UNIVERSITÄT BERN

Consensus in blockchains: Overview and recent results

Christian Cachin
University of Bern

ChainScience, Zürich, April 2024

Overview

- **for** *model* $\in$ all kinds of blockchain consensus **do**
  - describe *model*

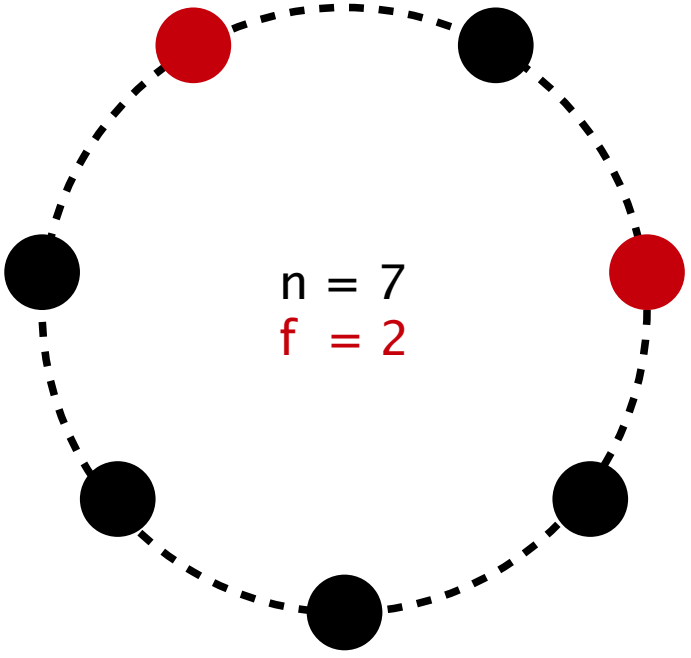
- **while** time lasts **do**
  - select some *result* $\in$ `https://crypto.unibe.ch/pub`
  - present *result*

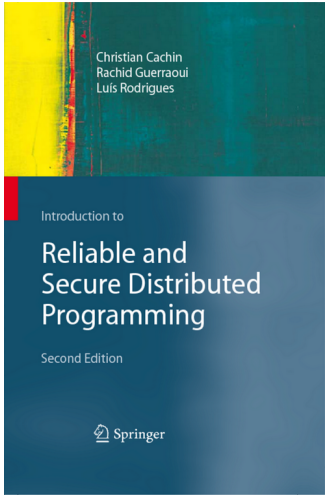
- Answer your questions

Consensus overview

1 – Threshold trust (BFT)

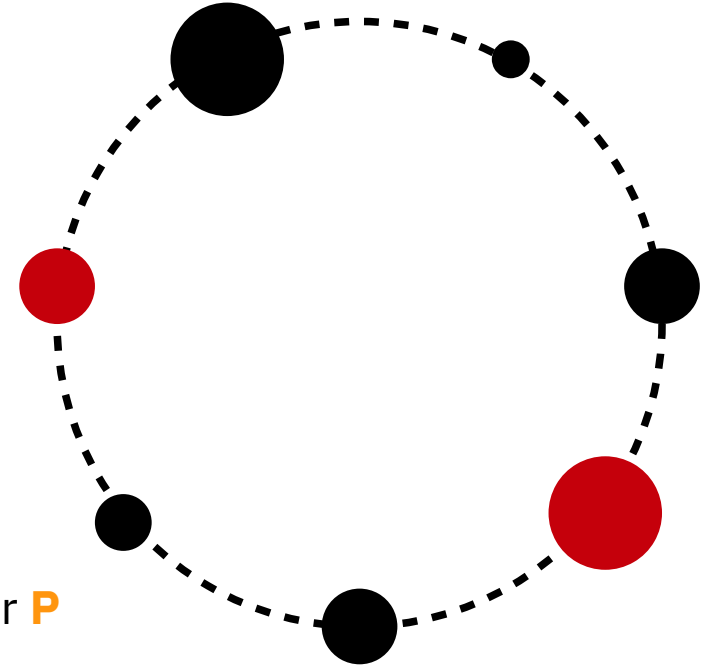
- Trust by numbers
  - n nodes total
  - f faulty (Byzantine) nodes
- Nodes are identified
  - Proof-of-Authority (PoA)
- Homogeneous and symmetric
- Requires n > 3f
- Tendermint/Cosmos, Internet Computer (DFINITY), VeChain, BNB SC, Hashgraph, TRON …





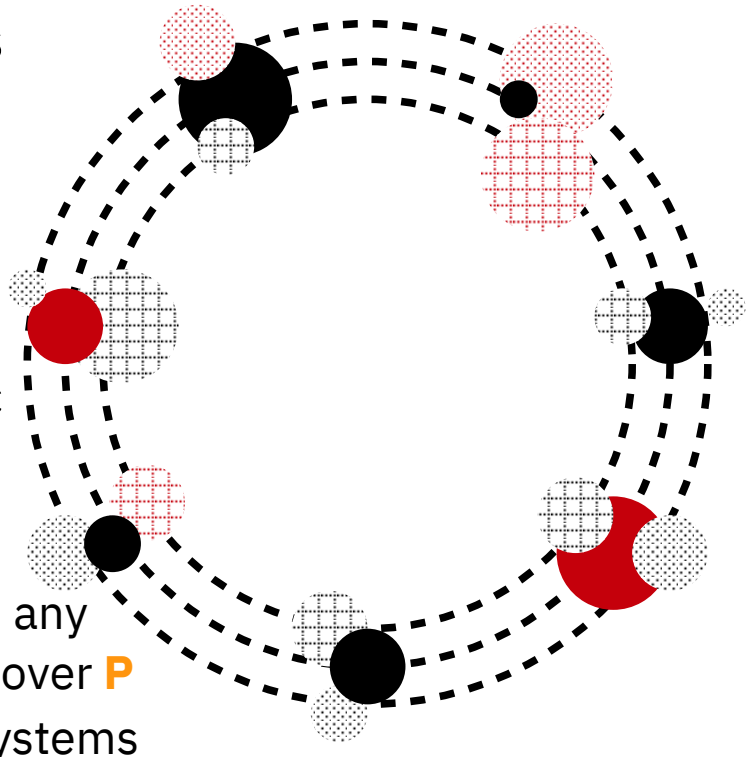
2 – Generalized trust

- 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 3 fail-prone sets must not cover **P**
- Not used by any cryptocurrency (!)



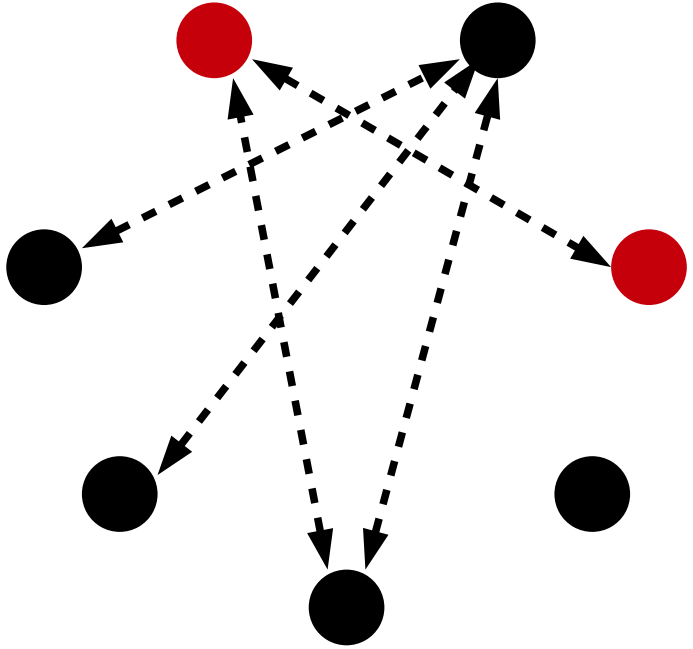
3 – Asymmetric trust

- Subjective generalized quorums
- Every node has its own Byz. quorum system on **P**
- Heterogeneous and asymmetric
- Requires B3-property
  - ∀ p, p' : any fail-prone set of p with any set of p' and any of both must not cover **P**
  - Consistent across nodes quorum systems
- Ripple, Stellar, [CT19]



