Introduction to Distributed Consensus

ECS 189F: Fall 2020

Slides are adopted from Gupta, Hellings, Rahnama, Sadoghi.

Building High Throughput Permissioned Blockchain Fabrics: Challenges and Opportunities, VLDB'20

Exploratory Systems Lab at UC Davis

Mission: *To pioneer a resilient data platform at scale, a distributed ledger centered around a democratic and decentralized computational model (ResilientDB Fabric) that further aims to unify secure transactional and real-time analytical processing (L-Store).*

- 1 Postdoc, 3 Ph.D. students, 7 M.Sc. and B.Sc. students.
- Recent papers at VLDB, ICDE, ICDCS, ICDT, DISC, EDBT, Middleware and more.
- Crossroad of distributed databases and blockchains .

Goal: Pioneering Resilient Data Platform at Scale.

Questions

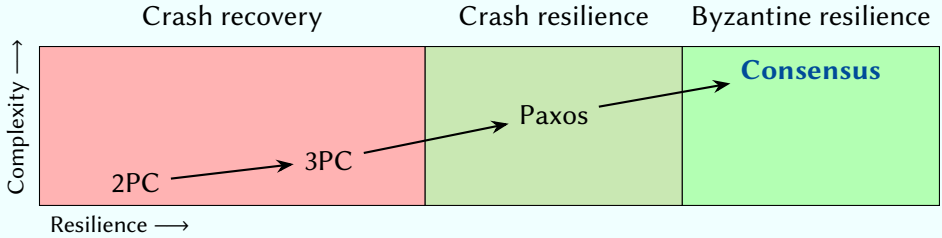
1. Why?
2. What is the relation with blockchains?
3. What do we already have?
4. Where can we improve?
5. What new tools do we need?

Towards high-performance resilient data processing:

Why?

Why resilient data processing?

Go beyond assumptions of traditional transaction processing!

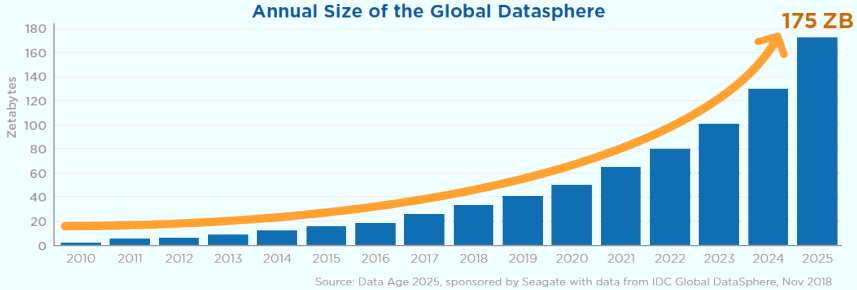


Example

- Provide continuous services during failures.
- Provide services in federated environments.

Why high-performance?

Support requirements of future applications!



- Ever-growing volumes of data (e.g., sensor networks).
- Ever-growing demands of applications (e.g., machine learning).

Towards high-performance resilient data processing:

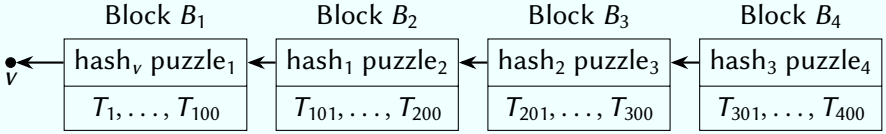
What is the relation with blockchains?

What is a blockchain?

What is a blockchain?

Bitcoin: Management of monetary tokens (Bitcoins)

- Open and decentralized transfer of tokens (*transactions*).
- History of transactions (*ledger*) stored in the blockchain.

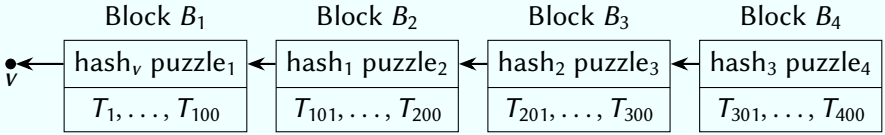


- *Many participants* hold a copy of the blockchain.
- Blockchain structure is *tamper-proof* by design.

What is a blockchain? - Malicious behavior

Bitcoin: Preventing malicious behavior

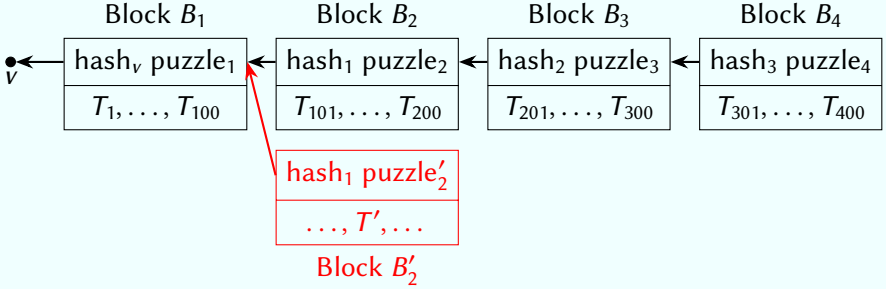
- Malicious attempts to change a chain.



What is a blockchain? - Malicious behavior

Bitcoin: Preventing malicious behavior

- Malicious attempts to change a chain.



- Longest chain has highest incentives.
- Making blocks (solving puzzles) is very costly.
- Malicious attempt leads to a *dead end*.

What is a blockchain? - A definition

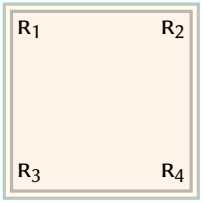
A resilient tamper-proof ledger maintained by many participants.

- *Ledger*.
  Append-only sequence of transactions.
  In database terms: a journal or log.
- *Resilient*.
  High availability via full replication among participants.
- *Tamper-proof*.
  Changes can only be made with majority participation.

Blockchains are *distributed fully-replicated systems*!

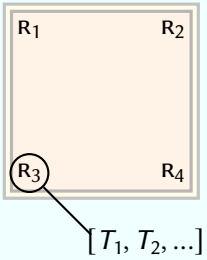
Components of blockchain systems

1. Replicas.



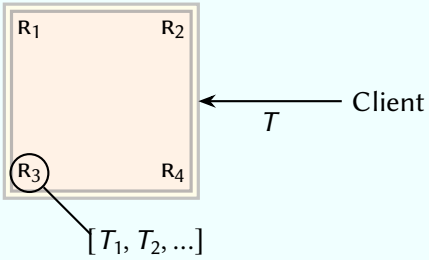
Components of blockchain systems

1. Replicas.
2. Holding the ledger of transactions.



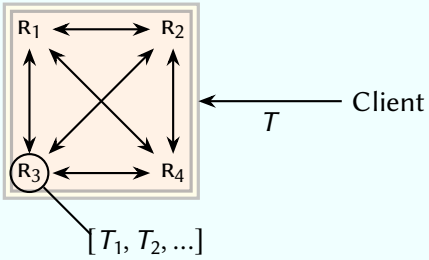
Components of blockchain systems

1. Replicas.
2. Holding the ledger of transactions.
3. Clients with new transactions.



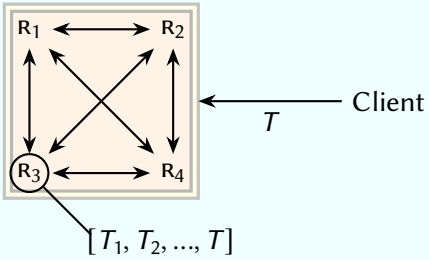
Components of blockchain systems

1. Replicas.
2. Holding the ledger of transactions.
3. Clients with new transactions.
4. Transaction agreement via consensus.



