Consensus in blockchains: An overview

Christian Cachin
University of Bern

a16z, August 2024

Some history

History 1 - CC

History 1 - CC

History 1 - CC

IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 18, NO. 4, APRIL 2000 593

Optimistic Fair Exchange of Digital Signatures

N. Asokan, *Member, IEEE*, Victor Shoup, *Member, IEEE*, and Michael Waidner, *Member, IEEE*

History 1 - CC

UNIVERSITÄT BERN

Random Oracles in Constantinople:
Practical Asynchronous Byzantine Agreement Using Cryptography
(Extended Abstract)

Christian Cachin Klaus Kursawe Victor Shoup

IBM Research
Zurich Research Laboratory
CH-8803 Rüschlikon, Switzerland
{cca,kku,sho}@zurich.ibm.com

History 2 - Consensus protocols

History 2 - Consensus protocols

- **1980 until 2000:** Theory research – many theorems, no systems, no prototypes

- **2000 until 2010:** Systems research – many prototypes, no products

UNIVERSITÄT BERN

BFTW^3: Why? When? Where? Workshop on Theory and Practice of Byzantine Fault Tolerance

Affiliated with DISC 2009

September 22, 2009
Elche, Spain

The workshop gathers researchers from both theory and systems communities and aims at understanding why the impressive research activity in the area of Byzantine fault-tolerance is not yet instantiated in practice. Has the moment for a wide deployment of BFT systems arrived, and if so, where BFT systems should be deployed in the first place?

Format

The workshop will consist of invited contributions. No published proceedings, the presentations may contain results that appeared or are going to appear elsewhere, work-in-progress reports, surveys and tutorials. A submission is expected to be a short (around two pages) abstract of the presentation.

UNIVERSITÄT BERN

BFTW^3: Why? When? Where? Workshop on Theory and Practice of Byzantine Fault Tolerance

Affiliated with DISC 2009

September 22, 2009
Elche, Spain

However, there are few visible instantiations of these results in practical systems. Industrial software tends to ignore the BFT-related research and heads for less consistent but (apparently) simpler and more efficient solutions (e.g., [5, 16, 11]).

In this workshop, we discussed the state of the art in BFT systems, and tried to understand why BFT systems have not seen a widespread adoption, and what we could do to increase the chances of deploying BFT systems.

History 2 - Consensus protocols

- **1985 until 2000:** Theory research – many theorems, no systems, no prototypes
- **2000 until 2010:** Systems research – many prototypes, no products
- **2010 onward:** Practice – deployment with cryptocurrencies
- **Today:** More theory research, more systems research, products, and deployments

Consensus protocols today

Overview

- Nine *models* of blockchain consensus
- Explore *generalized trust*
- Explore *asymmetric trust*
- Answer your questions

Nine models of consensus

1 – Threshold trust (BFT)

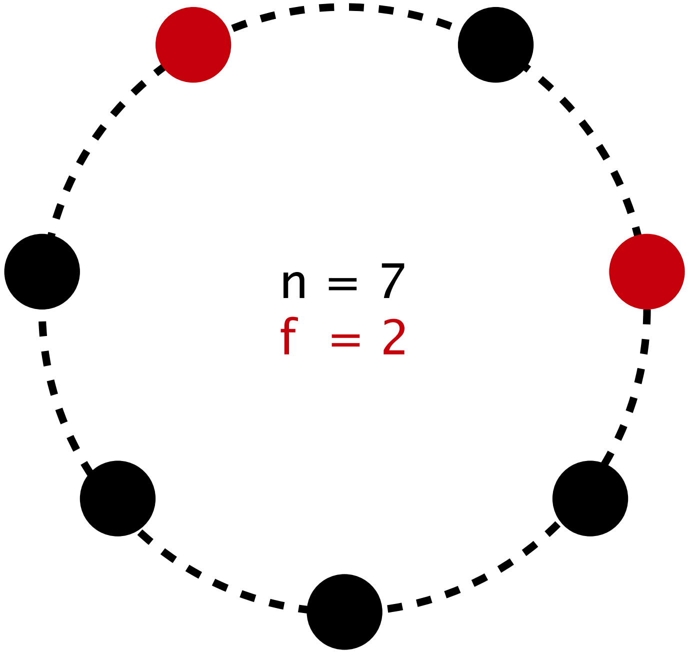
- Trust by numbers
  - n nodes total
  - f faulty (Byzantine) nodes

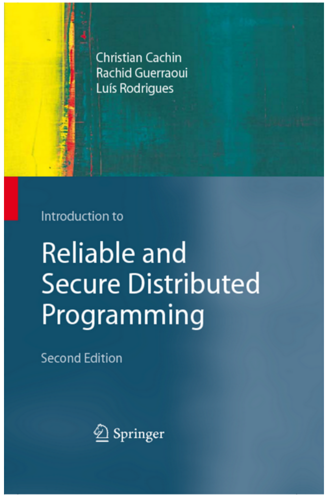
- Nodes are identified
  - Proof-of-Authority (PoA)