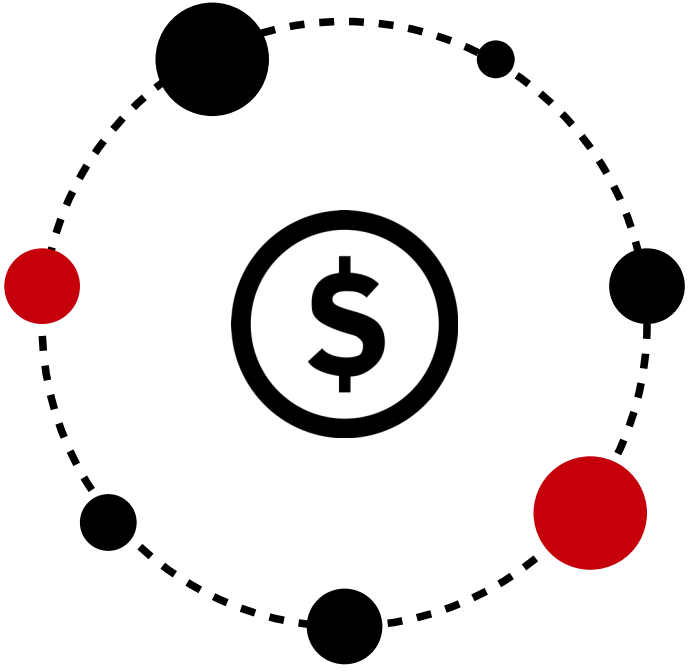
4 – Unstructured, probabilistic voting

- Random sampling of peers
- Exchange information and votes
- Often coupled with a DAG (directed acyclic graph) on transactions
- Avalanche, Conflux, IOTA-Tangle



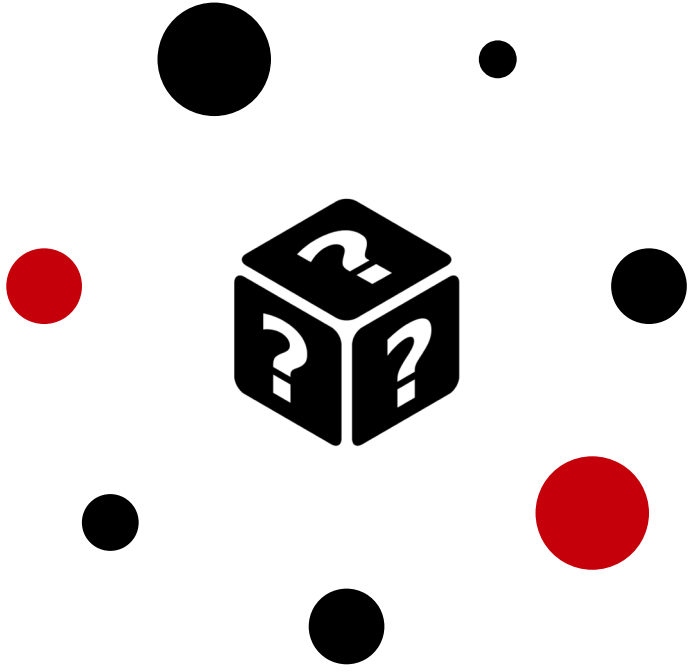
5 – Stake-based voting

- Stake determines voting power
  - Including delegated stake (DPoS)
- Protocols generalized from symmetric voting (BFT)
- Slashing of invested stake upon detection of misbehavior
- Tendermint/Cosmos, EOS, NEO, Aptos, SUI, BNB SC …



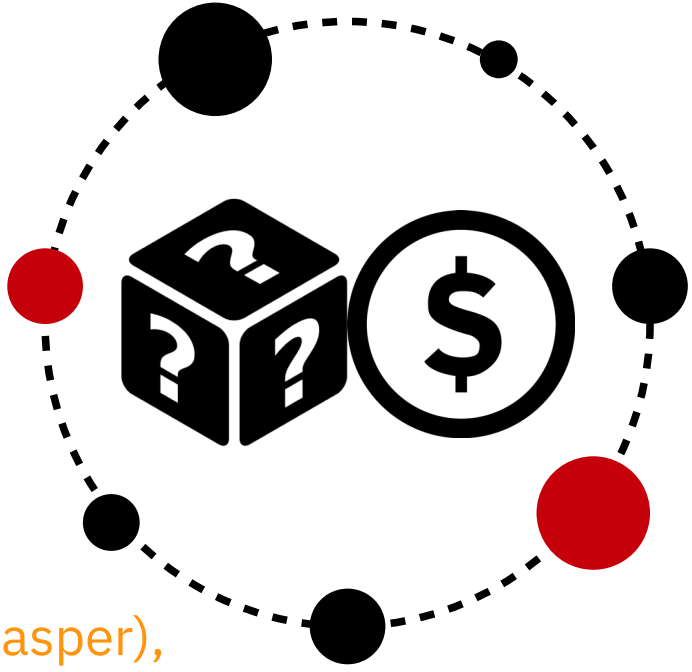
6 – Stake-based probabilistic choice

- Lottery according to stake
- Probabilistic leader election
- Cryptographic sortition using a verifiable random function (VRF)
- Cardano/Ouroboros …



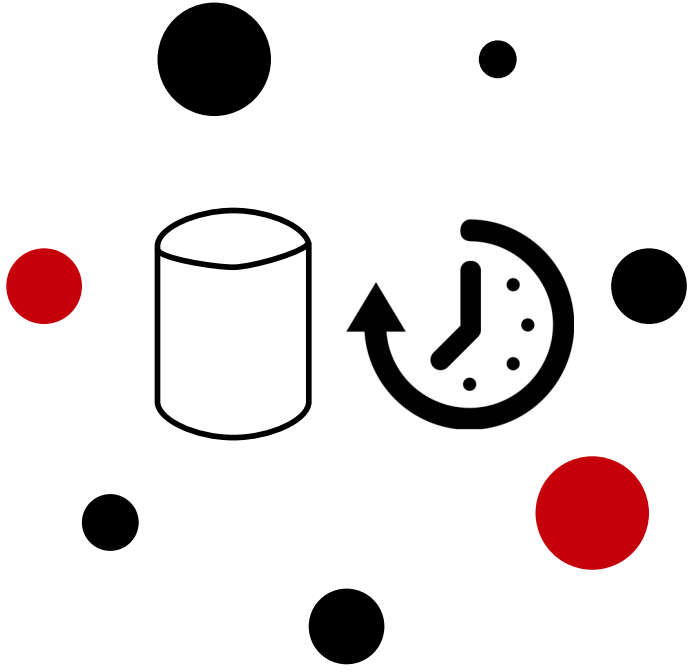
7 – Hybrid prob. choice and stake voting

- 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 ...



