Bitcoin and Nakamoto Consensus
Bank functionality

**State**: account balances
- Alice: $100
- Bob: $200
- Charlie: $50

**Events**: transactions
- Alice pays Bob $20
- Charlie pays Bob $50
- Charlie pays Alice $50
Bank with consensus

Alice

Bob

Ldr

Pay Bob $20
TX succeeded
Check account

Log replication

RAFT cluster
Consensus Issues

Raft is designed for *small* clusters with *mostly fixed* membership running in a *data center*
- Leader election doesn’t scale to large clusters (split vote)
- Membership change complicated, requires two phases
- Timeouts based around data center communication speeds (broadcast << election timeout << MTBF)

*How do we implement a decentralized bank?*
Proof of work

Puzzle based on a function $H()$ that is easy to evaluate but hard to invert

- I.e., given $y$, hard to find $x$ such that $H(x) = y$
- Only viable method is trial and error: compute $H(x_1)$, $H(x_2)$, $H(x_3)$, ...

Puzzle: find $x$ such that $H(x)$ starts with $k$ 0s

- $H(x) = 000000000...$
- For a random $x$, true with probability $2^{-k}$

Can make puzzle dependent on some data $D$

- Find $x$ such that $H(D | x) = 0^k ...$
Block

Block B1

Log entries
...
...
Puzzle solution

H(B1) = H(log entries || solution) = 0^k ...

Block B2

H(B1)

Log entries
...
...
Puzzle solution
Chaining

Each line’s puzzle depends on the previous one

- $L_n \rightarrow L_{n-1} \rightarrow \ldots \rightarrow L_1 \rightarrow L_0$
- To add $m$ blocks, must solve $m$ puzzles

Longest chain wins
Chain evolution
How fast does the chain grow?

Each person expects to solve puzzle/generate new line in time $t$

Among the $r$ processes, log grows at the speed of $t/r$ per line
  ◦ Why?

As more people participate
  ◦ $r$ grows
  ◦ Log grows faster
  ◦ More difficult to revise history!
Incentives for Logging

Security better if more people participated in logging

Incentivize users to log others’ transactions

- Transaction fees: pay me x% to log your data
- Mining reward: each block creates bitcoins
  - Replace “Alice minted x” entries with “Alice logged line L_n”

Payment protocol:

- Alice->Bob: here’s coin x
- Broadcast to everyone: Alice transfers x to Bob
- Bob: wait until transfer appears in a new log line
  - Optionally wait until a few more lines follow it
Putting it all together

Alice generated 50 BTC
Nonce: 1234

Bob generated 50 BTC
Nonce: 5678

Carol generated 50 BTC
Alice transferred 10 BTC to Bob + 1 BTC to Carol (fee)
Nonce: 9932

<table>
<thead>
<tr>
<th>Account</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>39 BTC</td>
</tr>
<tr>
<td>Bob</td>
<td>60 BTC</td>
</tr>
<tr>
<td>Carol</td>
<td>51 BTC</td>
</tr>
</tbody>
</table>
Logging Speed

How to set k?

- Too short: wasted effort due to broadcast delays & chain splits
- Too long: slows down transactions

Periodically adjust difficulty k such that one line gets added every 10 minutes

- Determined algorithmically based on timestamps of previous log entries

Current difficulty

- $26,065,690,694,469,441,028,096$ hashes to win ($\approx 2^{74.5}$)
Broadcast

All-to-all broadcast
- Every transaction (for logging)
- Every block (for chain growth)

How do you implement this?
- DHT (e.g., Chord)
- Gossip
Bandwidth

Data volume
- VISA network: 2000 tps
- Transaction: 0.5 – 1KB
- A single block (10 mins): 1.14 GB
- Total volume ~160 GB / day
  - Or twice that if you include transaction broadcasts

Bandwidth per node?
- On average, each node downloads / uploads each block once
- ~160 GB/day = 15 Mbps
- (only ~$50/month at EC2 prices!)
- Storage & CPU costs dominate