- Homogeneous and symmetric

- Requires $n > 3f$

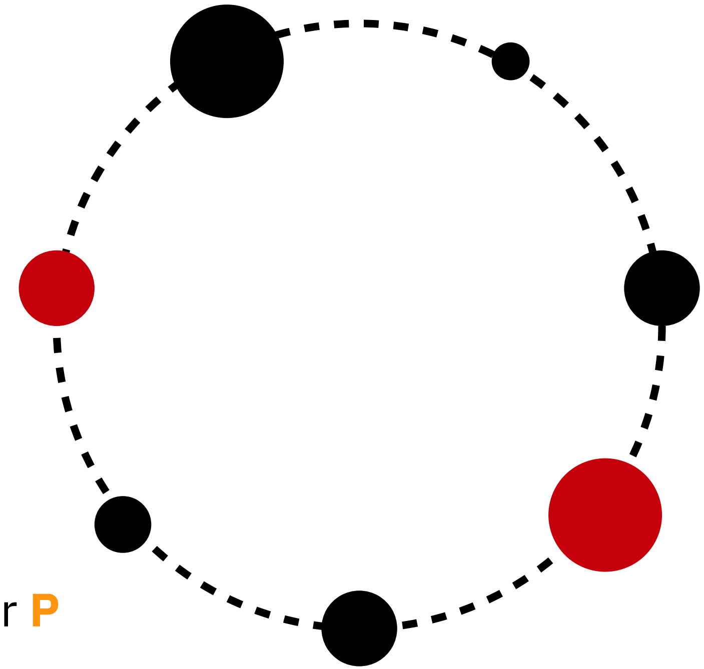
- Tendermint/Cosmos, Internet Computer (DFINITY), Hyperledger Fabric, VeChain, BNB SC, Hashgraph, TRON ...





2 – Generalized trust

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



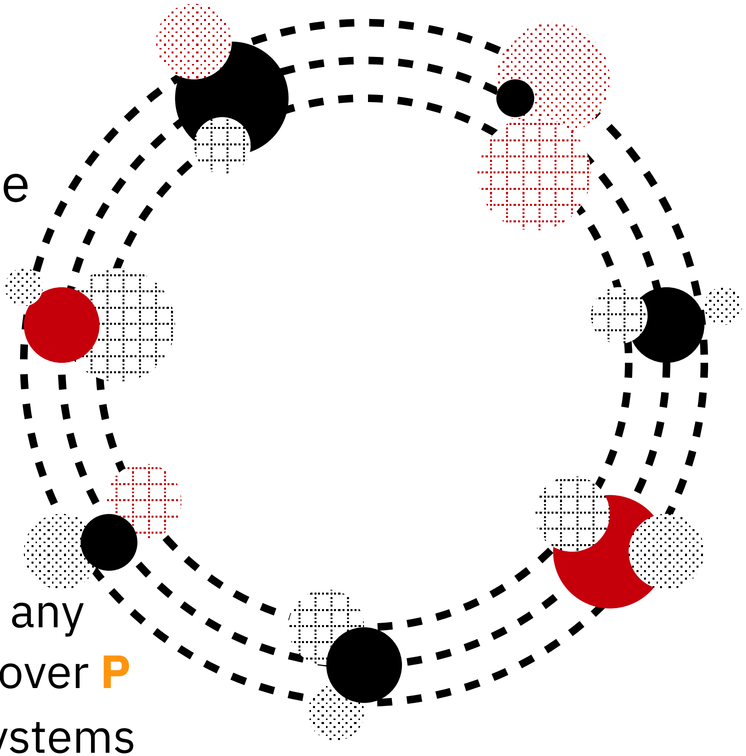
- Heterogeneous and symmetric

- Requires Q3-property
  - Any 3 fail-prone sets must not cover **P**

- Not used for consensus in any cryptocurrency (!)
  - But in distributed cryptography (Coinbase Vault)

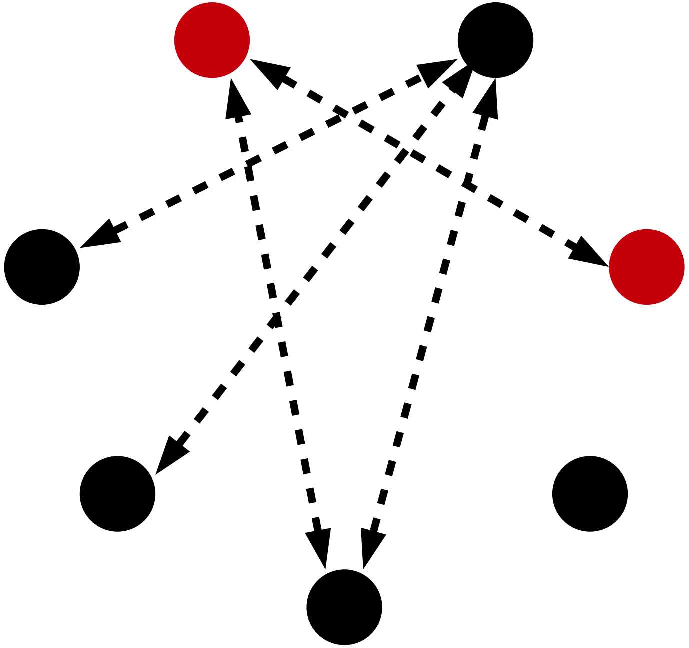
3 – Asymmetric trust

- Subjective generalized quorums
- Every node has its own fail-prone sets and 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, [ACTZ24]



4 – Unstructured, probabilistic voting

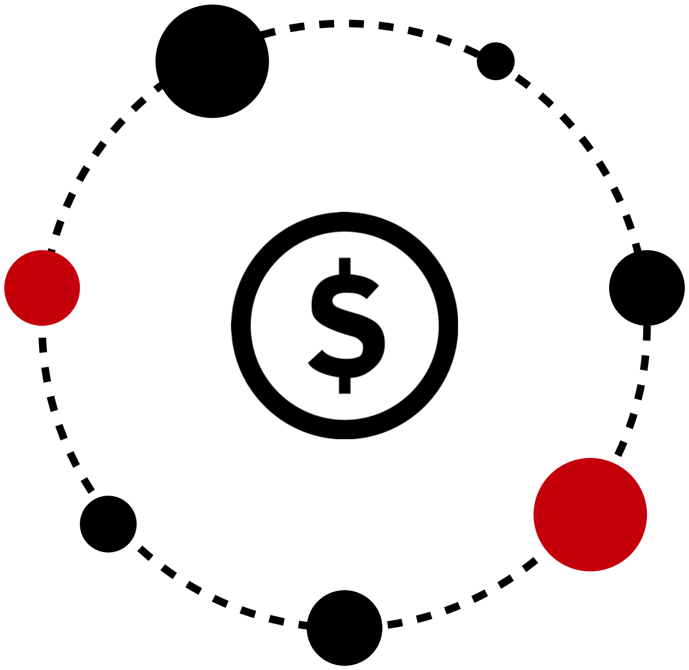
- Random sampling of peers
- Exchange information and votes
- Often coupled with a DAG (directed acyclic graph) on transactions
- Avalanche, Conflux, IOTA-Tangle