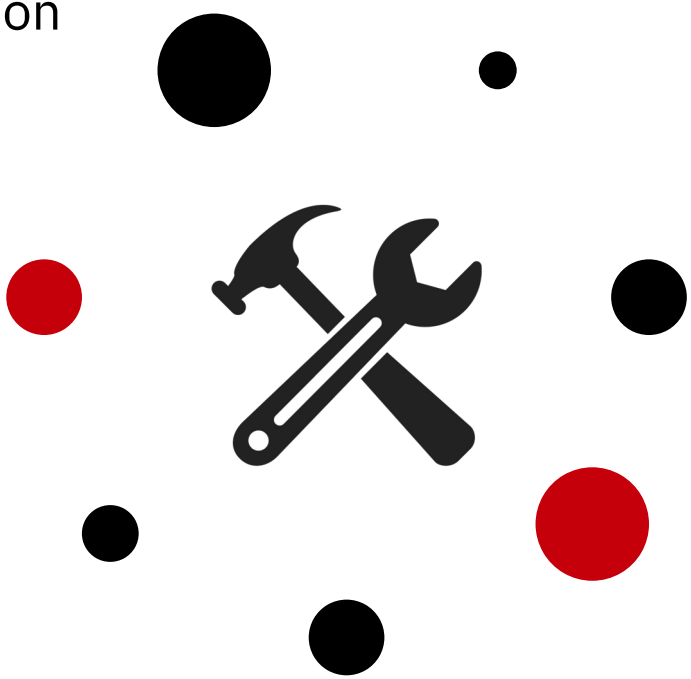
8 – Proof-of-space and proof-of-delay

- Storage space as resource
- Cryptographic ZK proofs for storage at particular time
- Time delay to prove storage investment over time
- Filecoin, Chia, Storj …



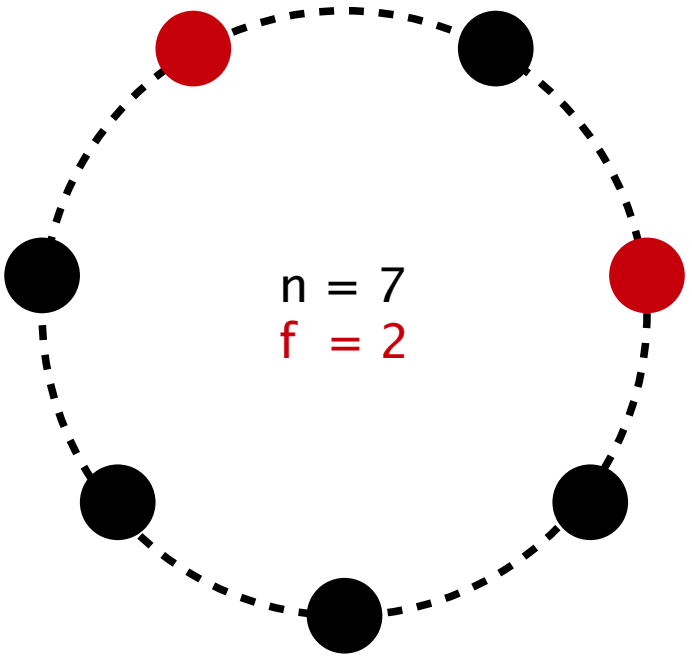
9 – Proof-of-work

- Demonstrate invested computation
- Nakamoto consensus
- Bitcoin and variations,
  Litecoin, Dogecoin,
  Ethereum (1.0) and variations,
  Ethereum Classic,
  Monero, ZCash ...



Recent results

Model 1: Threshold trust



Order fairness, frontrunning, and MEV

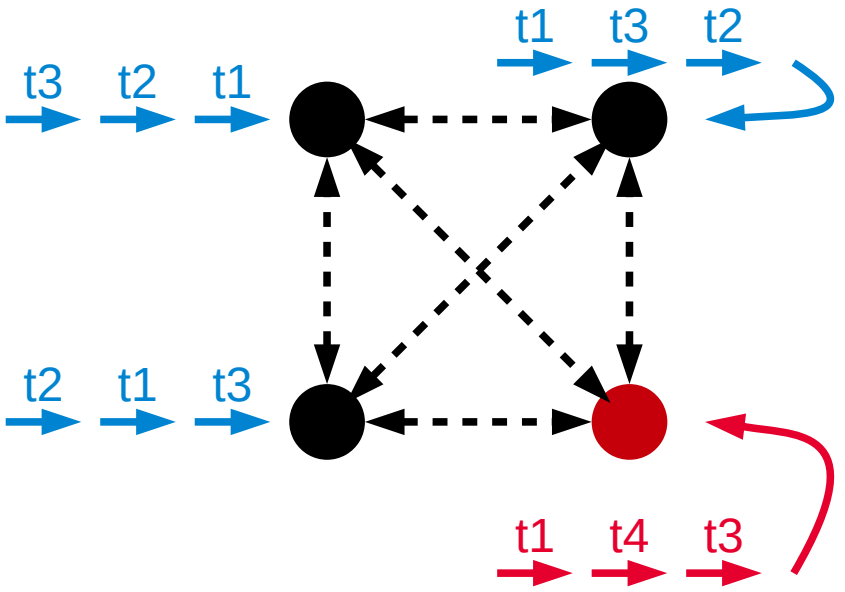
- Front-running and transaction-reordering attacks are common in DeFi
  - Online exchanges (DEXs) can be attacked by malicious insiders
  - Sandwich attack:
    - A customer trade ($X \to Y$) changes the relative asset price X/Y
    - Leader injects two malicious trades that "sandwich" the victim trade $X \to Y$:
      - $a \cdot Y \to b \cdot X$; $X \to Y$; $b \cdot X \to a^* \cdot Y$, where $a^* > a$ because price of Y to X has increased
- Maximal extractable value (MEV)
- Ordinary consensus protocols leaves the actual order open
  - See validity condition of total-order broadcast
- Validator nodes exploit their freedom and choose a profitable order

Fairness from receive-order

- Fairness means that output-order respects receive-order at nodes

- "If all nodes receive t1«t2, then t1«t2 in output order."

- But ...
  - There may be Condorcet cycles
  - Two correct nodes each have t1«t2 & t2«t3 & t3«t1
  - Faulty nodes may lie about their receive-order: t3«t4«t1

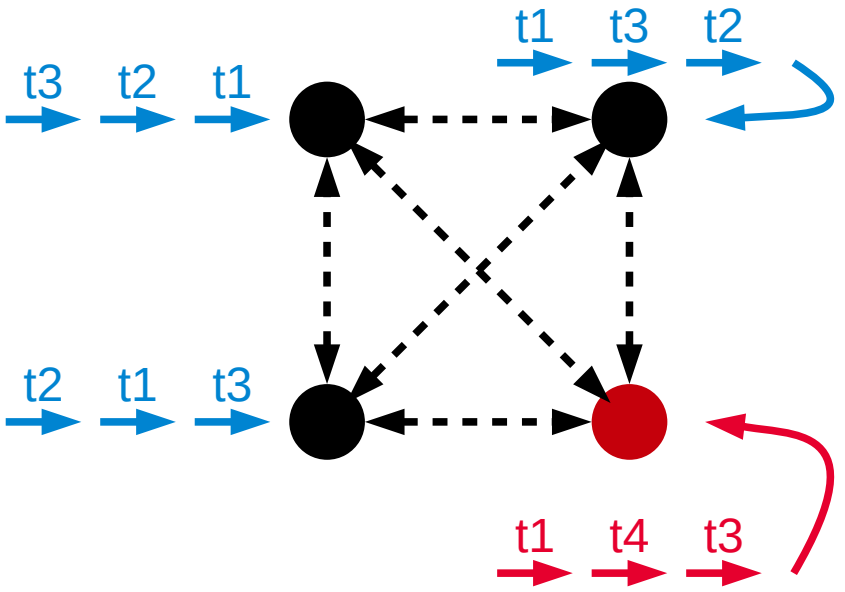


Block-wise order fairness [KZGJ20]

- If protocol is sure that t1«t2 in input for enough nodes, then *output* t1 *before* t2, written t1 ≺ t2 .
- If uncertain whether t1«t2 in input for enough nodes, then *output* t1 *together with* t2, denoted **[**t1|t2**]** , in a "block."

