Consensus in blockchains:
Overview and recent results

Christian Cachin
University of Bern

ViSP, 2023
Overview

• for \( \text{model} \in \) all kinds of blockchain consensus do
  – describe \( \text{model} \)

• while time lasts do
  – select some \( \text{result} \in \) https://crypto.unibe.ch/pub
  – present \( \text{result} \)

• Answer your questions
1 – Threshold trust

• Trust by numbers
  – \( n \) nodes total
  – \( f \) faulty (Byzantine) nodes

• Homogeneous and symmetric

• Requires \( n > 3f \)

• Tendermint, DiemBFT, Quorum ...

\[ n = 7 \]
\[ f = 2 \]
2 – Generalized trust

- Trust by generalized quorums
  - Set of nodes $P$
  - Fail-prone sets consisting of possibly Byzantine nodes
  - Byzantine quorum system

- Heterogeneous and symmetric

- Requires Q3-property
  - Any fail-prone sets must not cover $P$

- Not used by any cryptocurrency (!)
3 – Asymmetric trust

• Subjective generalized quorums

• Every node has its own Byz. quorum system on $P$

• Consistency across nodes' quorum systems

• Requires B3-property
  – $\forall p, p':$ any fail-prone set of $p$ with any set of $p'$ and any set of both must not cover $P$

• Ripple, Stellar, [CT19]
4 – Unstructured, probabilistic voting

- Random sampling of peers
- Exchange information and votes
- Usually coupled with a DAG on transactions
- Avalanche, Conflux, IOTA-Tangle
5 – Stake-based voting

• Stake determines voting power

• Protocols generalized from symmetric voting

• Cosmos, EOS, NEO, Aptos, SUI ...
6 – Stake-based probabilistic choice

- Lottery according to stake
- Probabilistic leader election
- Cryptographic sortition using a verifiable random function (VRF)
- Cardano/Ouroboros ...
7 – Hybrid prob. choice and stake voting

• Stake determines probability or voting power

• Mix of random choice with voting

• Slashing of invested stake upon detection of misbehavior

• Ethereum (LMD-GHOST & FFG-Casper), Polkadot (BABE & GRANDPA), Algorand ...
8 – Proof-of-space and proof-of-delay

• Storage space as resource

• Cryptographic ZK proofs for storage at particular time

• Time delay to prove storage investment over time

• Filecoin, Chia, Storj ...
9 – Proof-of-work

• Demonstrate invested computation

• Nakamoto consensus

• Bitcoin ...
Model 1: Threshold trust

\[ n = 7 \]
\[ f = 2 \]
Order fairness

- Front-running and transaction-reordering attacks in DeFi
- Maximal extractable value (MEV)
- Validity of consensus (total-order broadcast) leaves actual order open
- Validator nodes exploit their freedom and choose a profitable order
Order fairness: Respect the receive-order

Clients produce transactions.

Transactions arrive on validators at different times.

$m_2$ → $m'$ → $m''$ → $m'''$

Input order.
Condorcet: A fair order may not exist

Cycle in the order preferences!

[KZGJ20]: Fair order = deliver all at the “same time” = “block-order fairness”
Differential (block-)-order fairness [CMSZ22]

• $b(m,m')$: number of correct nodes that receive as input $m$ before $m'$

• $f$ out of $n$ corrupted nodes

• **Differential order fairness**: If $b(m,m') > b(m',m) + 2f$, then no correct node delivers $m'$ before $m$. (But protocol may deliver $m$ and $m'$ together, in same block.)

• Implemented by the quick order-fair atomic broadcast protocol, for $n > 3f$
Model 2: Generalized trust
Generalized trust – Byz. quorum systems

- Set of nodes \( P = \{p_1, ..., p_n\} \)
- Fail-prone system \( F \subseteq 2^P \):
  - All \( F \in F \) may fail together

- Quorum system \( Q \subseteq 2^P \), any \( Q \in Q \) is a "quorum" [MR98, HM00]

- \( F = \{pq, pr, qr, xy, xz, yz\} \)
- \( Q = \{rxyz, qxyz, pxyz, pqrz, pqry, pqrx\} \)

- Nodes are trusted **differently**
- All nodes trust **equally**
Do not trust in numbers [AC22]