5 – Stake-based voting

UNIVERSITÄT BERN

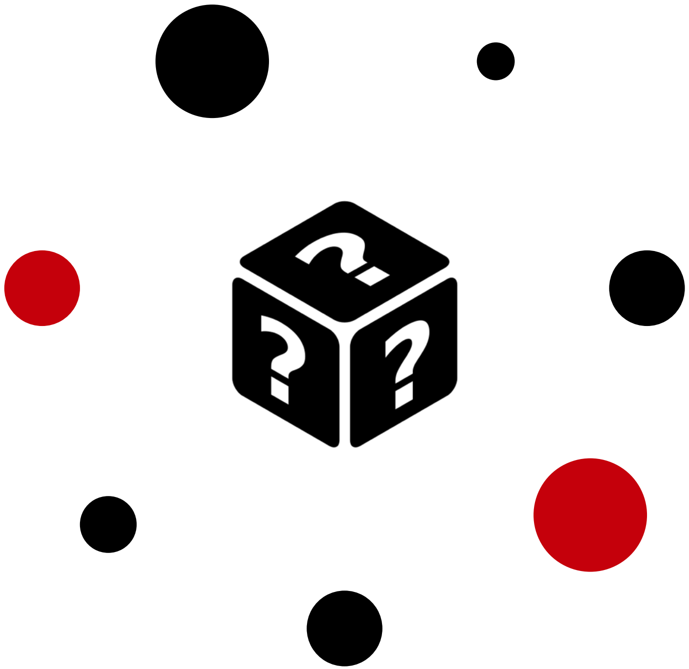
- Stake determines voting power
  - Including delegated stake (DPoS)
- Protocols generalized from symmetric voting (BFT)
- Slashing of invested stake upon detection of misbehavior
- Tendermint/Cosmos, EOS, NEO, Aptos, SUI, BNB SC ...



6 – Stake-based probabilistic choice

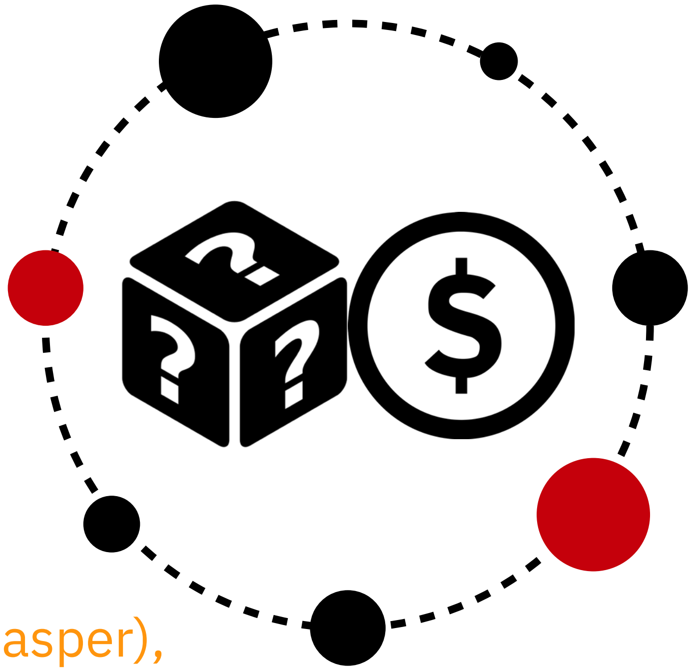
UNIVERSITÄT BERN

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



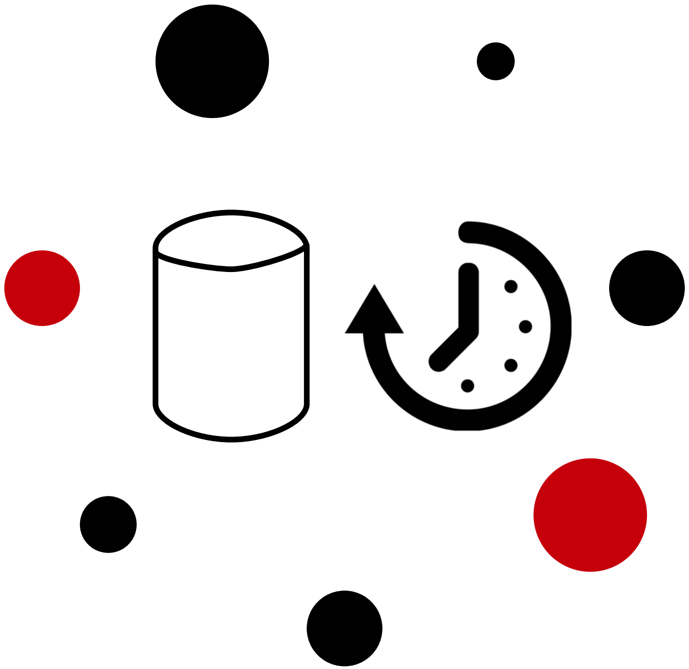
7 – Hybrid prob. choice and stake voting

- 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 ...