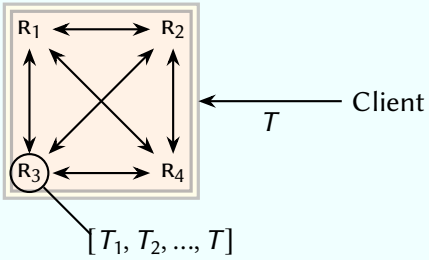
Components of blockchain systems

1. Replicas.
2. Holding the ledger of transactions.
3. Clients with new transactions.
4. Transaction agreement via consensus.
5. Append-only updates to ledger.



Components of blockchain systems

1. Replicas.
2. Holding the ledger of transactions.
3. Clients with new transactions.
4. Transaction agreement via consensus.
5. Append-only updates to ledger.
6. Cryptography.



Bitcoin: A permissionless blockchain

The participants are not known and can change.

Rationale: Fully decentralized and open cryptocurrencies

- Bitcoin, Ethereum, ...
- Scale to thousands of participants.
- Low transaction processing throughput.
- Very high transaction latencies.

We focus on permissioned blockchains

All participants are known.

Rationale: Data processing in managed environment

- Support different attack models than cryptocurrencies.
- Easier to support low latencies and high throughputs.
- Downside: changing participants is hard.

Many ideas also apply to permissionless blockchains.

Towards high-performance resilient data processing:

What do we already have?

We have consensus: PBFT, PAXOS, PoW, ...

Termination Each non-faulty replica decides on a transaction.

Non-divergence Non-faulty replicas decide on the same transaction.

We have consensus: PBFT, PAXOS, PoW, ...

Termination	Each non-faulty replica decides on a transaction.
Non-divergence	Non-faulty replicas decide on the same transaction.
Validity	Every decided-on transaction is a client request.
Response	Clients learn about the outcome of their requests.
Service	Every client will be able to request transactions.

We have consensus: PBFT, PAXOS, PoW, ...

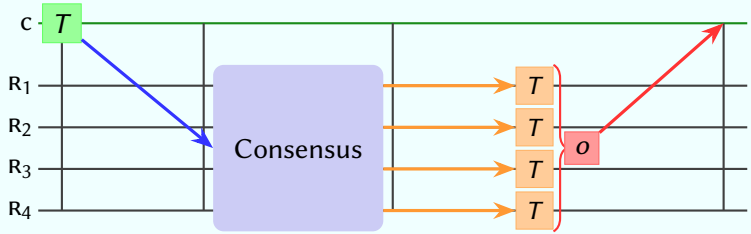
Termination Each non-faulty replica decides on a transaction.

Non-divergence Non-faulty replicas decide on the same transaction.

Validity Every decided-on transaction is a client request.

Response Clients learn about the outcome of their requests.

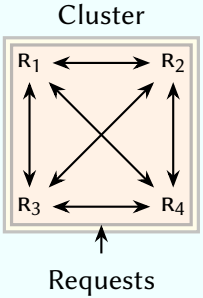
Service Every client will be able to request transactions.



Operating a fully-replicated ledger using consensus

Each replica maintains a copy of the ledger:
Append-only sequence of transactions.

1. Use consensus to select the ρ-th client request T.
2. Append T as the ρ-th entry to the ledger.
3. Execute T as the ρ-th entry, inform client.



Consistent state: Linearizable order and deterministic execution

On identical inputs, execution of transactions at all non-faulty replicas *must produce identical outputs*.

Distributed fully-replicated systems: The CAP Theorem

Consistency Does every participant have exactly the same data?

Availability Does the system continuously provide services?

Partitioning Can the system cope with network disturbances?

Theorem (The CAP Theorem)

Can provide at most two-out-of-three of these properties.

Distributed fully-replicated systems: The CAP Theorem

Consistency Does every participant have exactly the same data?

Availability Does the system continuously provide services?

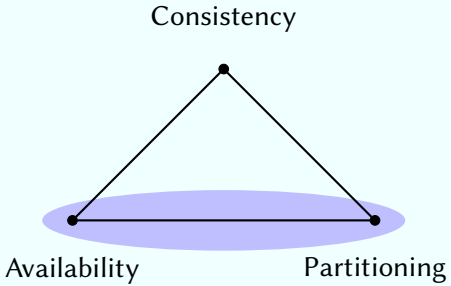
Partitioning Can the system cope with network disturbances?

Theorem (The CAP Theorem)

Can provide at most two-out-of-three of these properties.

CAP Theorem uses narrow definitions!

The CAP Theorem and Blockchains

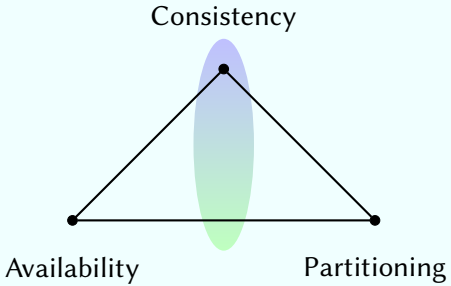


Permissionless Blockchains

Open membership focuses on Availability and Partitioning.

$\Longrightarrow$ Consistency not guaranteed (e.g., forks).

The CAP Theorem and Blockchains



Permissioned Blockchains

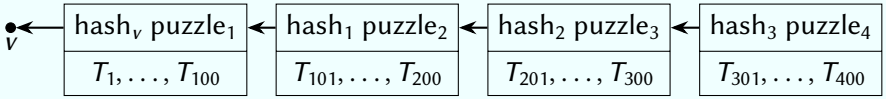
Consistency at all costs.

$\Longrightarrow$ Availability when communication is reliable.

$\Longrightarrow$ Partition-tolerance when network failure is limited and replicas are reliable.

What else do we have?

- A lot of *theory* on consensus: consensus is costly.
- PBFT: A practical Byzantine fault-tolerant consensus protocol.
- Tamper-proof *ledgers*.

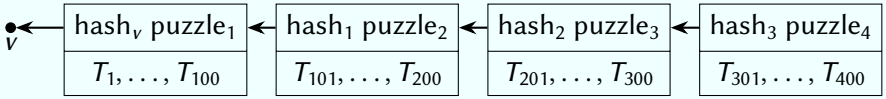


Exact details: depend on consensus, application, attack model, ...

- Many *cryptographic tools*.

What else do we have?

- A lot of *theory* on consensus: consensus is costly.
- PBFT: A practical Byzantine fault-tolerant consensus protocol.
- Tamper-proof *ledgers*.



Exact details: depend on consensus, application, attack model, ...

- Many *cryptographic tools*.

What about high-performance?

Theory on consensus: Summary

Limitations of practical consensus

- No asynchronous communication!
- Dealing with $\mathbf{f}$ malicious failures requires $\mathbf{n} > 3\mathbf{f}$ replicas.
- Worst-case: at least $\Omega(\mathbf{f} + 1)$ phases of communication.
- Worst-case: at least $\Omega(\mathbf{nf})$ signatures and $\Omega(\mathbf{n} + \mathbf{f}^2)$ messages.
- Network must stay connected when removing $2\mathbf{f}$ replicas.