- Ex. output sequence
  … t0 ≺ **[**t1|t2|t3**]** ≺ t4 ≺ …

- Application must deal with "concurrent" transactions in a block



Differential (block-)order fairness [CMSZ22]

UNIVERSITÄT BERN

- b(tx,tx'): number of correct nodes that *receive as input* tx before tx'

- **Differential order fairness**: Require that whenever b(tx,tx') > b(tx',tx) + 2f, no correct node outputs tx' before tx.
  - But protocol may output tx and tx' together, in same block.
  - n nodes, f of them corrupted

- **Theorem**: If b(tx,tx') ≤ b(tx',tx) + 2f, then no protocol can respect input order.

- With optimal resilience n = 3f + 1:
  - If t1«t2 for all correct nodes, then must output t1 ≺ t2.
  - Otherwise (some correct node t1«t2, some correct node t2«t1), then output **[**t1|t2**]**.

Quick order-fair atomic broadcast

UNIVERSITÄT BERN

- Every node reports its own received sequence to all with consistent broadcast

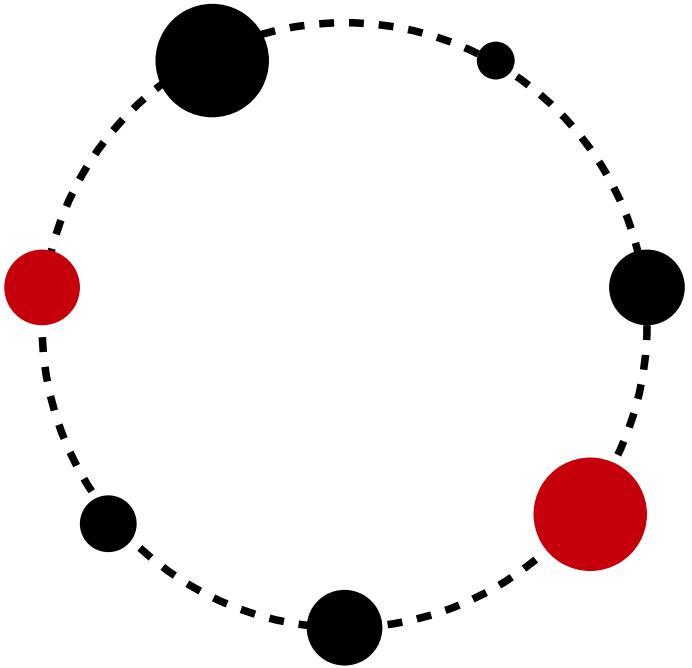
- Periodically, consensus protocol takes a cut | (across all received sequences)
  p1: ...«t1«t2«t3«t4|
  p2: ...«t2«t3«t1«t5«t4|«t6
  p3: ...«t3«t4«t1«t5|«t6
  p4: ...«t3«t1«t2|«t4 ...

- Compute graph of tx s.t. tx → tx' whenever b(tx,tx') > b(tx',tx) + 2f is possible

- Collapse cycles in tx graph to nodes, then output tx in topo-sort order of graph

- Complexity $O(n^2)$ messages, resilience $n > 3f$

Model 2: Generalized trust



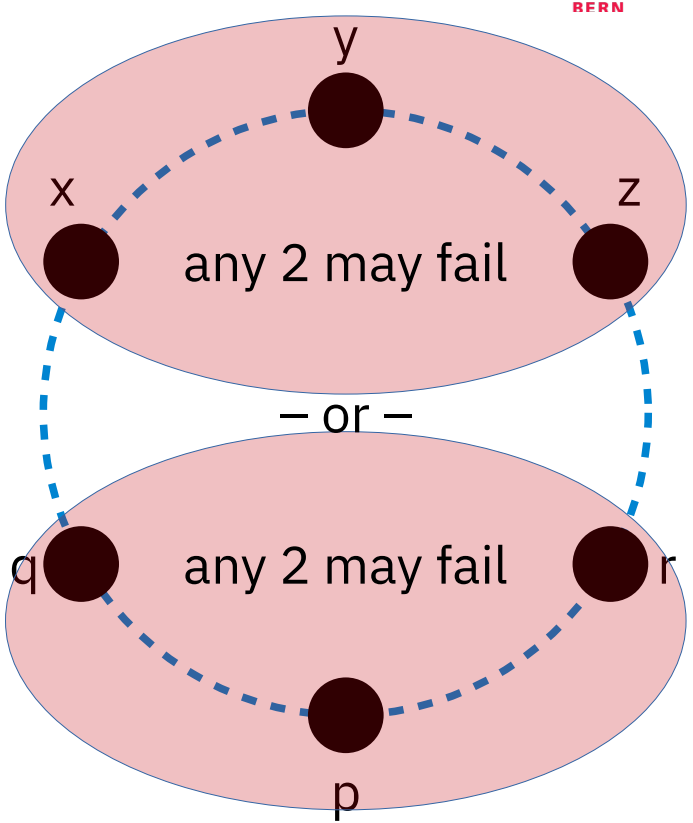
Generalized trust – Byz. quorum systems

- Set of nodes **P** = {p1, …, pn}
- Fail-prone system **F** ⊆ $2^{\mathbf{P}}$:
  – All F ∈ **F** may fail together

- Quorum system **Q** ⊆ $2^{\mathbf{P}}$, any Q ∈ **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 [AC23]

UNIVERSITÄT BERN

- **Consensus and distributed cryptography** beyond the threshold model
  – Threshold cryptography: nodes collectively hold a cryptographic key