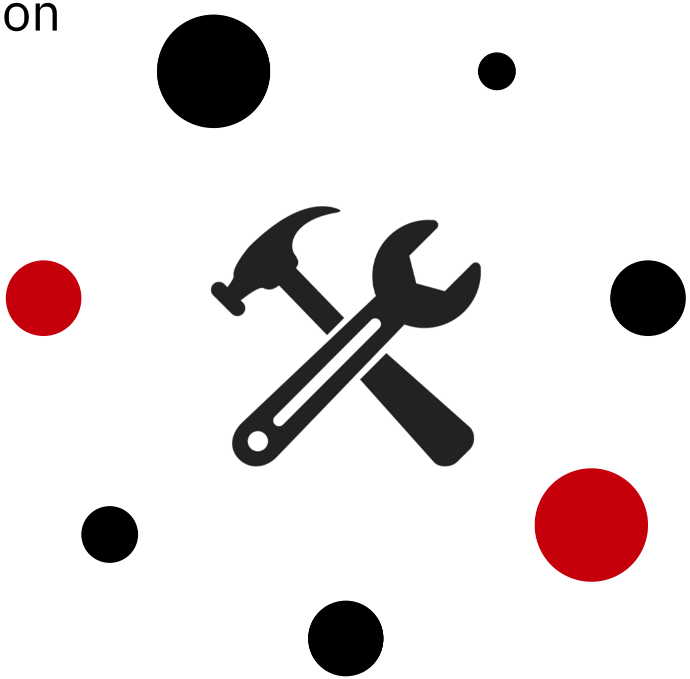
8 – Proof-of-space and proof-of-delay

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

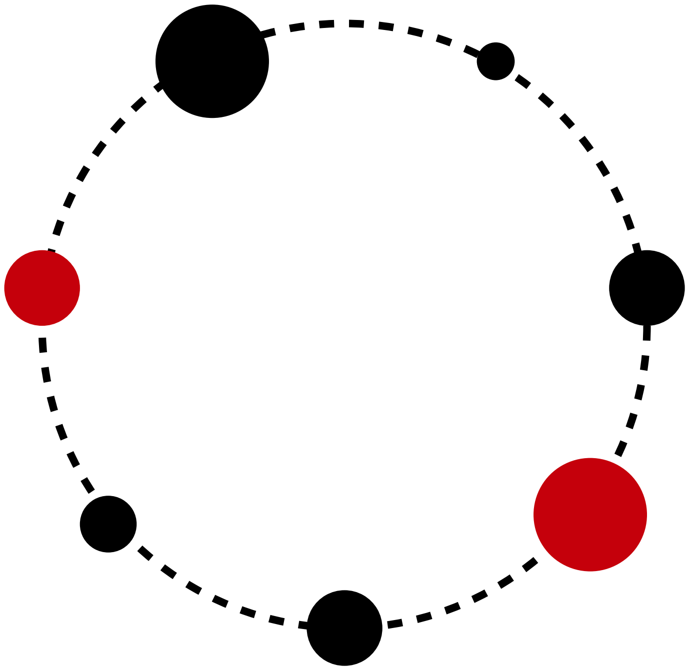


9 – Proof-of-work

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



Model 2 – Generalized trust



UNIVERSITÄT
BERN

Byzantine quorum systems

- Set of nodes $\mathbf{P} = \{p_1, \ldots, p_n\}$
- Fail-prone system $\mathbf{F} \subseteq 2^{\mathbf{P}}$: all nodes in some $F \in \mathbf{F}$ may fail together

- Quorum system $\mathbf{Q} \subseteq 2^{\mathbf{P}}$, where any $Q \in \mathbf{Q}$ is a "quorum", iff.
  - **Consistency:**

$$\forall\ Q_1, Q_2 \in \mathbf{Q}, \forall F \in \mathbf{F} : Q_1 \cap Q_2 \not\subseteq F.$$

  - **Availability:**

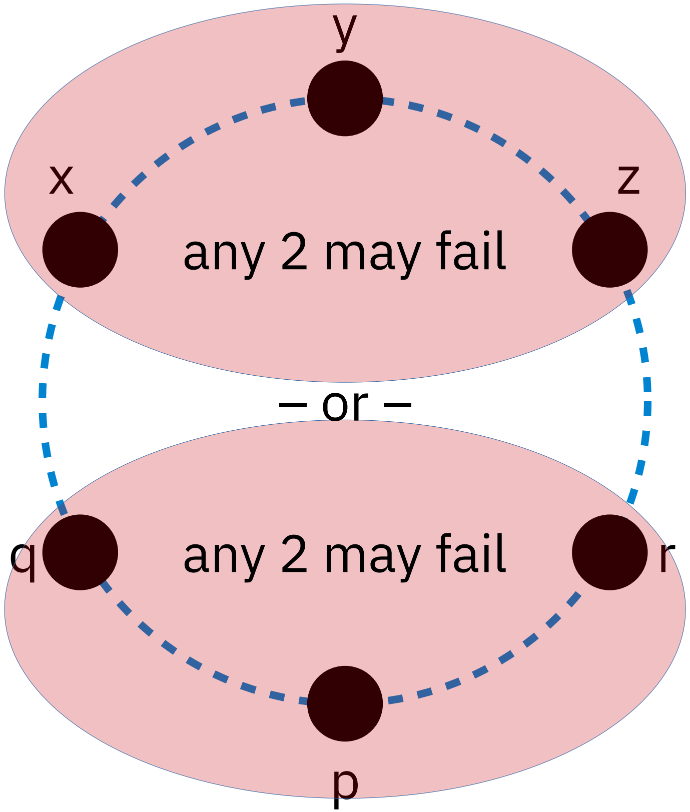
$$\forall F \in \mathbf{F} : \exists Q \in \mathbf{Q} : F \cap Q = \emptyset.$$

[Malkhi & Reiter, 1998]