Consensus in practice

Asynchronous communication, $\mathbf{n} > 3\mathbf{f}$, clique network:

$\Longrightarrow$ termination only when communication is reliable.

Towards high-performance resilient data processing:

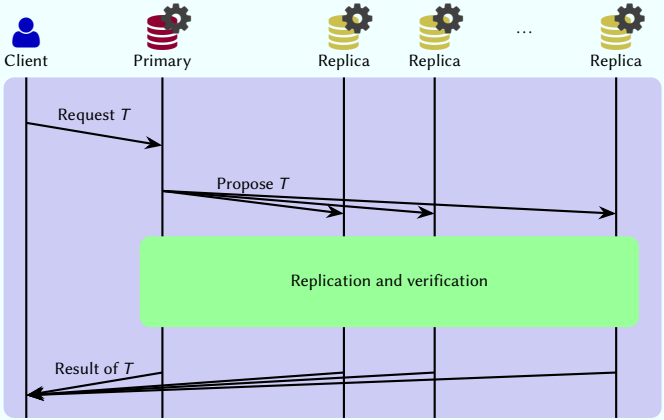
What do we already have?

PBFT

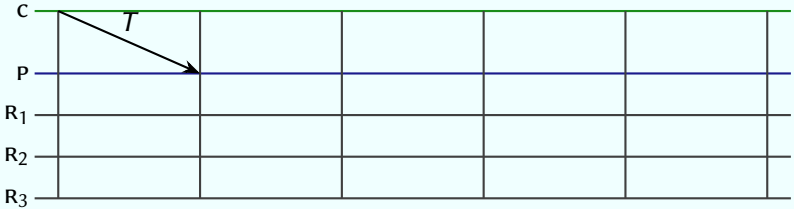
PBFT: Practical Byzantine Fault Tolerance

Primary Coordinates consensus: propose transactions to replicate.

Backup Accept transactions and verifies behavior of primary.

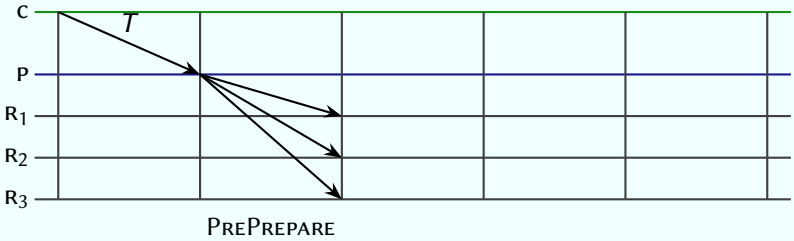


PBFT: Normal-case protocol in view v



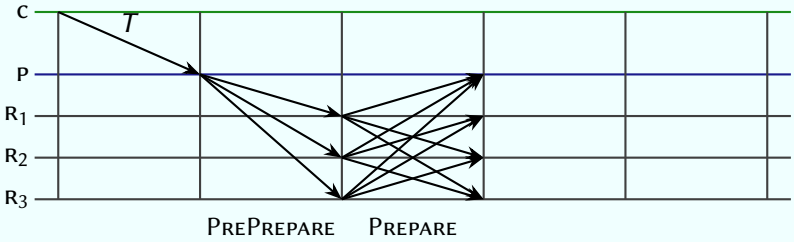
$\langle T \rangle_c$.

PBFT: Normal-case protocol in view v



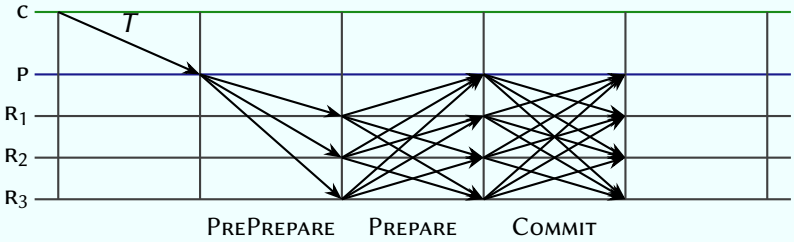
$\textsc{PrePrepare}(\langle T \rangle_c, v, \rho)$.

PBFT: Normal-case protocol in view v



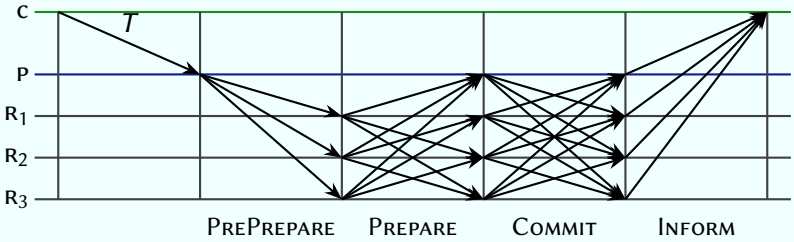
If receive PrePrepare message m: Prepare(m).

PBFT: Normal-case protocol in view v



If $\mathbf{n} - \mathbf{f}$ identical Prepare(m) messages: Commit(m).

PBFT: Normal-case protocol in view v



If $\mathbf{n} - \mathbf{f}$ identical Commit(m) messages: execute, Inform($\langle T \rangle_c, \rho, r$).

PBFT: Normal-case consensus

Theorem

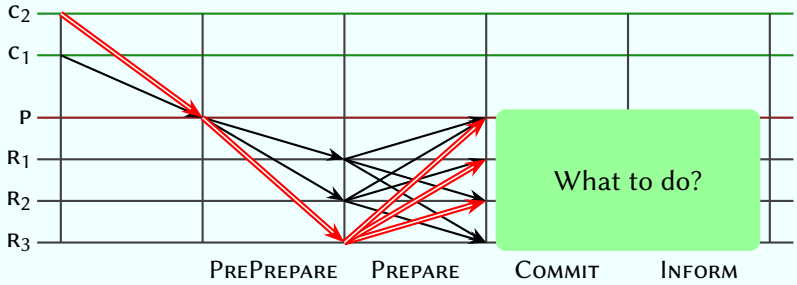
If the primary is non-faulty and communication is reliable, then the normal-case of PBFT *ensures consensus on T in round ρ.*

PBFT: Normal-case consensus

Theorem

If the primary is non-faulty and communication is reliable, then the normal-case of PBFT *ensures consensus on T in round ρ.*

Example (Byzantine primary, $\mathbf{n} = 4$, $\mathbf{f} = 1$, $\mathbf{n} - \mathbf{f} = 3$)

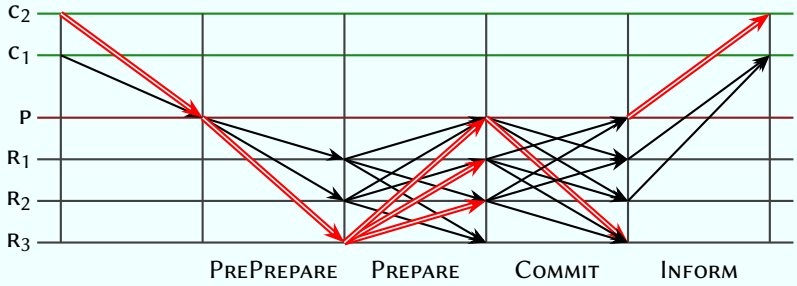


PBFT: Normal-case consensus

Theorem

If the primary is non-faulty and communication is reliable, then the normal-case of PBFT *ensures consensus on T in round ρ.*

Example (Byzantine primary, $\mathbf{n} = 4$, $\mathbf{f} = 1$, $\mathbf{n} - \mathbf{f} = 3$)



PBFT: A normal-case property when $\mathbf{n} > 3\mathbf{f}$

Theorem (Castro et al.)