- Theoretically well-known, practically never explored

- Example access structure (quorum set) of a validator in Stellar (SDF1)


```
{  "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": ["CoinqvestFinland", "CoinqvestHongKong", "CoinqvestGermany"]},
      {"select": 2, "out-of": ["SatoshiPayUS", "SatoshiPaySG", "SatoshiPayDE"]},
      {"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 [AC23]

UNIVERSITÄT BERN

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

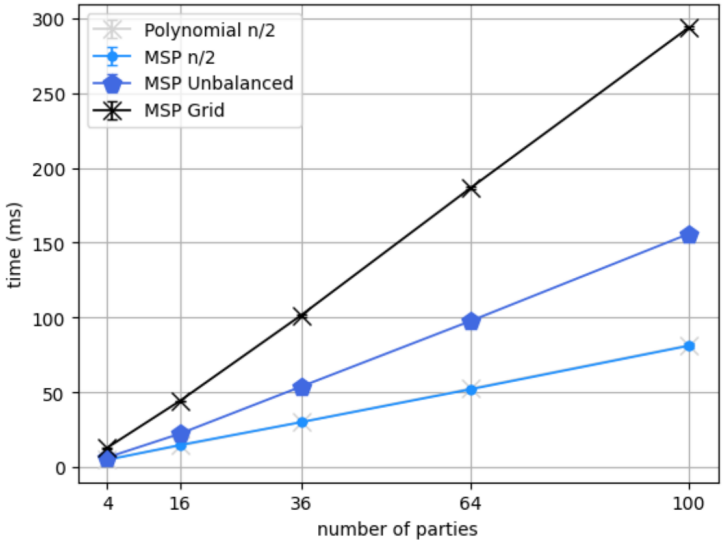
- Verifiable secret sharing (VSS)
- Common coin
- Distributed signatures

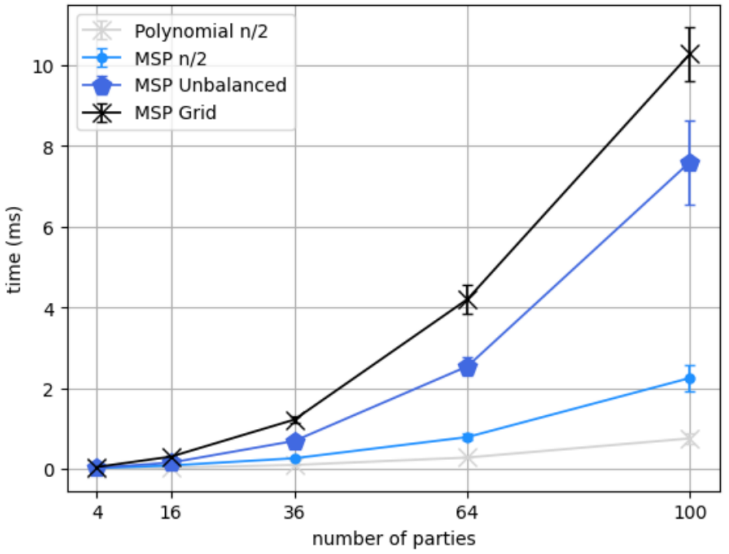
- Tools to generate MSP from a configuration file

- Benchmarks show the approach is practical

Do not trust in numbers: Verifiable Secret Sharing [AC23]

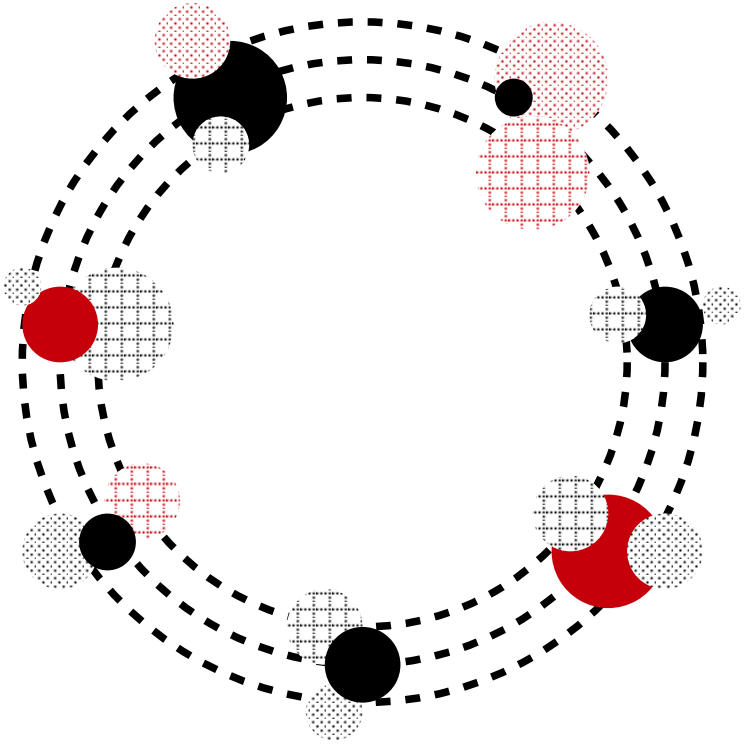
UNIVERSITÄT BERN





- Latencies of Share and Reconstruct op. in generalized verifiable secret sharing
- Polynomial (n/2), MSP (n/2), MSP (unbalanced) and MSP (grid) structures

Model 3: Asymmetric trust

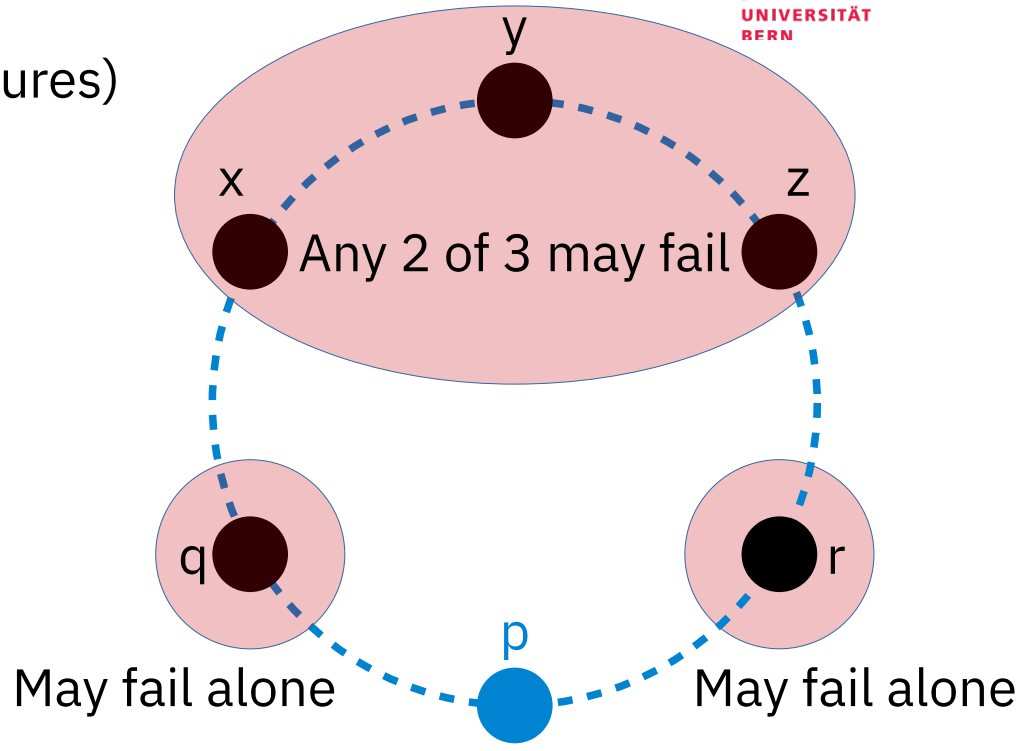


Asymmetric trust

- 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.*

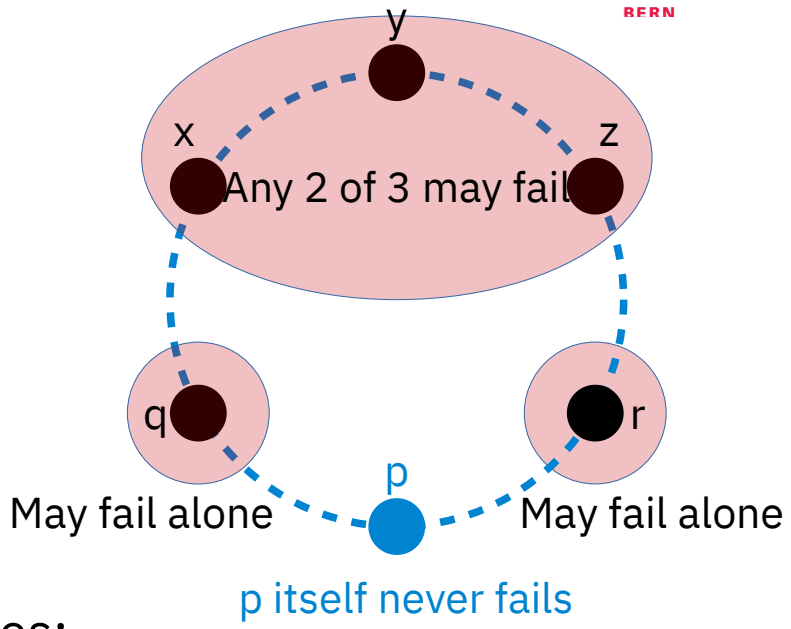
- For blockchainers:
  - *A node counts only the votes of nodes that it trusts. (Ripple, 2014)*
  - *Every node has a different idea about which other nodes are important. (Stellar, 2016)*

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
  ↔
  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 $F \notin \mathbf{F_i}^*$
  - Wise p_i – $p_i \notin F$ and $F \in \mathbf{F_i}^*$

- 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

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

upon invocation *write*(v) **do** // only if p_i is writer p_w
  $wts \leftarrow wts + 1$
  $\sigma \leftarrow_{\$} sign_w(\text{WRITE}\|w\|wts\|v)$
  send message $[\text{WRITE}, wts, v, \sigma]$ to all $p_j \in \mathcal{P}$
  wait for receiving a message $[\text{ACK}]$ from *more* than $\frac{n+f}{2}$ processes

upon invocation *read* **do** // only if p_i is reader p_r
  $rid \leftarrow rid + 1$
  send message $[\text{READ}, rid]$ to all $p_j \in \mathcal{P}$
  wait for receiving messages $[\text{VALUE}, r_j, ts_j, v_j, \sigma_j]$ from *more* than $\frac{n+f}{2}$ processes **such that**
    $r_j = rid$ **and** $verify_w(\sigma_j, \text{WRITE}\|w\|ts\|v_j)$
  return $highestval(\{(ts_j, v_j)\})$

upon receiving a message $[\text{WRITE}, ts', v', \sigma']$ from p_w **do** // every process
  if $ts' > ts$ **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** // every process
  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, σ: current state stored by p_i: timestamp, value, signature

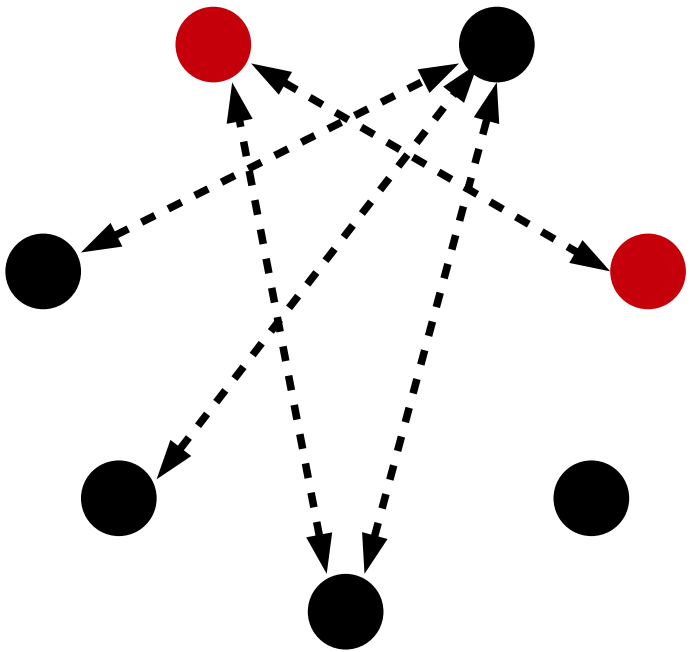