- Symmetric trust

Generalized trust – Byz. quorum systems

- Set of nodes **P** = {p_1, ..., p_n}
- Fail-prone system **F** $\subseteq 2^{\mathbf{P}}$
  – All F ∈ **F** may fail together
- Quorum system **Q** $\subseteq 2^{\mathbf{P}}$, 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**



Generalized trust

- **Consensus and distributed cryptography** beyond the threshold model

- Theoretically well-known, practically not explored for consensus

- Example threshold cryptosystem: CoreKMS Vault at Coinbase for wallet keys (Lindell, SBC 2024)

  - Quorum = Offline AND Human-Approvers AND MPC-Servers
    - Offline = One Offline party
    - Human-Approvers = M Approvers, any monotone access structure (AND, OR, threshold), e.g., 5-of-20
    - MPC-Servers: K Servers, K-of-K needed

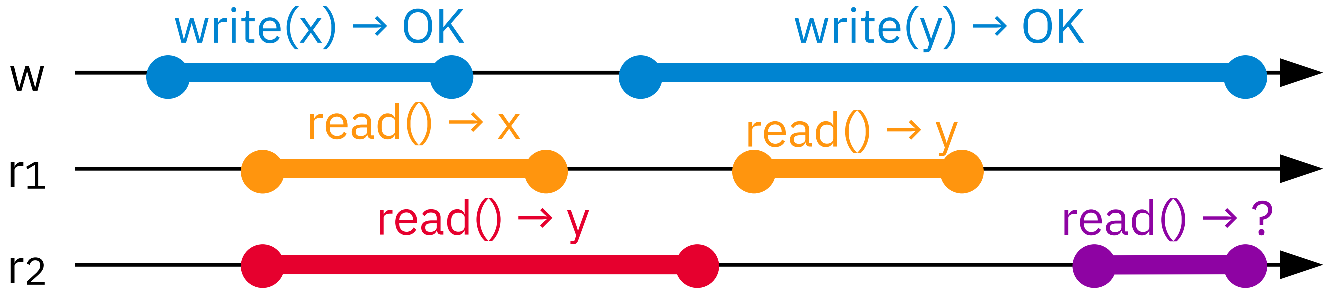
Example for generalized trust: RW Register

- A RW-register has two operations
  - Write(x) → OK
  - Read() → x
- Asynchronous (no clocks, no bounds on delays)
- Every operation defined by two events
  - Invocation (IN)
  - Completion (OUT)
- Simplification with a single-reader, single-writer (SRSW) register
  - Only process w ("Whit") or p_w may write
  - Only process r ("Ron") or p_r may read

Concurrency

- Operation o precedes o' whenever completion of o occurs before invocation of o'
- Otherwise, o and o' are concurrent
- How should the RW register behave when accessed concurrently?

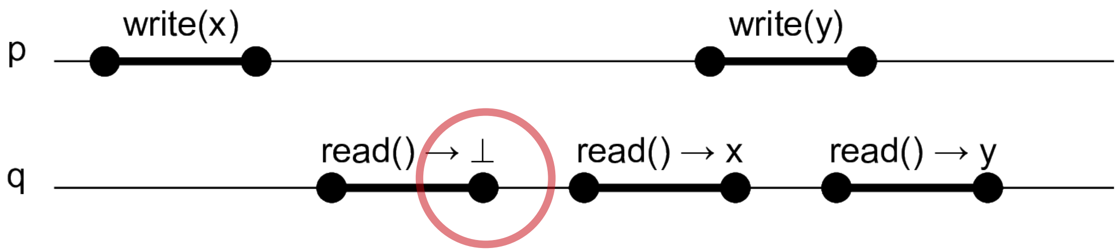
Semantics of MRSW register operations



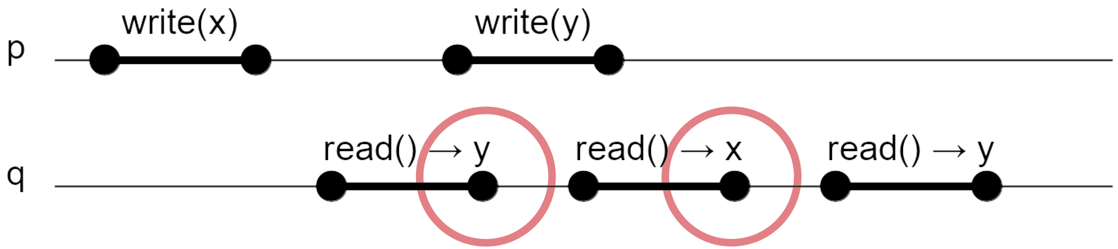
- Safe – Every read not concurrent with a write returns the most recently written value.
- Regular – Safe & any read concurrent with a write returns either the most recently written value or the concurrently written value: r2 may read() → x or y
- Atomic – Regular & all read and write operations occur atomically (= linearizable): r2 must read() → y

Example executions of SRSW register

- Not regular



- Regular



SRSW regular register

- Protocol with quorum system $\mathbf{Q} \subseteq 2^{\mathbf{P}}$ on nodes $\mathbf{P}$
- Fail-prone system $\mathbf{F} \subseteq 2^{\mathbf{P}}$ (any $F \in \mathbf{F}$ may fail together)

- Writer p_w maintains a logical timestamp ts
  - Increments ts for each write() operation
  - Issues digital signature s on pair (ts, v)
  - Sends timestamp/value/signature tuples (ts, v, s) to replica nodes
  - Waits for a quorum $Q \in \mathbf{Q}$ of replicas to acknowledge ("Byzantine quorum")

- Reader p_r asks replicas for their current (ts, v, s) tuples
  - Verifies that signature s from p_w is valid
  - Receives such tuples from a quorum $Q \in \mathbf{Q}$ of replicas ("Byzantine quorum")
  - Extracts value v with highest timestamp ts and returns v