If replicas R_i, $i \in \{1, 2\}$, *commit to* $m_i = \textsc{PrePrepare}(\langle T_i \rangle_{\mathrm{c}_i}, v, \rho)$, *then* $\langle T_1 \rangle_{\mathrm{c}_1} = \langle T_2 \rangle_{\mathrm{c}_2}$.

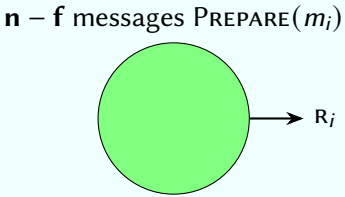
PBFT: A normal-case property when $\mathbf{n} > 3\mathbf{f}$

Theorem (Castro et al.)

If replicas R_i, $i \in \{1, 2\}$, *commit to* $m_i = \textsc{PrePrepare}(\langle T_i \rangle_{c_i}, v, \rho)$, *then* $\langle T_1 \rangle_{c_1} = \langle T_2 \rangle_{c_2}$.

Proof.

Replica R_i commits to m_i:



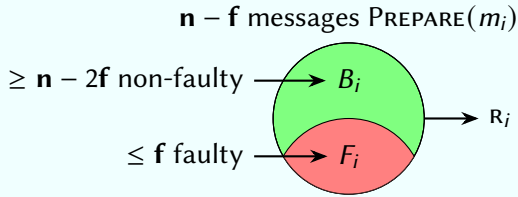
PBFT: A normal-case property when $\mathbf{n} > 3\mathbf{f}$

Theorem (Castro et al.)

If replicas R_i, $i \in \{1, 2\}$, *commit to* $m_i = \textsc{PrePrepare}(\langle T_i \rangle_{c_i}, v, \rho)$, *then* $\langle T_1 \rangle_{c_1} = \langle T_2 \rangle_{c_2}$.

Proof.

Replica R_i commits to m_i:



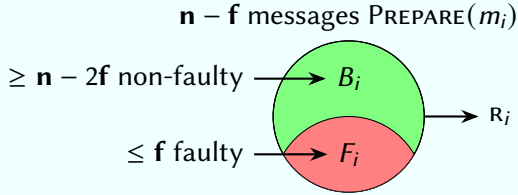
PBFT: A normal-case property when $\mathbf{n} > 3\mathbf{f}$

Theorem (Castro et al.)

If replicas R_i, $i \in \{1, 2\}$, *commit to* $m_i = \textsc{PrePrepare}(\langle T_i \rangle_{c_i}, v, \rho)$, *then* $\langle T_1 \rangle_{c_1} = \langle T_2 \rangle_{c_2}$.

Proof.

Replica R_i commits to m_i:



If $\langle T_1 \rangle_{c_1} \neq \langle T_2 \rangle_{c_2}$, then $B_1 \cap B_2 = \emptyset$ and $|B_1 \cup B_2| \geq 2(\mathbf{n} - 2\mathbf{f})$.

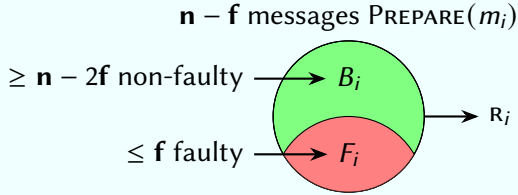
PBFT: A normal-case property when $\mathbf{n} > 3\mathbf{f}$

Theorem (Castro et al.)

If replicas R_i, $i \in \{1, 2\}$, *commit to* $m_i = \textsc{PrePrepare}(\langle T_i \rangle_{c_i}, v, \rho)$, *then* $\langle T_1 \rangle_{c_1} = \langle T_2 \rangle_{c_2}$.

Proof.

Replica R_i commits to m_i:



If $\langle T_1 \rangle_{c_1} \neq \langle T_2 \rangle_{c_2}$, then $B_1 \cap B_2 = \emptyset$ and $|B_1 \cup B_2| \geq 2(\mathbf{n} - 2\mathbf{f})$.

$$2(\mathbf{n} - 2\mathbf{f}) \leq \mathbf{n} - \mathbf{f}$$

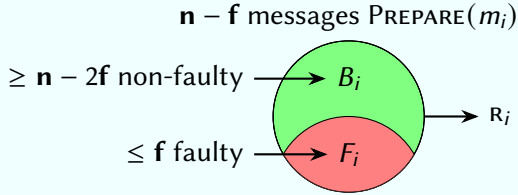
PBFT: A normal-case property when $\mathbf{n} > 3\mathbf{f}$

Theorem (Castro et al.)

If replicas R_i, $i \in \{1, 2\}$, *commit to* $m_i = \textsc{PrePrepare}(\langle T_i \rangle_{c_i}, v, \rho)$, *then* $\langle T_1 \rangle_{c_1} = \langle T_2 \rangle_{c_2}$.

Proof.

Replica R_i commits to m_i:



If $\langle T_1 \rangle_{c_1} \neq \langle T_2 \rangle_{c_2}$, then $B_1 \cap B_2 = \emptyset$ and $|B_1 \cup B_2| \geq 2(\mathbf{n} - 2\mathbf{f})$.

$$2(\mathbf{n} - 2\mathbf{f}) \leq \mathbf{n} - \mathbf{f} \qquad \text{iff} \qquad 2\mathbf{n} - 4\mathbf{f} \leq \mathbf{n} - \mathbf{f}$$

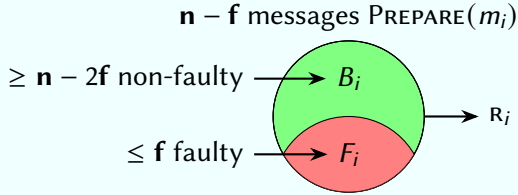
PBFT: A normal-case property when $\mathbf{n} > 3\mathbf{f}$

Theorem (Castro et al.)

If replicas R_i, $i \in \{1, 2\}$, *commit to* $m_i = \textsc{PrePrepare}(\langle T_i \rangle_{\mathrm{c}_i}, v, \rho)$, *then* $\langle T_1 \rangle_{\mathrm{c}_1} = \langle T_2 \rangle_{\mathrm{c}_2}$.

Proof.