- Distributed cryptography beyond the threshold model
- Theoretically well-known, practically never explored
- Example access structure (quorum set) of a validator in Stellar (SDF1)

```json
{
  "select": 6,
  "out-of": [
    {"select": 2, "out-of": ["Blockdaemon1", "Blockdaemon2", "Blockdaemon3"]},
    {"select": 2, "out-of": ["SDF1", "SDF2", "SDF3"]},
    {"select": 2, "out-of": ["WirexSingapore", "WirexUK", "WirexUS"]},
    {"select": 2, "out-of": ["CoinquestFinland", "CoinquestHongKong", "CoinquestGermany"]},
    {"select": 2, "out-of": ["FranklinTempleton1", "FranklinTempleton2", "FranklinTempleton3"]},
    {"select": 3, "out-of": ["LOBSTR1", "LOBSTR2", "LOBSTR3", "LOBSTR4", "LOBSTR5"]},
    {"select": 2, "out-of": ["Hercules", "Lyra", "Boötes"]}
  ]
}```
Do not trust in numbers [AC22]

- Practical implementation of generalized cryptosystems
  - Monotone span programs (MSP)

- Verifiable secret sharing (VSS)
- Common coin
- Distributed signatures

- Tools to generate MSP from a configuration file

- Benchmarks show the approach is practical
Do not trust in numbers: Verifiable Secret Sharing [AC22]

- Share and Reconstruct steps of generalized verifiable secret sharing
- Polynomial (n/2), MSP (n/2), MSP (unbalanced) and MSP (grid) structures
Model 3: Asymmetric trust
Asymmetric trust

- Subjective trust assumption of \( p \) (via failures)
  - \( p \) itself never fails
  - Neighbor nodes \( q \) and \( r \)
    
    *May fail alone, not together with others*
  - Remote nodes \( x \), \( y \), \( x \)
    
    *Any 2 of these 3 may fail together*

- Fail-prone system of node \( p \)
  \{q, r, xy, yz, xz\}

- Each one of the 6 nodes uses its own subjective trust like this
  → Asymmetric quorums

- Nodes are trusted **differently**.

Nodes trust **differently** (asymmetric).
Why asymmetric trust?

• For Romans:
  – *De gustibus non est disputandum.*
    (One cannot argue about taste.)

• For CISOs:
  – *One cannot argue about security assumptions.*

• For blockchainers:
  – *A node counts only the votes of nodes that it trusts.* (Ripple, 2014)
  – *Every node has a different idea about which other nodes are important.* (Stellar, 2016)
Example asymmetric quorum system

• Six nodes, arranged in a ring
• Failure assumptions of node $p$ as shown
• All others are (rotation-)symmetric to $p$

• Satisfies B3 property
  $\iff$
  There is an asymmetric quorum system

• Each node mistrusts some 2-set of other nodes: impossible with threshold Byzantine quorums!
Execution model

• An execution defines the actually faulty nodes $F$

• A node $p_i$ is one of
  – Faulty – $p_i \in F$
  – Naive $p_i$ – $p_i \notin F$ and $F \notin F_i^*$
  – Wise $p_i$ – $p_i \notin F$ and $F \in F_i^*$

• Safety and liveness hold only for wise nodes
  – Naive nodes may be cheated
    (cf. ordinary, symmetric model, when $f \geq n/3$: all nodes are naive!)

• Liveness depends on existence of a guild
  – A guild is a set of wise nodes that contains one quorum for each member node
SWMR regular register protocol with Byzantine processes (process $p_i$).

State

$\text{wts}$: sequence number of write operations, stored only by writer $p_w$

$\text{rid}$: identifier of read operations, used only by reader

$ts, v, \sigma$: current state stored by $p_i$: timestamp, value, signature

**upon invocation** $\text{write}(v)$ do

\[
\text{wts} \leftarrow \text{wts} + 1
\]

\[
\sigma \leftarrow \text{sign}_w(\text{WRITE}||w||\text{wts}||v)
\]

send message $[\text{WRITE}, \text{wts}, v, \sigma]$ to all $p_j \in \mathcal{P}$