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

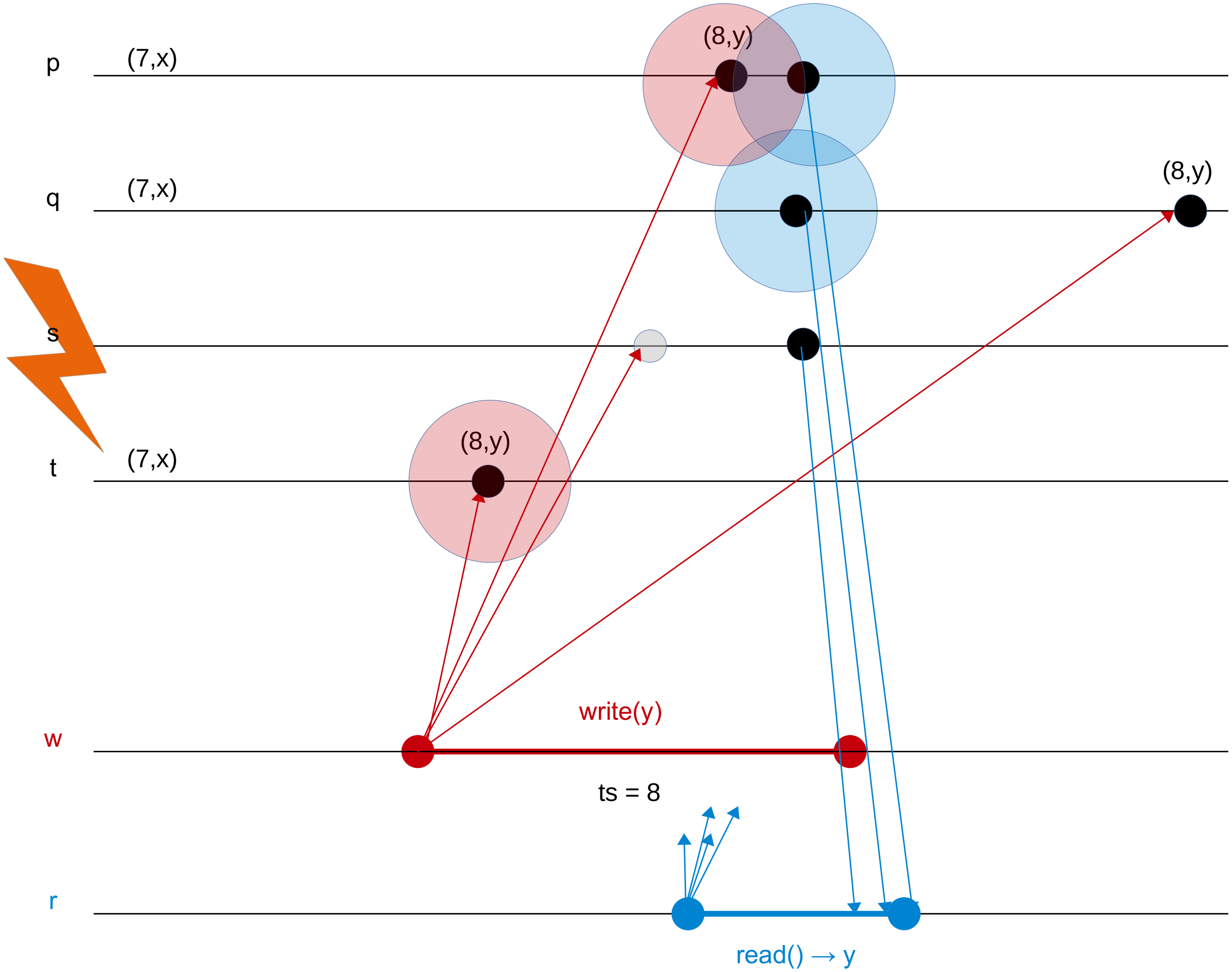
upon invocation *write*(v) **do** // only if p_i is writer p_w
 $wts \leftarrow wts + 1$
 $\sigma \leftarrow_\$ sign_w(\text{WRITE}\|w\|wts\|v)$
 send message $[\text{WRITE}, wts, v, \sigma]$ to all $p_j \in \mathcal{P}$
 wait for receiving a message $[\text{ACK}]$ from *more* than $\frac{n+f}{2}$ processes

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

Example SRSW register execution



Threshold crypto, generalized [AC23]

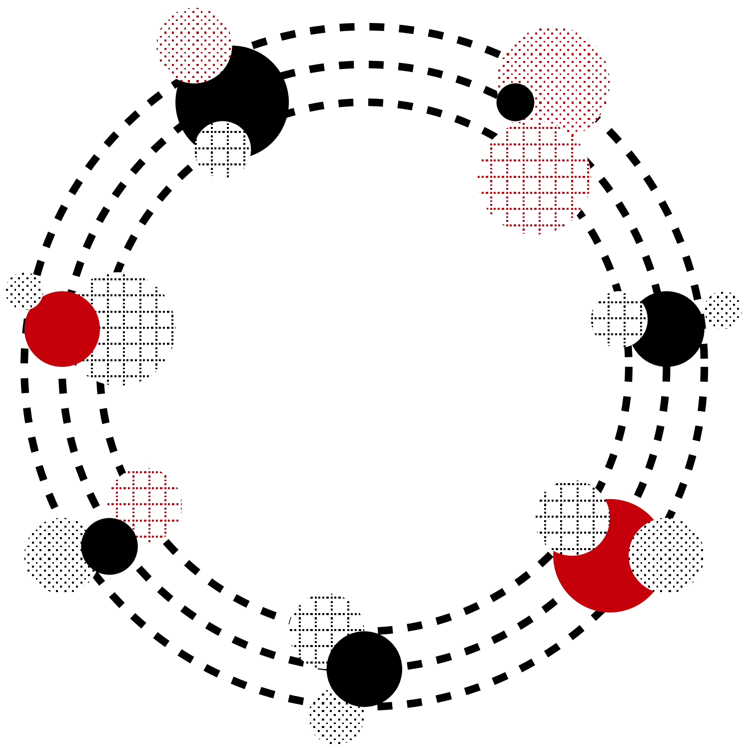
- Practical implementation of generalized distributed ("threshold") cryptosystems
  - Monotone span programs (MSP)

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

- Tools to generate MSP from a configuration file

- Benchmarks show the approach is practical

Model 3 – Asymmetric trust



Why asymmetric trust?

