Research Statement

Suyash Gupta

Any large-scale distributed application guarantees clients that their data will be available and consistent despite failures. Designing a distributed application requires (i) replicating data across multiple machines so that failure of a subset of replicas does not make the service unavailable and (ii) running an agreement protocol among the replicas to order client transactions. Unfortunately, these distributed applications face the following two challenges:

1. *Arbitrary Failures.* Omission failures, where replicas crash or fail to send some messages, are common and have received tremendous attention from industries. The remaining arbitrary or commission failures, which include cases where a replica can output unexpected responses, have received minimal attention until now when organizations like Meta [4], Google [21], and Alibaba [26] are observing silent data corruptions.
2. *Malicious Failures.* The crash of multinational banks [25] and the unexpected shutting down of user accounts [2] have deteriorated trust in systems where a select group of organizations make decisions and facilitated the rise of decentralized systems, where multiple untrusting parties replicate the data; some replicas may act malicious.

As arbitrary and Byzantine (malicious) failures threaten the trust among the replicas, distributed systems need to run Byzantine Fault-Tolerant (BFT) consensus protocol to order client transactions. Despite these benefits, BFT protocols have not seen mass adoption, and cryptocurrencies are the only major decentralized applications because existing BFT consensus protocols yield low throughput, high latencies and make excessive trust assumptions.

My Approach

My research focuses on designing high throughput and low latency yielding BFT consensus protocols and decentralized systems, which can guard against crash and Byzantine failures and can be employed in large-scale data-management applications [3, 5–20, 22–24]. Accordingly, my research can be divided into the following three parts:

1. *Optimizations for efficient agreement.* As agreement protocols help replicas order client transactions, despite any failures, they require replicas to be in constant communication, which reduces system performance. As a result, system designers have started optimizing existing agreement protocols for good cases (no failures), which has led to designs that are less resilient against failures, introduce unexpected bugs, and require trust in entities external to the system. **My research systematically identifies what bottlenecks existing agreement protocols and introduces new protocols that yield good performance without sacrificing resilience against failures.**
2. *Primitives for geographical scaling.* Existing decentralized systems use the number of replicas that can participate in the consensus as a metric for scalability. In contrast, organizations like Google, and Amazon term their deployed systems as scalable if the replicas are spread across the globe; failure of replicas at one location should not make the system unavailable. When replicas are geo-distributed, existing consensus protocols yield low performance due to low bandwidth and high round-trip cost. **My research supports the design of a truly decentralized system with geo-distributed replicas and presents new consensus and communication primitives for reaching an efficient agreement among these replicas.**
3. *Application to Large-Scale Data-Management.* Almost all the existing decentralized systems focus on facilitating fast crypto transactions, and to achieve this goal, they tirelessly try to reduce the cost of BFT consensus among the replicas by introducing new optimizations to existing protocols. Unfortunately, this focus has led to these systems missing efficient protocol implementations, like modularity, adaptability, pipelining, and parallelism. As a result, these systems provide minimal opportunities to exploit modern hardware and deploy efficient consensus protocols. **My research eliminates these challenges by designing a new decentralized platform that efficiently implements protocols and facilitates creating new protocols that benefit from modern hardware.**

1. Optimizations for Efficient Agreement

My initial research argues that the complexity of agreement protocols and their failure handling mechanisms motivates systems designers to optimize only for the good case. Instead, my research presents the design of new agreement protocols that yield good performance while having the ability to withstand failures.

Scaling Crash-Failure Resistant Commit Protocols (EasyCommit [14]). From Google's Spanner to CockroachDB, several modern distributed databases rely on the two-phase commit (2PC) protocol to reach an agreement among the participants on whether a transaction needs to be committed or aborted despite knowing that the 2PC protocol lacks liveness under failures. The alternative three-phase (3PC) protocol eliminates the limitations of 2PC but is unused because it requires an extra phase of communication, which increases the response latency for the clients. We bridge this gap through our EasyCommit protocol, which requires only two phases (like 2PC) and guarantees liveness under node failures (like 3PC). EasyCommit builds on top of two insights: (1) delay the commitment of updates to the database until the transmission of the global decision to all the participants, and (2) introduce message redundancy in the network by requiring participants to forward the global decision to each other. Our evaluation illustrates that EasyCommit achieves nearly the same throughput and latency as the 2PC protocol.

Optimizing BFT Consensus via Speculative Execution (PoE) [18]. As decentralized systems are managed by untrusting parties, employing agreement protocols that guard against crash failures is insufficient. For this reason, existing decentralized systems employ BFT consensus protocols. Yet, these protocols exhibit at least one of the following limitations. (1) They require at least two phases of quadratic communication complexity (PBFT). (2) They lack concurrency and yield high latencies (HotStuff). (3) They yield good performance only under no failures (SBFT and Zyzzyva). (5) They increase replication costs (Q/U). PoE eliminates these challenges by building on top of the following three insights. First, it is safe for replicas to speculatively execute a transaction once they have determined there is no equivocation, which helps to cut down a phase from PBFT's consensus. Second, under speculative execution, a client can mark the request committed after it receives matching responses from the two-thirds majority of replicas, which comes at no additional cost, as replicas are expected to respond to the client post-execution. Final, rollbacking a request, speculatively executed by less than a majority of non-faulty replicas, is cheap. PoE achieves up to 80% more throughput than existing BFT protocols in the presence of failures.

