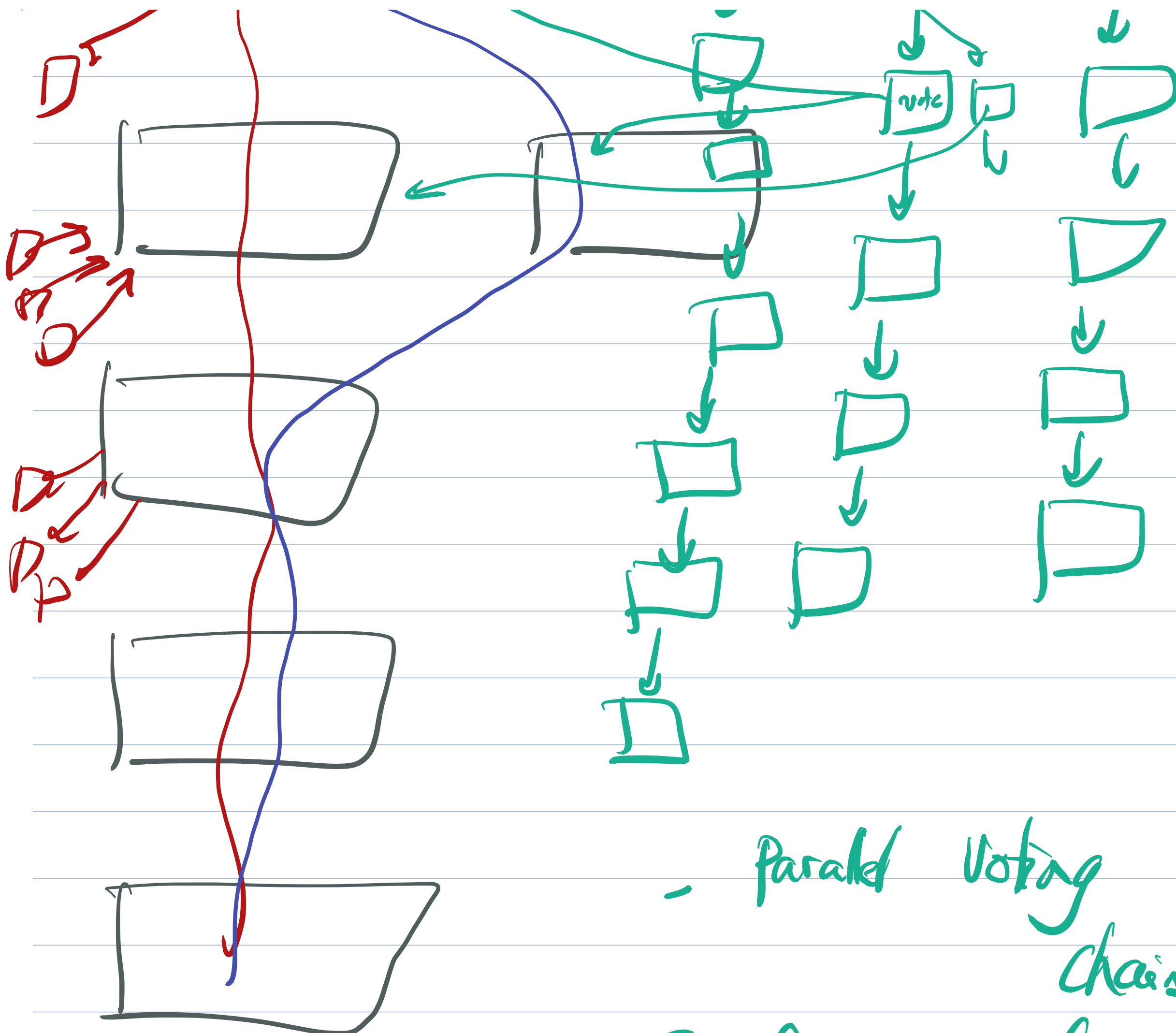
* votes come in parallel??

* and elect a proposal block per level.

* how many votes are there?

Prism: many parallel Voting chains





- Parallel voting chains
- fixed # of voting chains

Proposal rule

Voting rule

Mining rule:

e.g. 1000.

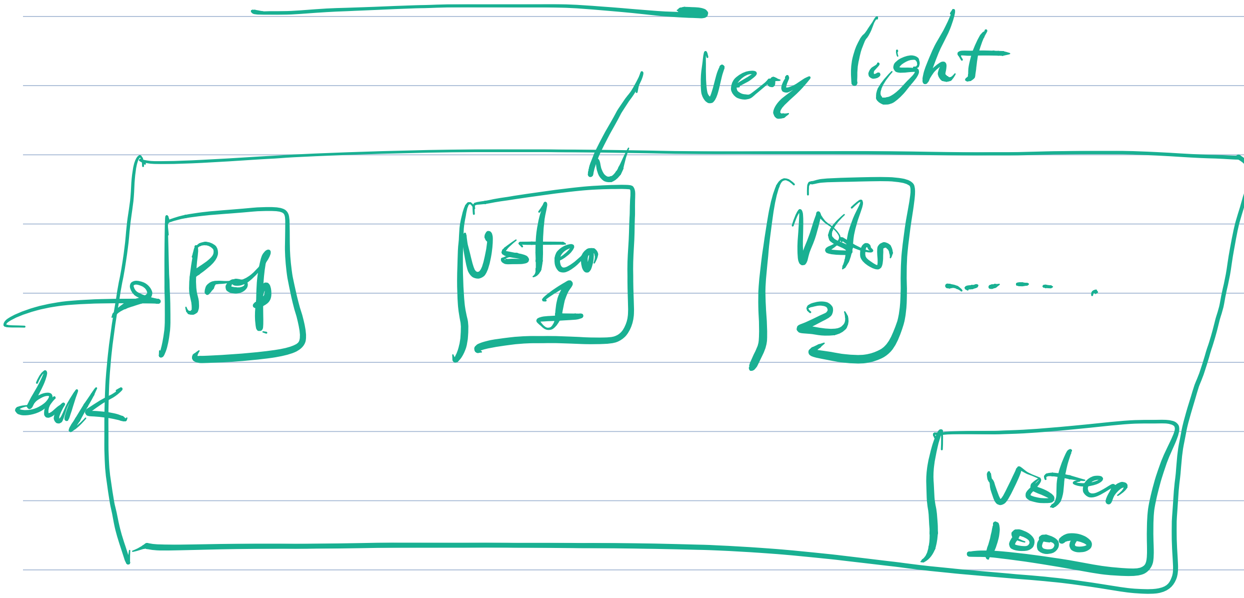
honest miner picks to be prop or voting block at random

But how do you prevent adv. from congregating?

1001

<u>Sortition</u> or many-to-1 mining.

<u>Superblock:</u>



mine superblock

hash(superblock) decides

which role the superblock plays.