Permissionlessness:

- Byzantine parties < 50%.
- Fraction of Byzantine computation power < 50%.

In expectation, next block is produced in 10 mins

\[ \rightarrow 20 \text{ CPUs} \rightarrow 10 \text{ mins} \]
\[ 40 \text{ CPUs} \rightarrow 5 \text{ mins} \]

\[ \approx 2 \times 7 \times 24 \times 6 \]

\[ \frac{0.000001}{a+1} \]

Dynamic availability:

- 1 block in expectation every 10 mins \( \leftarrow \) 12 secs; 2 sec
- The time to propagate a block is \( \approx 10 \) secs.

\( \triangle \)

Q: Forking rate observed in Bitcoin vs. Ethereum.
Network assumption:

- Synchrony: any message sent by a party will arrive within some bounded delay $\Delta$.
- Economic incentives: Block rewards: $6.25$ BTC $\Rightarrow$ $11K \approx 120K$
- Transaction fees:
- Network connectivity: Peer-to-peer network.

Communication complexity: every party is connected to every other party.

Adding block: $O(n^2)$

Committing a block: $O(n^2 - k)$

Latency of a commit: $k$ blocks ($k$ confirmations)

$n$ parties, $t$ are corrupt.

Lamport, Shostak, Pease 1980: Byzantine General Problem.
Broadcast (BB)

Commander

(Attack, retreat)

Agreement: No two honest generals take different actions.

Termination: Eventually every honest general to attack or to retreat.

Validity: If commander is honest, then all honest generals follow commander's input.

Byzantine agreement: Every party has an input $v_i$.

Agreement: Same as BB.

Termination: Same as BB.

Validity: if all honest parties have input $v_i = v$, then all honest parties output $v$.

Bitcoin: general changes.

Validity:

<table>
<thead>
<tr>
<th>BB</th>
<th>Bitcoin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only parties interested in convincing an external source.</td>
<td></td>
</tr>
</tbody>
</table>

the result:
- Validity:
- Single-shot:

State machine replication:
Multiple server replicas (faults), some of which may be faulty, and they provide you with an interface of a single non-faulty server.

Ideal world:

Single non-faulty server.

txn, txn2, ...

clients

SMR

txn, txn2

clients

Safety: No two non-faulty server replicas output different values at the same position of the log.

Liveness: Non-faulty server replicas keep committing new client transactions (valid).

**Intention:** If at any point, a party receives a value \( v \), then that party will share it with everyone.

If I know something in round 9, all other parties will know it in \( r+1 \).

**Round 1:** Commander (sender) sends some value \( v \) to all parties.

**Round 2:** If I receive a value \( v \) in round 1, send that value to all other parties.

**Commit:** If I receive exactly one value \( v' \), then output \( v' \).

*O.w.* output 1.

**Attack 1:** Commander does not send anything.
**Attack 2**: Commander sends 2 different values

\[ C \]

**Round 1**:

\[ \begin{array}{c}
\downarrow & \downarrow & \downarrow \\
\uparrow & \uparrow & \uparrow \\
M & M & M \\
\end{array} \]

**Round 2**:

\[ \begin{array}{c}
\downarrow & \downarrow & \downarrow \\
\uparrow & \uparrow & \uparrow \\
M & M & M \\
\end{array} \]

All honest see \( v \) & \( v' \); output \( \bot \).

**Attack 3**: Commander sends \( v \) to half the parties.

**Attack 3'**: Commander sends nothing to honest parties; sends value \( v \) to only malicious parties.

\[ \begin{array}{c}
\downarrow & \downarrow & \downarrow \\
\uparrow & \uparrow & \uparrow \\
M & M & M \\
\end{array} \]