Replica R_i commits to m_i:

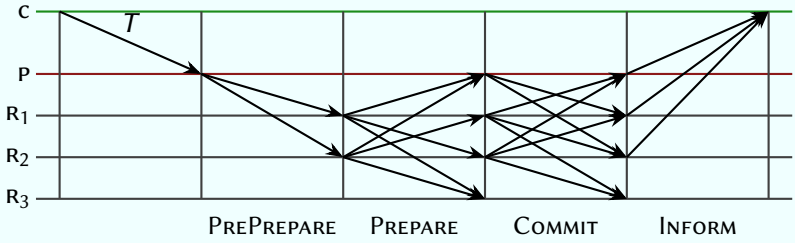


If $\langle T_1 \rangle_{\mathrm{c}_1} \neq \langle T_2 \rangle_{\mathrm{c}_2}$, then $B_1 \cap B_2 = \emptyset$ and $|B_1 \cup B_2| \geq 2(\mathbf{n} - 2\mathbf{f})$.

$$2(\mathbf{n} - 2\mathbf{f}) \leq \mathbf{n} - \mathbf{f} \quad \text{iff} \quad 2\mathbf{n} - 4\mathbf{f} \leq \mathbf{n} - \mathbf{f} \quad \text{iff} \quad \mathbf{n} \leq 3\mathbf{f}. \qquad \square$$

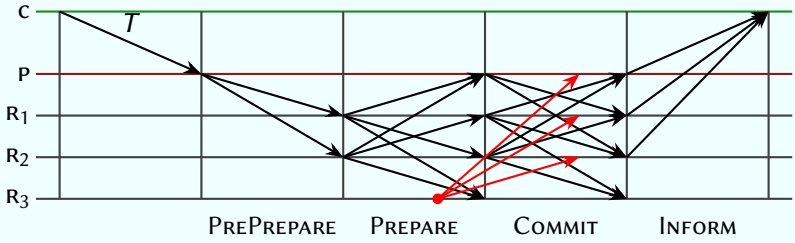
PBFT: Primary failure versus malicious replicas

Primary P is faulty
ignores R_3



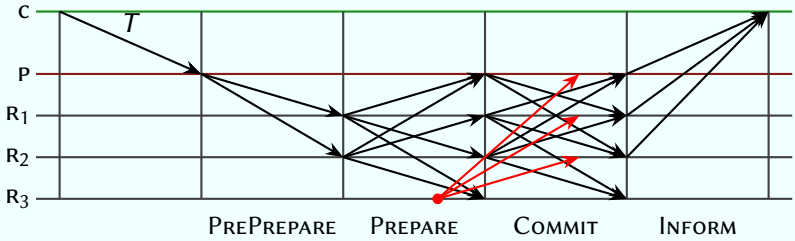
PBFT: Primary failure versus malicious replicas

Primary P is faulty
ignores R_3

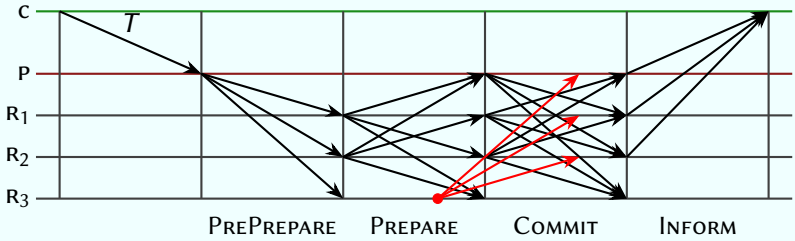


PBFT: Primary failure versus malicious replicas

Primary P is faulty
ignores R_3

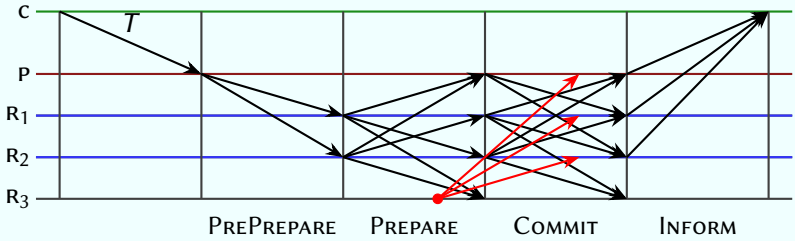


Replica R_3 is malicious
pretends to be ignored

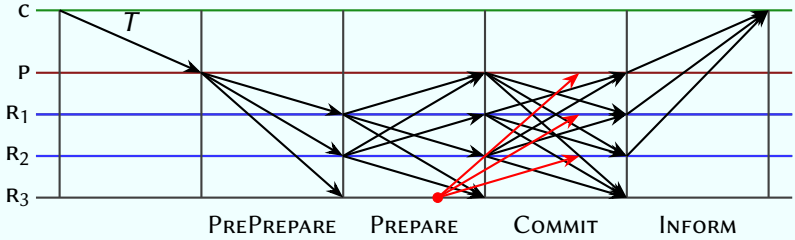


PBFT: Primary failure versus malicious replicas

Primary P is faulty
ignores R_3



Replica R_3 is malicious
pretends to be ignored



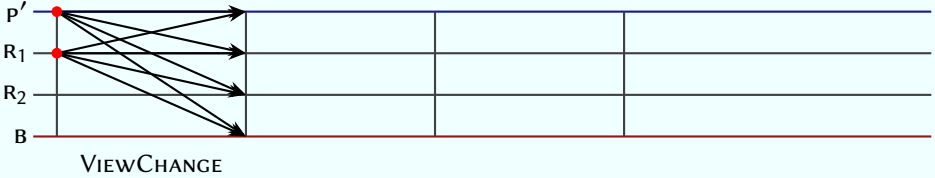
PBFT: Detectable primary failures

If the primary behaves faulty to $> \mathbf{f}$ non-faulty replicas, then failure of the primary is detectable.

Replacing the primary: View-change at replica R

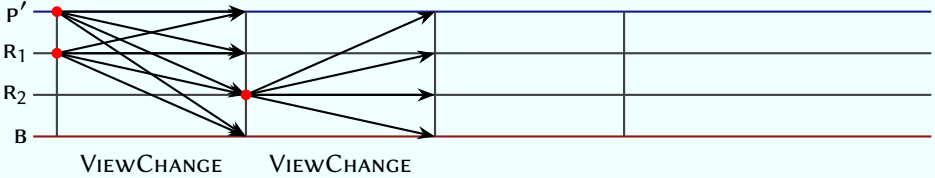
1. R detects *failure* of the current primary P.
2. R chooses a new primary P′ (the next replica).
3. R provides P′ with its *current state*.
4. P′ proposes a *new view*.
5. If the new view is valid, then R switches to this view.

PBFT: A view-change in view v



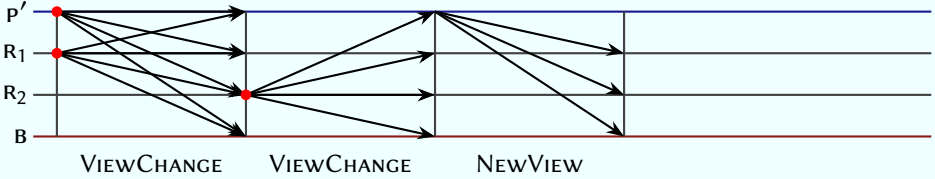
Send ViewChange(E, v) with E all prepared transactions.

PBFT: A view-change in view v



Indirect failure detection by R_2.

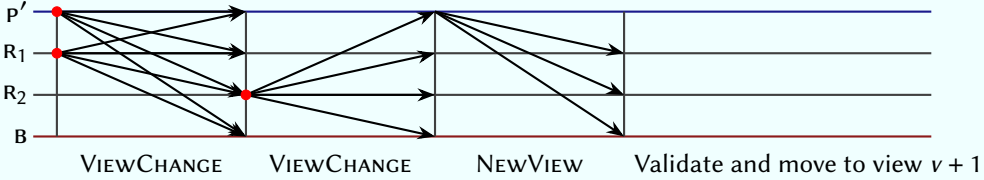
PBFT: A view-change in view v



If $\mathbf{n} - \mathbf{f}$ valid ViewChange(E, v) messages: NewView$(v + 1, \mathcal{E}, \mathcal{N})$.