wait for receiving a message $[\text{ACK}]$ from more than $\frac{n+f}{2}$ processes

**upon invocation** $\text{read}$ do

\[
\text{rid} \leftarrow \text{rid} + 1
\]

send message $[\text{READ}, \text{rid}]$ to all $p_j \in \mathcal{P}$

wait for receiving messages $[\text{VALUE}, r_j, ts_j, v_j, \sigma_j]$ such that

\[
r_j = \text{rid} \text{ and } \text{verify}_w(\sigma_j, \text{WRITE}||w||\text{ts}||v_j)
\]

return $\text{highesťval}([\{ts_j, v_j\}])$

**upon** receiving a message $[\text{WRITE}, ts', v', \sigma']$ from $p_w$ do

\[
\text{if } ts' > ts \text{ then}
\]

\[
(ts, v, \sigma) \leftarrow (ts', v', \sigma')
\]

send message $[\text{ACK}]$ to $p_w$

**upon** receiving a message $[\text{READ}, r]$ from $p_r$ do

send message $[\text{VALUE}, r, ts, v, \sigma]$ to $p_r$
Asymmetric SWMR regular register protocol (process $p_i$).

**State**

- $wts$: sequence number of write operations, stored only by writer $p_w$
- $rid$: identifier of read operations, used only by reader
- $ts, v, \sigma$: current state stored by $p_i$; timestamp, value, signature

**upon invocation** write($v$) do

1. $wts \leftarrow wts + 1$
2. $\sigma \leftarrow \text{sign}_w(\text{WRITE}, wts, v)$
3. send message [WRITE, $wts, v, \sigma$] to all $p_j \in \mathcal{P}$
4. wait for receiving a message [ACK] from all processes in some quorum $Q_w \subseteq Q_w$

**upon invocation** read do

1. $rid \leftarrow rid + 1$
2. send message [READ, $rid$, $r$] to all $p_j \in \mathcal{P}$
3. wait for receiving messages [VALUE, $r_j, ts_j, v_j, \sigma_j$, $\text{verify}_w(\sigma_j, \text{WRITE}, wts, v_j)$] from all processes in some $Q_r \subseteq Q_r$, such that
4. $r_j = rid \text{ and } \text{verify}_w(\sigma_j, \text{WRITE}, wts, v_j)$
5. return highestval($\{ (ts_j, v_j) | j \in Q_r \}$)

**upon** receiving a message [WRITE, $ts', v', \sigma'$] from $p_w$ do

1. if $ts' > ts$ then
   - $(ts, v, \sigma) \leftarrow (ts', v', \sigma')$
   - send message [ACK] to $p_w$

**upon** receiving a message [READ, $r$] from $p_r$ do

1. send message [VALUE, $r, ts, v, \sigma$] to $p_r$
Model 4: Unstructured, probabilistic voting
Analysis of Avalanche consensus [ACT22]

• Metastable consensus: Avalanche and the snow family of protocols

• Transactions form a DAG, a directed acyclic graph

• Transactions may conflict

• Nodes sample other nodes and ask for their opinion
Avalanche consensus

- While true do
  - Select a new transaction $T$
  - Query $k$ parties with $T$
  - If more than $\alpha$ positive answers
    - Update the DAG and increment the counter for acceptance of every ancestor
  - Else
    - Reset the counter for acceptance of every ancestor to 0

$2 \geq \alpha = 2$
Analysis of Avalanche consensus [ACT22]

• Detailed pseudocode of Avalanche protocol

• Independent analysis

• Illustrates a potential problem

• For other reasons, Ava Labs/Avalanche abandons the DAG protocol in March '23
Model 9: Proof-of-work
Use any resource for consensus [ACLVZ22]

- Longest-chain consensus based on an abstract resource
- Formal model of a resource allocator
- Resources: work, stake, storage ...
- Which features must a resource have to enable consensus?
Thank you!

Web – https://crypto.unibe.ch/

Blog – https://cryptobern.github.io/

Twitter – https://twitter.com/cczurich/
References
Literature

- Model 1: Threshold trust
Literature

- **Model 2: Generalized trust**
  
  
• Model 3: Asymmetric trust
Literature

- Model 3: Asymmetric trust
Literature

- Model 4: Unstructured, probabilistic voting
• Models 5-7: Stake based
Literature

• Model 9