upon invocation *write*(v) **do** // only if p_i is writer p_w
  $wts \leftarrow wts + 1$
  $\sigma \leftarrow_{\$} sign_w(\text{WRITE}\|w\|wts\|v)$
  send message $[\text{WRITE}, wts, v, \sigma]$ to all $p_j \in \mathcal{P}$
  wait for receiving a message $[\text{ACK}]$ from all processes in some quorum $Q_w \in \mathcal{Q}_w$

upon invocation *read* **do** // only if p_i is reader p_r
  $rid \leftarrow rid + 1$
  send message $[\text{READ}, rid]$ to all $p_j \in \mathcal{P}$
  wait for receiving messages $[\text{VALUE}, r_j, ts_j, v_j, \sigma_j]$ from all processes in some $Q_r \in \mathcal{Q}_r$ **such that**
    $r_j = rid$ **and** $verify_w(\sigma_j, \text{WRITE}\|w\|ts\|v_j)$
  return $highestval(\{(ts_j, v_j) | j \in Q_r\})$

upon receiving a message $[\text{WRITE}, ts', v', \sigma']$ from p_w **do** // every process
  if $ts' > ts$ **then**
    $(ts, v, \sigma) \leftarrow (ts', v', \sigma')$
  send message $[\text{ACK}]$ to p_w

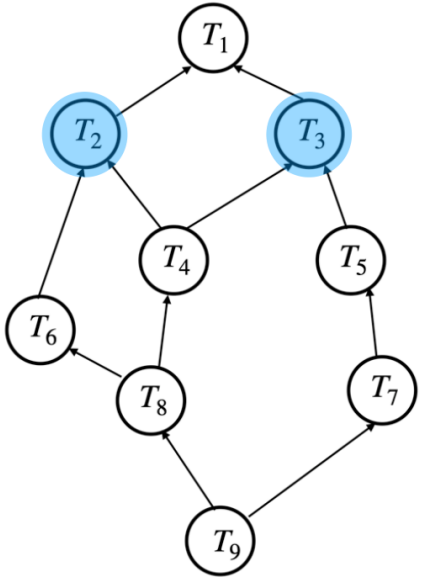
upon receiving a message $[\text{READ}, r]$ from p_r **do** // every process
  send message $[\text{VALUE}, r, ts, v, \sigma]$ to p_r

Model 4: Unstructured, probabilistic voting



Analysis of Avalanche consensus I [ACT22]

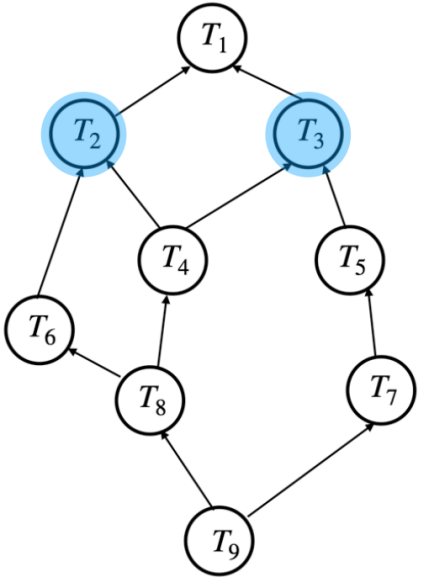
- Metastable consensus: Avalanche and the snow family of protocols
- Nodes sample k other nodes randomly and ask for their opinion
- Transactions form a DAG, a directed acyclic graph
- Transactions without dependencies (T2 and T3) may be delivered (accepted) in any order
- Transactions may conflict



T2 and T2 independent

Avalanche consensus

- **while** TRUE **do**
  - select some transaction T
  - pick k random parties and query them about T
  - **if** more than α positive results **then**
    - update DAG: for every ancestor T' of T, increment counter(T') for acceptance
  - **else**
    - update DAG: for every ancestor T' of T, reset (to 0) counter(T') for acceptance
  - **if** (∃ T* that is not conflicting ∧ counter(T*) ≥ β1) ∨ (∃ T* that is conflicting ∧ counter(T*) ≥ β2) **then**
    - **deliver (accept)** T



Conflicting tx can come to exist in the DAG.

Referencing them cleverly can delay acceptance of innocent tx.

Analysis of Avalanche consensus I [ACT22]

UNIVERSITÄT BERN

- Detailed pseudocode of Avalanche protocol
- Independent analysis
- Illustrates a potential problem
  - Adversary may delay acceptance of a victim transaction arbitrarily
- For other reasons, Ava Labs/Avalanche abandons the DAG protocol in March '23

Analysis of Avalanche consensus II [ACS24]

UNIVERSITÄT BERN

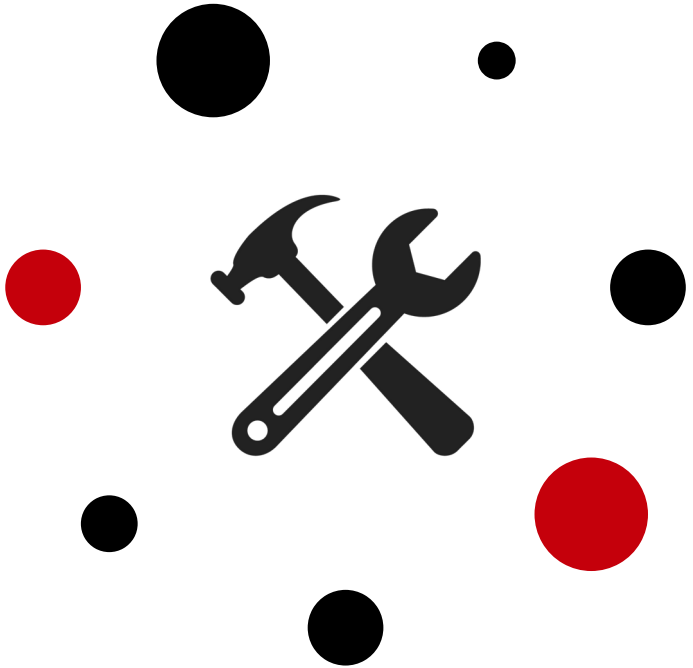
- Avalanche protocol family
  - Slush → Snowflake → Snowball → [Snowman →] Avalanche

- Revisit randomized polling of Slush as plurality consensus
  - Number of nodes n