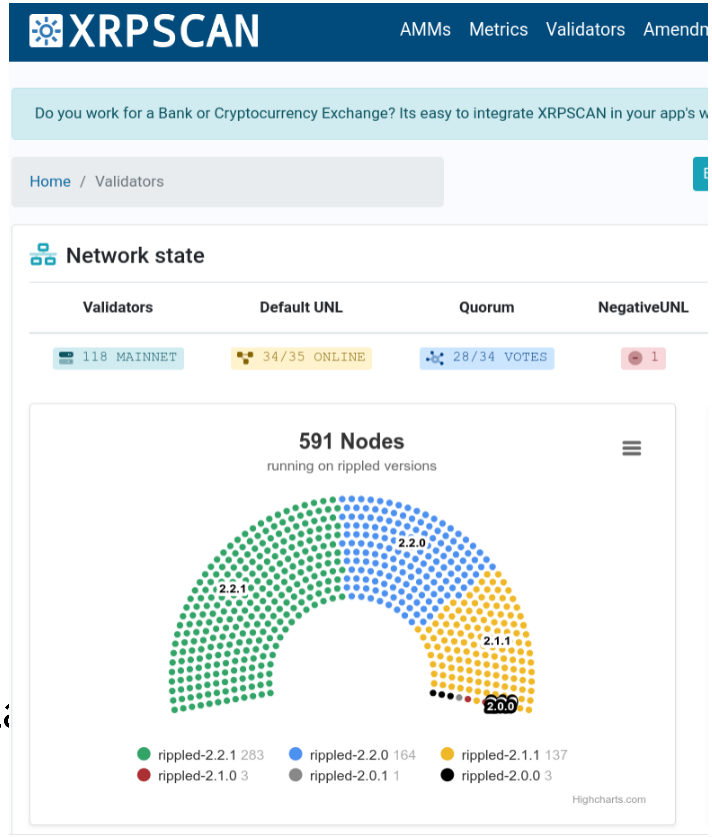
- For Romans:
  - *De gustibus non est disputandum.*
    (One cannot argue about taste.)

- 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)*

Quorums in XRPL / Ripple

- XRP started in 2012, among top by market cap
- Independently developed consensus protocol
  - Pseudocode and formal analysis later [ACM21]
- Each node declares which other nodes it trusts
  - Unique Node List (UNL)
  - UNLs of two nodes must overlap
- Default UNL (35 nodes)
  - Published by XRP Ledger Foundation (formerly: Ripple La
- Every node may choose its own UNL (in principle)



Federated quorums in Stellar

UNIVERSITÄT BERN

- Stellar (XLM) started as a fork of XRPL/Ripple

- Federated Byz. consensus in 2015 (Mazieres)
  - Every node may specify its own quorum sets

- Ex. SDF1 validator

```
{  "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"]}
]}
```

Asymmetric Byzantine quorum systems

- Asymmetric fail-prone system $\mathbf{F} = [\mathbf{F}_1, \ldots, \mathbf{F}_n]$, where $\mathbf{F}_i \subseteq 2^{\mathbf{P}}$ is the fail-prone system for p_i; all nodes in some $F \in \mathbf{F}_i$ may fail together (... according to p_i)

- Asymmetric quorum system $\mathbf{Q} = [\mathbf{Q}_1, \ldots, \mathbf{Q}_n]$, where $\mathbf{Q}_i \subseteq 2^{\mathbf{P}}$ is the quorum system for p_i and any $Q_i \in \mathbf{Q}_i$ is a "quorum for p_i", iff.

  – Consistency:

$$\forall\, p_i, p_j, \forall\, Q_i \in \mathbf{Q}_i, \forall\, Q_j \in \mathbf{Q}_j, \forall\, F \in \mathbf{F}_i^* \cap \mathbf{F}_j^* : Q_i \cap Q_j \not\subseteq F.$$

  – Availability:

$$\forall\, p_i, \forall\, F \in \mathbf{F}_i : \exists\, Q \in \mathbf{Q}_i : F \cap Q = \emptyset.$$

