Lecture 9  Scaling latency

Module 1: Bitcoin (lectures 2-7)

Module 2: Scaling Bitcoin (lectures 8-15)

- improving performance, throughput (#8)
- while still retaining longest chain protocol, storage & compute (#8)
- more or less the software stack is uncharged, energy (#12)

(1) Bitcoin latency.

\[ \text{time from when a tx was broadcast until the tx is confirmed in the ledger} \]

\[ \text{Tx is put into a block B then mined} \]
Tx is broadcast

Block $b$ is broadcast

is $k$-deep in the longest chain

-real bottleneck

-depends on how large $k$ is

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

Traffic of blocks

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

Latency $\approx k \cdot \frac{1}{(\lambda + \beta)^{-\frac{1}{2}}}$

network delay

$\Delta = D + \frac{B \times \text{block size}}{C \times \text{capacity}}$

\(D\), \(C\)
Latency \approx k \cdot \Delta \cdot \frac{1}{(1 + f) 2\Delta} \cdot \frac{1}{1 + (1 - \beta) 2\Delta}

for low forking: 2\Delta \ll 1.

how about \( k \)?

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

\[ k \approx \frac{1}{c} \cdot \log \left( \frac{1}{P_e} \right) \]

\( k \) constant depends on \( \beta \).

Latency \approx \frac{1}{c} \cdot \frac{\log \left( \frac{1}{P_e} \right)}{2\Delta} \cdot \frac{1}{(1 - \beta) 2\Delta} \text{ (security)}
\[ z < \frac{1}{\lambda} \].

\[
\begin{aligned}
\text{Bitcoin:} & \quad \frac{1}{\lambda} \approx 10 \text{ minutes} \\
& \quad k = 200 \\
& \text{then latency} = 16 \text{ hours.}
\end{aligned}
\]

Only way to improve latency is

* reduce \( k \); but this reduces security

* increase \( \lambda \); but this reduces security.

\[
\begin{aligned}
\text{Ethereum:} & \quad \lambda = 15s \\
& \quad k = 100 \\
& \text{latency} = 25 \text{ minutes.}
\end{aligned}
\]

Way better than Bitcoin performance.
Question: Can one make relatively small changes to the longest chain protocol if 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.
Module 3

For now blackbox BFT consensus.

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

Challenge: to use the blackbox one needs to have a fixed set of participants.

Idea: elect the BFT participants (Committee) from the longest chain itself.

\[ (PK_1, PK_2, PK_3, PK_4) \]

for a committee.
The committee evolves over time as the longest chain progresses. This keeps refreshing the Committee for BFT protocol. - Vulnerable to an adaptive 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 1.0 which scaled throughput; last lecture.

Optimal latency.

Decoupling Principle:

Security Voting
$k$-deep confirmation rule is a form of voting.

Can think of one block = 1 vote beneath B.

= $k$-votes in sequence.

Really need $k$ large - sample the miners.
Idea: vote in parallel.

* votes from in parallel??

and elect a proposal block per level.

* how many votes are there?

Prism: voting chains

many parallel
This document contains a diagram and handwritten text. The text includes:

- **Proposal rule**: e.g., 1000
- **Mining rule**: honest miner picks to be part of voting block at random
- **Fixed # of voting chains**: e.g., 1000
- **Parallel voting chains**

The diagram illustrates the flow of various processes, including decision-making and flowchart-like structures.
But how do you prevent adu. from congregating? Sortition or many-to-1 mining.

Superblock:

- Prop
- Voter 1
- Voter 2
- ...

Very light

Hash (superblock) decides
which role the superblock plays.