  - Number of queries k
  - Security parameter β

- Consensus needs $\mathbf{\Omega}(\log n / \log k)$ rounds

- A variation Slush achieves consensus in $\mathbf{O}(\beta + \log n)$ rounds

Model 9: Proof-of-work

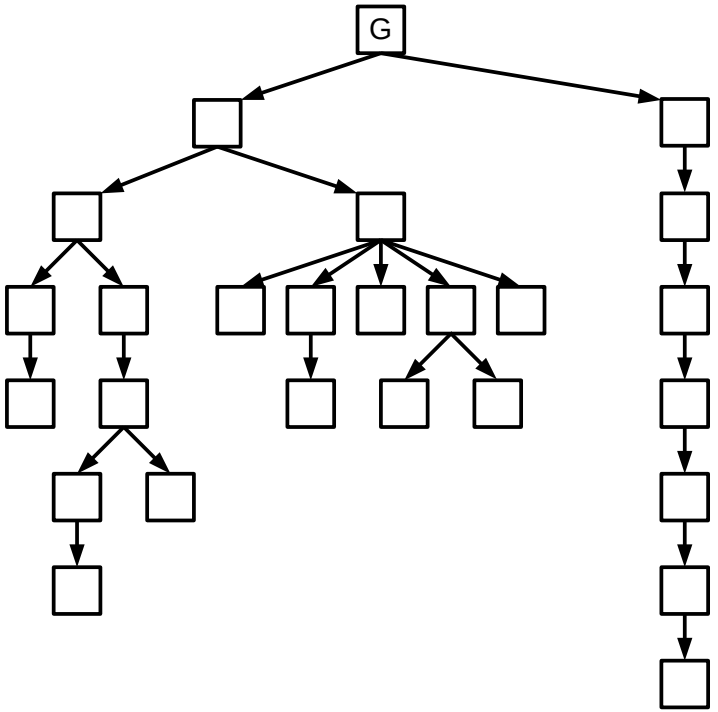


Medium: A bridge from Bitcoin to GHOST [ACP21]

UNIVERSITÄT BERN

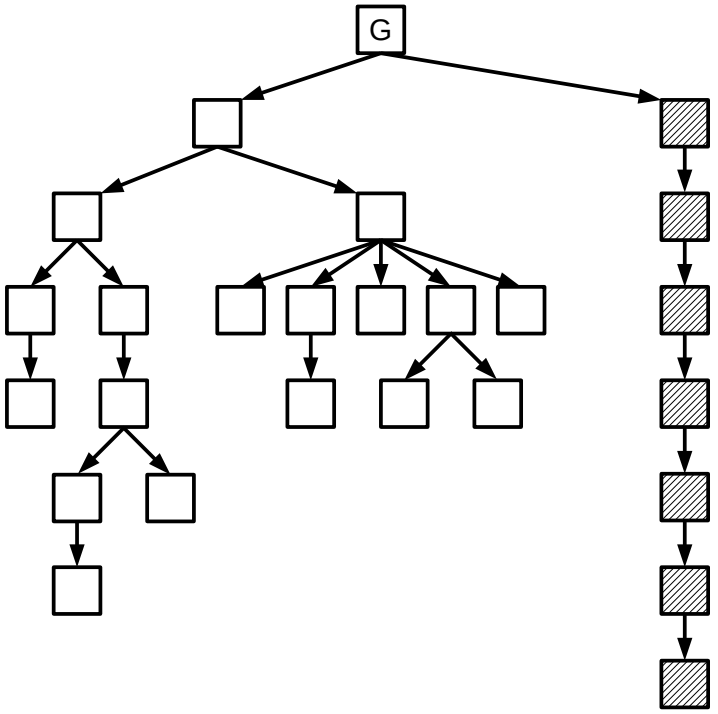
- Nakamoto consensus selects the "longest" chain
- GHOST selects Greedily the Heaviest-Observed Sub-Tree
- GHOST better acknowledges work invested into "stale blocks" on short forks
  - But GHOST also appears to make long-range attacks more feasible
- Is there a tradeoff between Nakamoto's longest-chain rule and GHOST?

Medium: A bridge from Bitcoin to GHOST



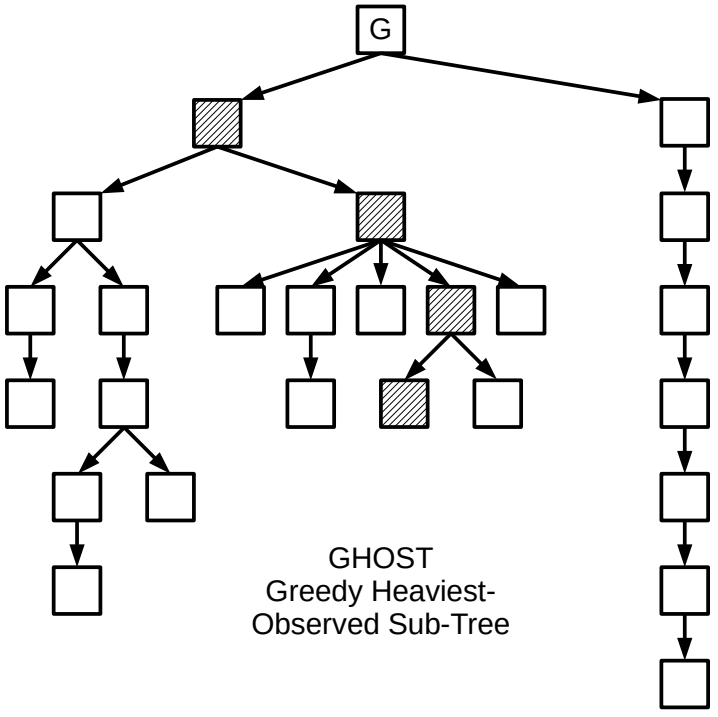
Blocks in subtree at G: 9 10 7

Medium: A bridge from Bitcoin to GHOST



Longest chain / Nakamoto

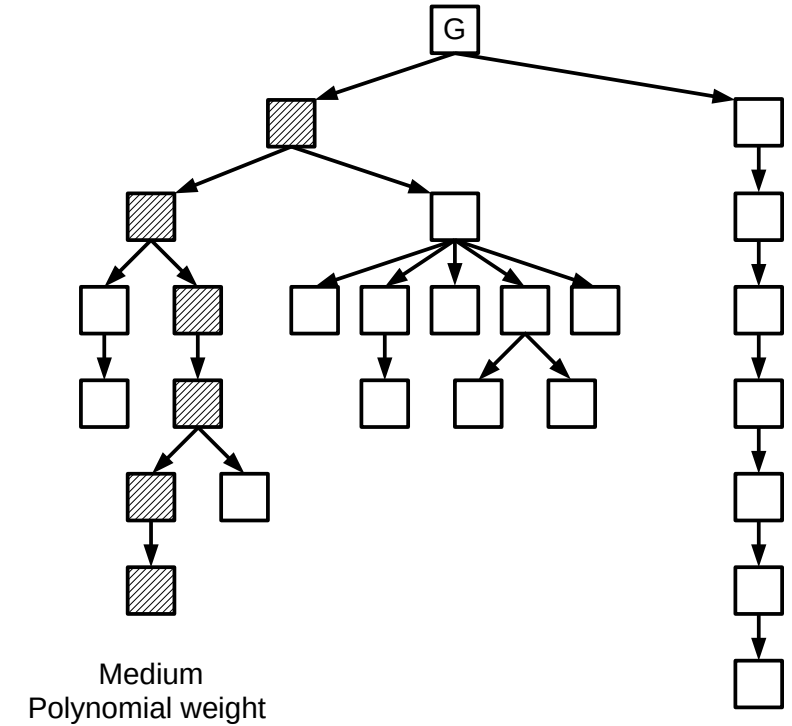
Medium: A bridge from Bitcoin to GHOST



Medium: Fork selection rule

- Nakamoto consensus counts only the length of the chain
- GHOST counts only the number of blocks
- Medium weighs each block exponentially with its depth
  - A block at depth d counts c^d
  - Weight is a polynomial in depth d, evaluated at constant $c \geq 1$
- Special cases
  - $c = 1$: every block counts irrespective of depth ↔ GHOST rule
  - $c = \infty$: a block counts only through its depth ↔ Nakamoto rule

Medium: A bridge from Bitcoin to GHOST



Medium
Polynomial weight
in depth with const = 2

Consensus from an abstract resource [ACLVZ22]

- Longest-chain consensus protocol that uses an abstract resource
- Formal model of a resource allocator
- Resources: work, stake, storage ...
- Which features must a resource have to enable consensus?

UNIVERSITÄT BERN

Thank you!

Web – https://crypto.unibe.ch/

Blog – https://cryptobern.github.io/

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

References

Literature

- Model 1: Threshold trust
  - [CMSZ22] Cachin, C., Mićić, J., Steinhauer, N. & Zanolini, L. (2022). Quick Order Fairness. Proc. Financial Cryptography and Data Security (FC), LNCS 13411, 316–333. https://doi.org/10.1007/978-3-031-18283-9_15

Literature

UNIVERSITÄT BERN

- Model 2: Generalized trust
  - [AC20] Alpos, O., & Cachin, C. (2020). Consensus Beyond Thresholds: Generalized Byzantine Quorums Made Live. Proc. 39th Symposium on Reliable Distributed Systems (SRDS), 31–40. https://doi.org/10.1109/SRDS51746.2020.00010
  - [ACZ21] Alpos, O., Cachin, C., & Zanolini, L. (2021). How to Trust Strangers: Composition of Byzantine Quorum Systems. Proc. 40th Symposium on Reliable Distributed Systems (SRDS), 120–131. https://doi.org/10.1109/SRDS53918.2021.00021
