Mechanising Blockchain Consensus

George Pîrlea and Ilya Sergey
Context

- Hundreds of deployed public blockchains

- $600 625 675 735 755 780 820 billion total market cap

  (7 day progression since Jan 1st)
This work

- Formalised a blockchain consensus protocol in Coq

- Proved eventual consistency in a clique topology
Motivation

1. Understand blockchain consensus
   - what it is
   - how it works: example
   - why it works: our formalisation

2. Lay foundation for verified practical implementation
   - verified Byzantine-tolerant consensus layer
   - platform for verified smart contracts

Future work
What it does
• transforms a set of transactions into a globally-agreed sequence

• “distributed timestamp server” (Nakamoto2008)
\[ \{tx_1, tx_3, tx_5, tx_4, tx_2\} \]

\[ [tx_5, tx_3] \rightarrow [tx_4] \rightarrow [tx_1, tx_2] \]

\[ tx_5 \rightarrow tx_3 \rightarrow tx_4 \rightarrow tx_1 \rightarrow tx_2 \]
\{tx_1, tx_3, tx_5, tx_4, tx_2\}

\[
[tx_5, tx_3] \leftarrow [tx_4] \leftarrow [tx_1, tx_2]
\]

\[
tx_5 \rightarrow tx_3 \rightarrow tx_4 \rightarrow tx_1 \rightarrow tx_2
\]
\{tx_1, tx_3, tx_5, tx_4, tx_2\}

\[
\[
\emptyset \leftarrow [tx_5, tx_3] \leftarrow [tx_4] \leftarrow [tx_1, tx_2]
\]

GB = genesis block

tx_5 \rightarrow tx_3 \rightarrow tx_4 \rightarrow tx_1 \rightarrow tx_2
How it works
• **distributed**
  • multiple **nodes**
• all start with same GB

what everyone eventually agrees on

view of all participants’ state
• **distributed**
  • multiple nodes
  • message-passing over a network

• all start with same GB
• **distributed**
  • multiple nodes
  • message-passing over a network

• all start with same GB
• have a transaction pool
• **distributed**
  • multiple nodes
  • message-passing over a network

• all start with same GB
• have a transaction pool
• can **mint blocks**
- **distributed** => concurrent
  - multiple nodes
  - message-passing over a network

- multiple transactions can be issued and propagated concurrently
- **distributed** => **concurrent**
  - multiple nodes
  - message-passing over a network

- blocks can be minted without full knowledge of all transactions
• **chain fork** has happened, but nodes don’t know
• as block messages propagate, nodes become aware of the fork
Problem: need to choose

- blockchain “promise” = one globally-agreed chain
  - each node must choose one chain
  - nodes with the same information must choose the same chain
Problem: need to choose

• blockchain “promise” = one globally-agreed chain
  • each node must choose one chain
  • nodes with the same information must choose the same chain
Problem: need to choose

- blockchain “promise” = one globally-agreed chain
  - each node must choose one chain
  - nodes with the same information must choose the same chain
Problem: need to choose

• blockchain “promise” = one globally-agreed chain
  • each node must choose one chain
  • nodes with the same information must choose the same chain
Solution: fork choice rule

- Fork choice rule (FCR, $\succ$):
  - given two blockchains, says which one is “heavier”
  - imposes a strict total order on all possible blockchains
  - same FCR shared by all nodes

- Nodes adopt “heaviest” chain they know
FCR (>)

... > [GB, A, C] > ... > [GB, A, B] > ... > [GB, A] > ... > [GB] > ...

Bitcoin: FCR based on “most cumulative work”
Quiescent consistency

- **distributed**
  - multiple nodes
  - all start with GB
  - message-passing over a network
  - equipped with same FCR

- quiescent consistency: when all block messages have been delivered, everyone agrees
Why it works
<table>
<thead>
<tr>
<th>Definitions</th>
<th>• blocks, chains, block forests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters and</td>
<td>• hashes are collision-free</td>
</tr>
<tr>
<td>assumptions</td>
<td>• \textit{FCR} imposes strict total order</td>
</tr>
<tr>
<td>Invariant</td>
<td>• local state + messages “in flight” = global</td>
</tr>
<tr>
<td>Quiescent</td>
<td>• when all block messages are delivered, everyone agrees</td>
</tr>
</tbody>
</table>
Blocks and chains

\[ b \in \text{Block} ::= \{ \text{prev} : \text{Hash}; \, \text{txs} : \text{Tx}^*; \, \text{pf} : \text{Proof} \} \]

\[ c \in \text{Chain} \triangleq \text{Block}^* \]

- proof-of-work
- proof-of-stake
- proof that this block was minted in accordance to the rules of the protocol
Minting and verifying

\[ \text{try to generate a proof} = \text{"ask the protocol for permission" to mint} \]

\[ \text{mkProof}: \text{Addr} \rightarrow \text{Chain} \rightarrow \text{option Proof} \]

\[ \text{VAF}: \text{Proof} \rightarrow \text{Time} \rightarrow \text{Chain} \rightarrow \text{bool} \]

\[ \text{validate a proof} = \text{ensure protocol rules were followed} \]
Resolving conflict

\[ FCR : \text{Chain} \rightarrow \text{Chain} \rightarrow \text{bool} \]
Assumptions

• Hash functions are collision-free

$$\text{hash\_inj} \quad : \forall x \ y, \ \#x = \#y \implies x = y$$

• FCR imposes a strict total order on all blockchains

$$\text{FCR\_rel} \quad : \forall c_1 \ c_2, \ c_1 = c_2 \lor c_1 > c_2 \lor c_2 > c_1$$

$$\text{FCR\_trans} \quad : \forall c_1 \ c_2 \ c_3, \ c_1 > c_2 \land c_2 > c_3 \implies c_1 > c_3$$

$$\text{FCR\_nrefl} \quad : \forall c, \ c > c \implies \text{False}$$
Invariant: local state + “in-flight” = global
Invariant: local state + “in-flight” = global
Invariant is inductive

- State 1: invariant holds
- State 2: invariant holds
- State 3: invariant holds
- State 4: invariant holds
- State 5: invariant holds
Invariant implies QC

• QC: when all blocks delivered, everyone agrees

How:
  • local state + “in-flight” = global
  • use FCR to extract “heaviest” chain out of local state
    • since everyone has same state & same FCR
      ➢ consensus
Reusable components

• Reference implementation of block forests
• Per-node protocol logic
• Network semantics
• Clique invariant, QC property, various theorems

https://github.com/certichain/toychain
Future work

• Network semantics with nodes joining/leaving at will

• Improved invariants:
  • non-clique topologies
  • network partitions
  • Byzantine faults

• Verified smart contracts platform
Take away

• Formalisation of a blockchain consensus protocol in Coq:
  • minimal set of required security primitives
  • per-node protocol logic & data structures
  • network semantics

• global eventual consistency in a clique topology

https://github.com/certichain/toychain