- $\mathcal{E}$ contains $\mathbf{n} - \mathbf{f}$ valid ViewChange messages.
- $\mathcal{N}$ contains no-op proposals for *missing rounds*.

PBFT: A view-change in view v



Move to view $v+1$ if NewView$(v+1, \mathcal{E}, \mathcal{N})$ is valid.

- $\mathcal{E}$ contains $\mathbf{n} - \mathbf{f}$ valid ViewChange messages.
- $\mathcal{N}$ contains no-op proposals for *missing rounds*.

PBFT: A property of view-changes when $\mathbf{n} > 3\mathbf{f}$

Theorem (Castro et al.)

Let NEWVIEW$(v', \mathcal{E}, \mathcal{N})$ *be a well-formed* NEWVIEW *message.*
If a set S *of* $\mathbf{n} - 2\mathbf{f}$ *non-faulty replicas committed to* m *in view* $v < v'$*,*
then $\mathcal{E}$ *contains a* VIEWCHANGE *message preparing* m*.*

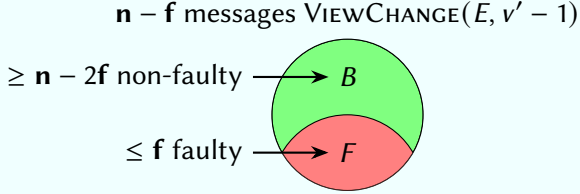
PBFT: A property of view-changes when $\mathbf{n} > 3\mathbf{f}$

Theorem (Castro et al.)

Let NewView$(v', \mathcal{E}, \mathcal{N})$ *be a well-formed* NewView *message.*
If a set S *of* $\mathbf{n} - 2\mathbf{f}$ *non-faulty replicas committed to* m *in view* $v < v'$*,*
then $\mathcal{E}$ *contains a* ViewChange *message preparing* m*.*

Proof.

The ViewChange messages in $\mathcal{E}$:



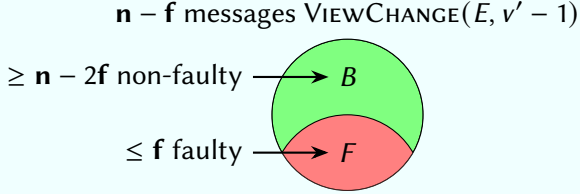
PBFT: A property of view-changes when $\mathbf{n} > 3\mathbf{f}$

Theorem (Castro et al.)

Let NewView$(v', \mathcal{E}, \mathcal{N})$ *be a well-formed* NewView *message.*
If a set S *of* $\mathbf{n} - 2\mathbf{f}$ *non-faulty replicas committed to* m *in view* $v < v'$*,*
then $\mathcal{E}$ *contains a* ViewChange *message preparing* m*.*

Proof.

The ViewChange messages in $\mathcal{E}$:



if $S \cap B = \emptyset$, then $|S \cup B| \geq 2(\mathbf{n} - 2\mathbf{f})$, a contradiction! □

Towards high-performance resilient data processing:

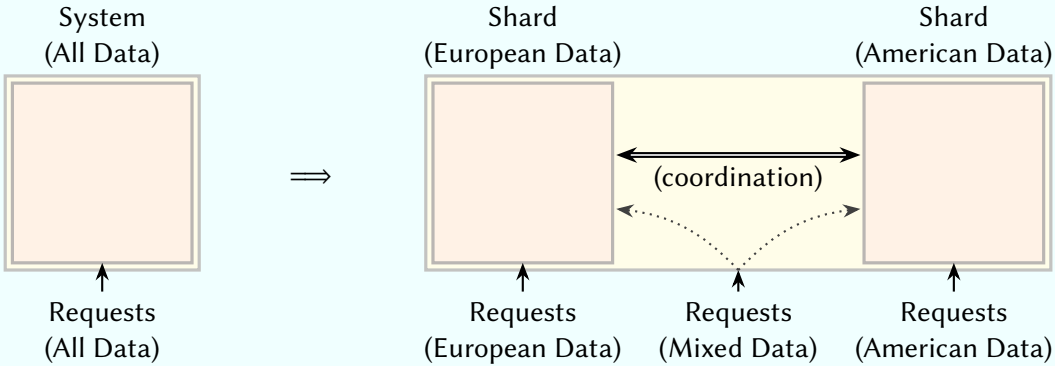
Where can we improve?

A look at high-performance data processing

Scalability: adding resources $\Longrightarrow$ adding performance.

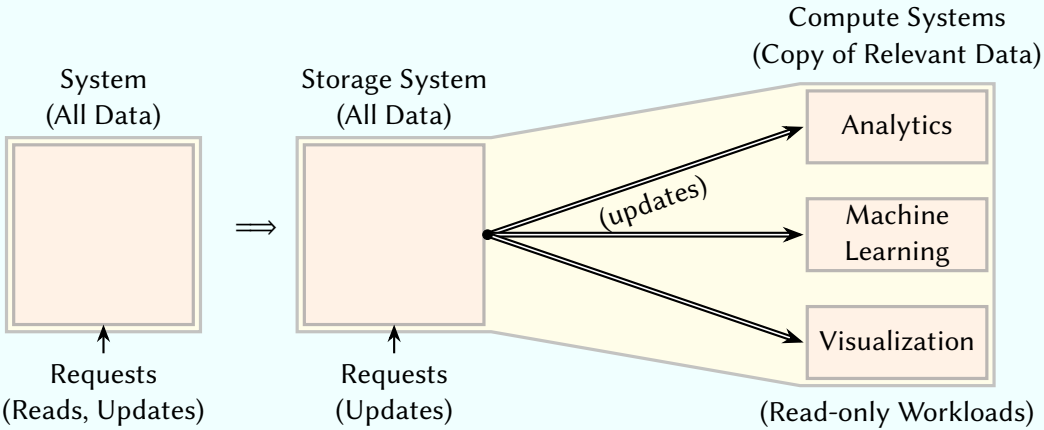
Full replication: adding resources (replicas) $\Longrightarrow$ less performance!

Sharding and Geo-scale aware sharding



Adding shards ⟹ adding throughput (parallel processing), adding storage.

Role Specialization: Read-only workloads



Specializing roles ⟹ adding throughput (parallel processing, specialized hardware, …).

Towards high-performance resilient data processing:

What new tools do we need?

Central ideas for improvement

Reminder

We can make a resilient cluster that manages data: *blockchains*.

- **Sharding**: make each shard an independent blockchain.
  Requires: *reliable communication between blockchains.*
  Permissionless blockchains: relays, atomic swaps!
- **Role Specialization**: make the storage system a blockchain.
  Requires: *reliable read-only updates of the blockchain.*
  Permissionless blockchains: light clients!

Consensus is of no use here if we want efficiency.

Towards high-performance resilient data processing:

Concluding remarks

Conclusion

We need an extensive toolbox!

	(permissioned)	(permissionless)
▶ Consensus	PBFT, Paxos… GeoBFT, RCC…	PoW, PoS, …
▶ Cross-blockchain communication	Cluster-sending… Cerberus…	Relays, atomic swaps
▶ Read-only participation	Byzantine learning	Light clients

High-performance resilient data processing is nearby.

Ongoing work

Initial results are available

- Cluster-sending: DISC 2019, doi: 10.4230/LIPIcs.DISC.2019.45.
- Byzantine learning: ICDT 2020, doi: 10.4230/LIPIcs.ICDT.2020.17.
- Geo-aware consensus: VLDB 2020, doi: 10.14778/3380750.3380757.