  - [AC23] Alpos, O. & Cachin, C. (2023). Do Not Trust in Numbers: Practical Distributed Cryptography With General Trust. Proc. Stabilization, Safety, and Security of Distributed Systems (SSS), 536-551. https://doi.org/10.1007/978-3-031-44274-2_40

Literature

- Model 3: Asymmetric trust
  - [AZ21] Cachin, C., & Zanolini, L. (2021). Asymmetric Asynchronous Byzantine Consensus. Proc. ESORICS Workshops on Data Privacy Management (DPM), Cryptocurrencies and Blockchain Technology (CBT) LNCS 13140, 192–207. https://doi.org/10.1007/978-3-030-93944-1_13
  - [AZ20] Cachin, C., & Zanolini, L. (2020). From Symmetric to Asymmetric Asynchronous Byzantine Consensus. e-print, arXiv:2005.08795v3 [cs.DC]. https://arxiv.org/abs/2005.08795v3
  - [CLZ22] Cachin, C., Losa, G., & Zanolini, L. (2022). Quorum Systems in Permissionless Proc. 26th International Conference on Principles of Distributed Systems (OPODIS), 17:1–17:22. https://doi.org/10.4230/LIPIcs.OPODIS.2022.17

Literature

- Model 3: Asymmetric trust
  - [ACM21] Amores-Sesar, I., Cachin, C., & Mićić, J. (2021). Security Analysis of Ripple Consensus. Proc. 24th International Conference on Principles of Distributed Systems (OPODIS), 10:1–10:16. https://doi.org/10.4230/LIPIcs.OPODIS.2020.10

Literature

- Model 4: Unstructured, probabilistic voting
  - [ACT22] Amores-Sesar, I., Cachin, C., & Tedeschi, E. (2022). When is Spring coming? A Security Analysis of Avalanche Consensus. Proc. 26th International Conference on Principles of Distributed Systems (OPODIS), 10:1–10:22. https://doi.org/10.4230/LIPIcs.OPODIS.2022.10

  - [ACS24] Amores-Sesar, I., Cachin, C., & Schneider, P. (2024). An Analysis of Avalanche Consensus. e-print, arxiv:2401.02811 [cs.DC]. https://arxiv.org/abs/2401.02811

Literature

- Models 5-7: Stake based
  - [B21] Bürk, T. (2022). Blockchain consensus protocols based on stake. Master thesis, Institute of Computer Science, University of Bern. https://crypto.unibe.ch/archive/theses/2021.msc.timo.buerk.pdf
  - [AC23] Alpos, O., Cachin, C., Holmgaard Kamp, S., & Buus Nielsen, J. (2023). Practical Large-Scale Proof-Of-Stake Asynchronous Total-Order Broadcast. Proc. 5th Conference on Advances in Financial Technologies (AFT), 31:1–31:22. https://doi.org/10.4230/LIPIcs.AFT.2023.31

Literature

- Model 9
  - [ACP21] Amores-Sesar, I., Cachin, C., & Parker, A. (2021). Generalizing Weighted Trees: A Bridge from Bitcoin to GHOST. Proc. 3rd ACM Conference on Advances in Financial Technologies (AFT), 156–169. https://doi.org/10.1145/3479722.3480995

  - [ACLVZ22] Azouvi, S., Cachin, C., Le, D. V., Vukolic, M., & Zanolini, L. (2022). Modeling Resources in Permissionless Longest-Chain Total-Order Broadcast. Proc. 26th International Conference on Principles of Distributed Systems (OPODIS), 19:1–19:23. https://doi.org/10.4230/LIPIcs.OPODIS.2022.19