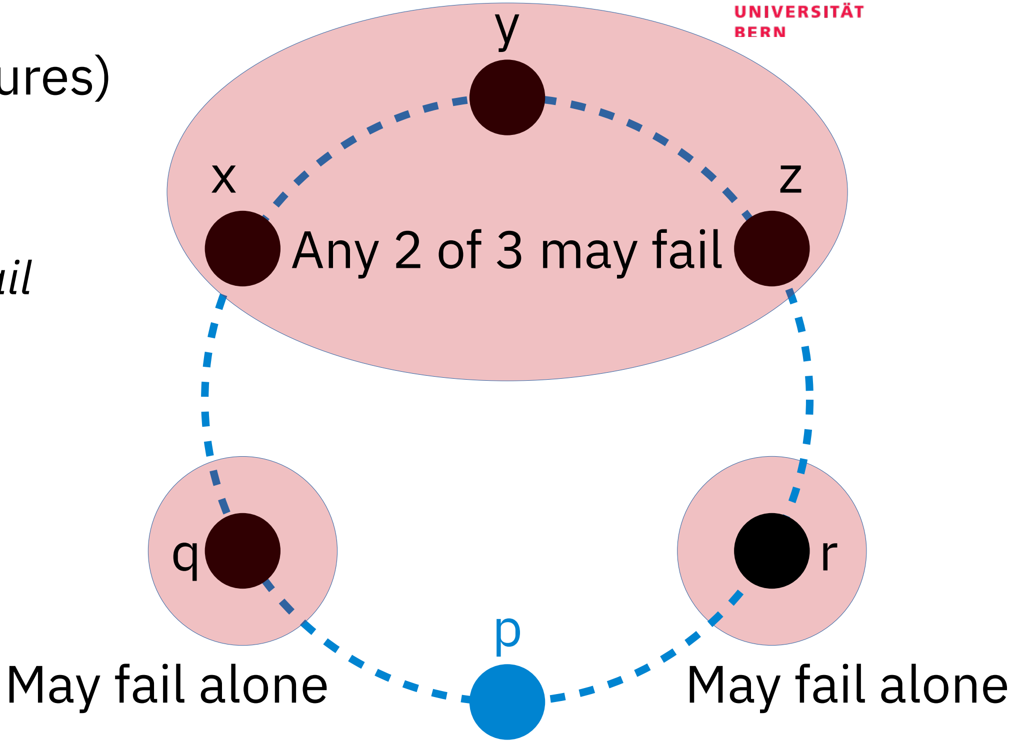
(Based on Damgård, Desmedt, Fitzi, Nielsen, Asiacrypt 2007)

Asymmetric trust [ACTZ24]

UNIVERSITÄT BERN

- Subjective trust assumption of p (via failures)
  - p itself *never* fails
  - Neighbor nodes q and r: *May only fail alone*
  - Remote nodes x, y, x: *Any 2 of these 3 may fail*
- Fail-prone system **F** ⊆ $2^{\mathbf{P}}$ of p

  {q, r, xy, yz, xz}
  - All F ∈ **F** may fail for p
- Each one of the 6 nodes uses its own subjective trust like this → Asymmetric quorums
- Nodes are trusted **differently**.



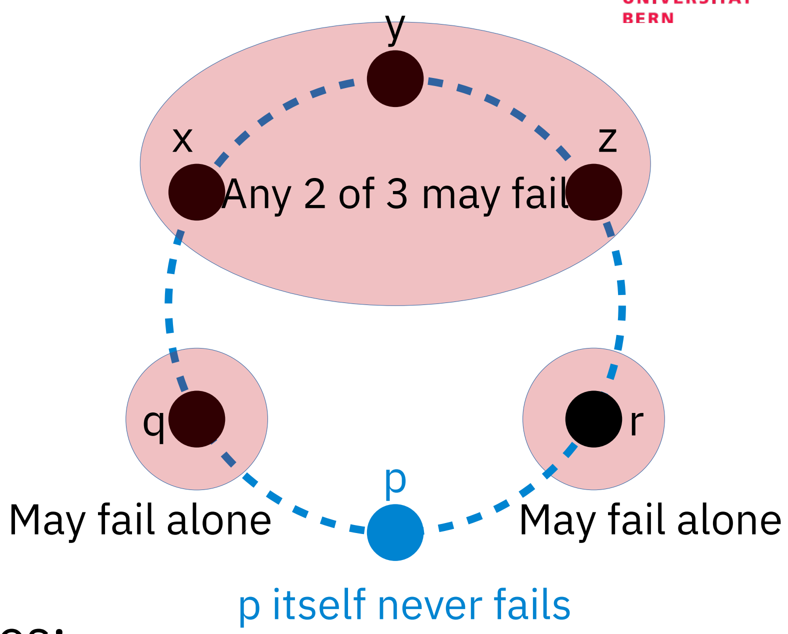
p itself never fails
Nodes trust **differently** (asymmetric).

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 [WRITE, wts, v, σ] to all $p_j \in \mathcal{P}$
  wait for receiving a message [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 [READ, rid] to all $p_j \in \mathcal{P}$
  wait for receiving messages [VALUE, r_j, ts_j, v_j, σ_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 [WRITE, ts', v', σ'] from p_w **do** // every process
  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** // every process
  send message [VALUE, r, ts, v, σ] 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

Protocols with asymmetric quorums

- RW-Register emulations
- Byzantine consistent and reliable broadcasts
- Randomized binary consensus
- Related work
  - Flexible consensus
  - Heterogeneous Paxos

Conclusion

UNIVERSITÄT BERN

- Byzantine-tolerant consensus protocols matter and are here to stay
- Assumptions are more important than protocols

Conclusion

- Byzantine-tolerant consensus protocols matter and are here to stay
- Assumptions are more important than protocols

- Generalized trust
- Asymmetric trust

- Links
  - Web: https://crypto.unibe.ch/
  - Blog: https://cryptobern.github.io/
  - Twitter/X: https://x.com/cczurich/

Thanks

- This work has been supported by
  - Swiss National Science Foundation (SNSF);
  - Donation from Stellar Development Foundation; and a
  - Sui Academic Research Award.

- Links
  - Web: https://crypto.unibe.ch/
  - Blog: https://cryptobern.github.io/
  - Twitter/X: https://x.com/cczurich/

b UNIVERSITÄT BERN

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

- 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

UNIVERSITÄT BERN

- Model 3: Asymmetric trust

  - [ACTZ24] Alpos, O., Cachin, C., Tackmann, B., & Zanolini, L. (2024). Asymmetric Distributed Trust. Distributed Computing, 37(3), 247-277. https://doi.org/10.1007/s00446-024-00469-1
  - [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
  - [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 3: Asymmetric trus

Literature

UNIVERSITÄT BERN

- 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. In Y. Emek (Ed.), Proc. Structural Information and Communication Complexity (SIROCCO) (Vol. 14662, pp. 27–44). Springer. https://doi.org/10.1007/978-3-031-60603-8_2

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

UNIVERSITÄT BERN

- 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