More about us and our work

- Expolab Creativity Unfolded — https://expolab.org/.
- ResilientDB Security, Privacy Reloaded — https://resilientdb.com/.

References I

Ittai Abraham et al. *Synchronous Byzantine Agreement with Expected $O(1)$ Rounds, Expected $O(n^2)$ Communication, and Optimal Resilience*. 2018.

Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. "CAPER: A Cross-application Permissioned Blockchain". In: *Proceedings of the VLDB Endowment* 12.11 (2019), pp. 1385–1398. ISSN: 2150-8097. DOI: 10.14778/3342263.3342275.

Elli Androulaki et al. "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains". In: *Proceedings of the Thirteenth EuroSys Conference*. ACM, 2018, 30:1–30:15. DOI: 10.1145/3190508.3190538.

Pierre-Louis Aublin, Sonia Ben Mokhtar, and Vivien Quéma. "RBFT: Redundant Byzantine Fault Tolerance". In: *2013 IEEE 33rd International Conference on Distributed Computing Systems*. IEEE, 2013, pp. 297–306. DOI: 10.1109/ICDCS.2013.53.

Pierre-Louis Aublin et al. "The Next 700 BFT Protocols". In: *ACM Transactions on Computer Systems* 32.4 (2015), 12:1–12:45. DOI: 10.1145/2658994.

Eric Brewer. "CAP twelve years later: How the "rules" have changed". In: *Computer* 45.2 (2012), pp. 23–29. DOI: 10.1109/MC.2012.37.

References II

Eric A. Brewer. “Towards Robust Distributed Systems (Abstract)”. In: *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*. ACM, 2000, pp. 7–7. DOI: 10.1145/343477.343502.

Christian Cachin and Marko Vukolic. “Blockchain Consensus Protocols in the Wild (Keynote Talk)”. In: *31st International Symposium on Distributed Computing*. Vol. 91. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 1:1–1:16. DOI: 10.4230/LIPIcs.DISC.2017.1.

Michael Casey et al. *The Impact of Blockchain Technology on Finance: A Catalyst for Change*. Tech. rep. International Center for Monetary and Banking Studies, 2018. URL: https://www.cimb.ch/uploads/1/1/5/4/115414161/geneva21_1.pdf.

Miguel Castro and Barbara Liskov. “Practical Byzantine Fault Tolerance and Proactive Recovery”. In: *ACM Transactions on Computer Systems* 20.4 (2002), pp. 398–461. DOI: 10.1145/571637.571640.

Miguel Correia et al. “Low complexity Byzantine-resilient consensus”. In: *Distributed Computing* 17.3 (2005), pp. 237–249. DOI: 10.1007/s00446-004-0110-7.

Richard A. DeMillo, Nancy A. Lynch, and Michael J. Merritt. “Cryptographic Protocols”. In: *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*. ACM, 1982, pp. 383–400. DOI: 10.1145/800070.802214.

References III

Tien Tuan Anh Dinh et al. “Untangling Blockchain: A Data Processing View of Blockchain Systems”. In: *IEEE Transactions on Knowledge and Data Engineering* 30.7 (2018), pp. 1366–1385. DOI: 10.1109/TKDE.2017.2781227.

D. Dolev. “Unanimity in an unknown and unreliable environment”. In: *22nd Annual Symposium on Foundations of Computer Science*. IEEE, 1981, pp. 159–168. DOI: 10.1109/SFCS.1981.53.

D. Dolev and H. Strong. “Authenticated Algorithms for Byzantine Agreement”. In: *SIAM Journal on Computing* 12.4 (1983), pp. 656–666. DOI: 10.1137/0212045.

Danny Dolev. “The Byzantine generals strike again”. In: *Journal of Algorithms* 3.1 (1982), pp. 14–30. DOI: 10.1016/0196-6774(82)90004-9.

Danny Dolev, Cynthia Dwork, and Larry Stockmeyer. “On the Minimal Synchronism Needed for Distributed Consensus”. In: *Journal of the ACM* 34.1 (1987), pp. 77–97. DOI: 10.1145/7531.7533.

Danny Dolev and Rüdiger Reischuk. “Bounds on Information Exchange for Byzantine Agreement”. In: *Journal of the ACM* 32.1 (1985), pp. 191–204. DOI: 10.1145/2455.214112.

Partha Dutta, Rachid Guerraoui, and Marko Vukolic. *The Complexity of Asynchronous Byzantine Consensus*. Tech. rep. EPFL, 2004. URL: http://infoscience.epfl.ch/record/52690.

References IV

Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. “Consensus in the Presence of Partial Synchrony”. In: *Journal of the ACM* 35.2 (1988), pp. 288–323. DOI: 10.1145/42282.42283.

Paul Feldman and Silvio Micali. “Optimal Algorithms for Byzantine Agreement”. In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. ACM, 1988, pp. 148–161. DOI: 10.1145/62212.62225.

Michael J. Fischer and Nancy A. Lynch. “A lower bound for the time to assure interactive consistency”. In: *Information Processing Letters* 14.4 (1982), pp. 183–186. DOI: 10.1016/0020-0190(82)90033-3.

Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. “Impossibility of Distributed Consensus with One Faulty Process”. In: *Journal of the ACM* 32.2 (1985), pp. 374–382. DOI: 10.1145/3149.214121.

Juan A. Garay and Yoram Moses. “Fully Polynomial Byzantine Agreement for Processors in Rounds”. In: *SIAM Journal on Computing* 27.1 (1998), pp. 247–290. DOI: 10.1137/S0097539794265232.

Yossi Gilad et al. “Algorand: Scaling Byzantine Agreements for Cryptocurrencies”. In: *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 51–68. DOI: 10.1145/3132747.3132757.

References V

Seth Gilbert and Nancy Lynch. “Brewer’s Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services”. In: *SIGACT News* 33.2 (2002), pp. 51–59. DOI: 10.1145/564585.564601.

William J. Gordon and Christian Catalini. “Blockchain Technology for Healthcare: Facilitating the Transition to Patient-Driven Interoperability”. In: *Computational and Structural Biotechnology Journal* 16 (2018), pp. 224–230. DOI: 10.1016/j.csbj.2018.06.003.

Jim Gray. “Notes on Data Base Operating Systems”. In: *Operating Systems, An Advanced Course*. Springer-Verlag, 1978, pp. 393–481. DOI: 10.1007/3-540-08755-9_9.

Suyash Gupta, Jelle Hellings, and Mohammad Sadoghi. “Brief Announcement: Revisiting Consensus Protocols through Wait-Free Parallelization”. In: *33rd International Symposium on Distributed Computing (DISC 2019)*. Vol. 146. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 44:1–44:3. DOI: 10.4230/LIPIcs.DISC.2019.44.

Suyash Gupta, Jelle Hellings, and Mohammad Sadoghi. *Fault-Tolerant Distributed Transactions on Blockchains*. (to appear). 2020.

References VI

Suyash Gupta, Jelle Hellings, and Mohammad Sadoghi. “RCC: Resilient Concurrent Consensus for High-Throughput Secure Transaction Processing”. In: *37th International Conference on Data Engineering (ICDE)*. IEEE, 2021.

Suyash Gupta, Sajjad Rahnama, and Mohammad Sadoghi. “Permissioned Blockchain Through the Looking Glass: Architectural and Implementation Lessons Learned”. In: *Proceedings of the 40th International Conference on Distributed Computing Systems*. IEEE, 2020.

Suyash Gupta et al. “An In-Depth Look of BFT Consensus in Blockchain: Challenges and Opportunities”. In: *Proceedings of the 20th International Middleware Conference Tutorials*. ACM, 2019, pp. 6–10. DOI: 10.1145/3366625.3369437.

Suyash Gupta et al. “ResilientDB: Global Scale Resilient Blockchain Fabric”. In: *Proceedings of the VLDB Endowment* 13.6 (2020), pp. 868–883. DOI: 10.14778/3380750.3380757.

Suyash Gupta et al. “Tutorial: Blockchain Consensus Unraveled: Virtues and Limitations”. In: *Proceedings of the 14th ACM International Conference on Distributed and Event-based Systems*. ACM, 2020. DOI: 10.1145/3401025.3404099.

References VII

Jelle Hellings and Mohammad Sadoghi. “Brief Announcement: The Fault-Tolerant Cluster-Sending Problem”. In: *33rd International Symposium on Distributed Computing (DISC 2019)*. Vol. 146. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 45:1–45:3. DOI: 10.4230/LIPIcs.DISC.2019.45.

Jelle Hellings and Mohammad Sadoghi. “Coordination-Free Byzantine Replication with Minimal Communication Costs”. In: *23rd International Conference on Database Theory (ICDT 2020)*. Vol. 155. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020, 17:1–17:20. DOI: 10.4230/LIPIcs.ICDT.2020.17.

Jelle Hellings et al. “Cerberus: Minimalistic Multi-shard Byzantine-resilient Transaction Processing”. In: *CoRR* abs/2008.04450 (2020). arXiv: 2008.04450. URL: https://arxiv.org/abs/2008.04450.

Maurice Herlihy. “Blockchains from a Distributed Computing Perspective”. In: *Communications of the ACM* 62.2 (2019), pp. 78–85. DOI: 10.1145/3209623.

Matt Higginson et al. *The promise of blockchain*. Tech. rep. McKinsey&Company, 2017. URL: https://www.mckinsey.com/industries/financial-services/our-insights/the-promise-of-blockchain.

References VIII

Muhammad El-Hindi et al. “BlockchainDB – Towards a Shared Database on Blockchains”. In: *Proceedings of the 2019 International Conference on Management of Data*. Amsterdam, Netherlands: ACM, 2019, pp. 1905–1908. DOI: 10.1145/3299869.3320237.

Muhammad El-Hindi et al. “BlockchainDB: A Shared Database on Blockchains”. In: *Proceedings of the VLDB Endowment* 12.11 (2019), pp. 1597–1609. DOI: 10.14778/3342263.3342636.

Dan Holtby, Bruce M. Kapron, and Valerie King. “Lower bound for scalable Byzantine Agreement”. In: *Distributed Computing* 21.4 (2008), pp. 239–248. DOI: 10.1007/s00446-008-0069-x.

Maged N. Kamel Boulos, James T. Wilson, and Kevin A. Clauson. “Geospatial blockchain: promises, challenges, and scenarios in health and healthcare”. In: *International Journal of Health Geographics* 17.1 (2018), pp. 1211–1220. DOI: 10.1186/s12942-018-0144-x.

Rüdiger Kapitza et al. “CheapBFT: Resource-efficient Byzantine Fault Tolerance”. In: *Proceedings of the 7th ACM European Conference on Computer Systems*. ACM, 2012, pp. 295–308. DOI: 10.1145/2168836.2168866.

Ramakrishna Kotla et al. “Zyzzyva: Speculative Byzantine Fault Tolerance”. In: *ACM Transactions on Computer Systems* 27.4 (2009), 7:1–7:39. DOI: 10.1145/1658357.1658358.

References IX

Leslie Lamport. “Paxos Made Simple”. In: *ACM SIGACT News, Distributed Computing Column 5* 32.4 (2001), pp. 51–58. DOI: 10.1145/568425.568433.

Leslie Lamport. “The implementation of reliable distributed multiprocess systems”. In: *Computer Networks (1976)* 2.2 (1978), pp. 95–114. DOI: 10.1016/0376-5075(78)90045-4.

Leslie Lamport, Robert Shostak, and Marshall Pease. “The Byzantine Generals Problem”. In: *ACM Transactions on Programming Languages and Systems* 4.3 (1982), pp. 382–401. DOI: 10.1145/357172.357176.

Laphou Lao et al. “A Survey of IoT Applications in Blockchain Systems: Architecture, Consensus, and Traffic Modeling”. In: *ACM Computing Surveys* 53.1 (2020). DOI: 10.1145/3372136.

Jean-Philippe Martin and Lorenzo Alvisi. “Fast Byzantine Consensus”. In: *IEEE Transactions on Dependable and Secure Computing* 3.3 (2006), pp. 202–215. DOI: 10.1109/TDSC.2006.35.

Shlomo Moran and Yaron Wolfstahl. “Extended impossibility results for asynchronous complete networks”. In: *Information Processing Letters* 26.3 (1987), pp. 145–151. DOI: 10.1016/0020-0190(87)90052-4.

References X

Arvind Narayanan and Jeremy Clark. “Bitcoin’s Academic Pedigree”. In: *Communications of the ACM* 60.12 (2017), pp. 36–45. DOI: 10.1145/3132259.

Senthil Nathan et al. “Blockchain Meets Database: Design and Implementation of a Blockchain Relational Database”. In: *Proceedings of the VLDB Endowment* 12.11 (2019), pp. 1539–1552. DOI: 10.14778/3342263.3342632.

Faisal Nawab and Mohammad Sadoghi. “Blockplane: A Global-Scale Byzantizing Middleware”. In: *35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 124–135. DOI: 10.1109/ICDE.2019.00020.

Michael Okun. “On the round complexity of Byzantine agreement without initial set-up”. In: *Information and Computation* 207.12 (2009), pp. 1351–1368. DOI: 10.1016/j.ic.2009.07.002.

M. Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems*. Springer, 2020. DOI: 10.1007/978-3-030-26253-2.

M. Pease, R. Shostak, and L. Lamport. “Reaching Agreement in the Presence of Faults”. In: *Journal of the ACM* 27.2 (1980), pp. 228–234. DOI: 10.1145/322186.322188.

References XI

Michael Pisa and Matt Juden. *Blockchain and Economic Development: Hype vs. Reality*. Tech. rep. Center for Global Development, 2017. URL: https://www.cgdev.org/publication/blockchain-and-economic-development-hype-vs-reality.

Dale Skeen. *A Quorum-Based Commit Protocol*. Tech. rep. Cornell University, 1982.

Maarten van Steen and Andrew S. Tanenbaum. *Distributed Systems*. 3th. Maarten van Steen, 2017. URL: https://www.distributed-systems.net/.

Gadi Taubenfeld and Shlomo Moran. "Possibility and impossibility results in a shared memory environment". In: *Acta Informatica* 33.1 (1996), pp. 1–20. DOI: 10.1007/s002360050034.

Gerard Tel. *Introduction to Distributed Algorithms*. 2nd. Cambridge University Press, 2001.

Maofan Yin et al. "HotStuff: BFT Consensus with Linearity and Responsiveness". In: *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*. ACM, 2019, pp. 347–356. DOI: 10.1145/3293611.3331591.