Leaderless Parallel BFT Consensus (RCC) [10]. All the aforementioned BFT protocols follow the primary backup model, where a replica designated as the primary initiates the consensus. This model underutilizes the available network resources as the throughput of these protocols is determined mainly by the outgoing bandwidth of the primary. Regularly switching the primaries does not help either, as a switch happens after all the proposals of the previous primary are processed. RCC makes the following observations to eliminate these challenges. (1) The throughput of any primary backup protocol can be increased by enabling concurrent consensus, in which many primaries concurrently propose transactions. (2) Despite faulty replicas, the consensus instances with non-faulty primaries can always propose transactions at maximum throughput. (3) Performance of non-faulty instances remains unchanged while recovering from detectable failed instances. (4) Undetectable failures, including coordinated attacks where multiple faulty instances work together to restrict system liveness, can be resolved using lightweight checkpointing. Our evaluation shows that RCC yields up to 2.7× higher throughput than primary-backup protocols.

2. Primitives for Geographical Scaling.

In any decentralized system, it is fair to assume that the replicas may be located all across the globe. Yet, existing BFT protocols assume the availability of high bandwidth and low latency links between the replicas. My research proves that this assumption degrades system performance and presents the design of agreement protocols that support distributing participants across the WAN.

GeoScale EasyCommit [15]. Considering that databases like Spanner and CockroachDB have clusters of machines across the globe, the underlying commit protocol should take geographical distance into account. Under such a setting, even our EasyCommit protocol cannot yield acceptable performance. Our GeoScale EasyCommit (GEC) protocol resolves this situation by building on top of two insights: (1) Depending on the distance between the participants, it groups them into clusters. (2) In each cluster, GEC designates a leader. All the leaders relay the votes of the participants of their clusters and work together to reach an agreement.

GeoBFT [20]. A similar insight as GEC holds in the case of BFT protocols, as decentralized systems tend to attract parties from all over the world. However, grouping replicas into clusters, although necessary, is not sufficient to yield good performance as no single replica can be designated to communicate on behalf of its cluster. Any of the selected leaders can act Byzantine, and it will be hard to pinpoint which leader is Byzantine. The Byzantine replicas can exploit this dilemma to perform continuous attacks without getting detected. GeoBFT acknowledges these challenges and resolves them as follows. (1) It requires each cluster to run consensus on its requests. (2) Post-consensus, each cluster asks its leader to run the novel inter-cluster sharing protocol to send ordered requests to a majority of replicas in the other clusters. (3) Once a cluster has ordered requests from all the clusters, it executes them in deterministic order. Our evaluation of GeoBFT against state-of-the-art BFT protocols illustrates that it yields up to 6× more throughput.

RingBFT [23]. Despite using GeoBFT, full replication remains a bottleneck; all the clusters receive and execute all the ordered requests. Some decentralized systems avoid full replication by dividing replicas into shards and assigning each of them a unique set of data items. These systems expect this design to yield high throughput as each shard can process transactions independently. Unfortunately, these systems ignore common yet expensive cross-shard transactions, which require access to data from multiple shards and yield poor performance in practice. RingBFT achieves a high throughput consensus for cross-shard transactions (CST) without impacting the consensus throughput for single-shard transactions. It makes two observations: (1) Shards can be envisioned in a logical ring topology. (2) At most two rotations around the ring are sufficient to achieve consensus on any CST. Our results illustrate that RingBFT outputs up to 18× more throughput than the state-of-the-art protocols.

3. Application to Large-Scale Data-Management.

As a systems and databases researcher, illustrating the applicability of the BFT protocols in the wild and increasing their adoption is the prime goal of my research.

ResilientDB [12, 24]. The first step towards fulfilling this goal was to understand what is lacking in the architecture of the existing decentralized systems. Almost all of these systems aim to facilitate efficient crypto transactions, and thus, they only care about deploying efficient BFT protocols that help their replicas reach fast consensus. Consequently, these systems overlook efficiently implementing these protocols. These systems (1) do not pipeline different phases of consensus, (2) support single threaded ordering and execution, and (3) present a monolithic codebase that cannot readily deploy newer protocols and optimizations. As a result, they yield throughput at least $100\times - 1000\times$ lower than traditional distributed databases. My thesis presents ResilientDB, which presents a multi-threaded pipelined architecture and modularizes decentralized architecture into three key components: request broadcast, consensus, and execution so that developers can optimize each component separately. ResilientDB's design has been recognized as an industry standard for decentralized systems and is now incubating under the Apache Software Foundation [1].

FlexiTrust [17]. Next, my research wants to apply modern hardware to an efficient decentralized system like ResilientDB and gauge if this combination can yield a better BFT consensus protocol. We observed that some existing BFT protocols (known as TrustBFT) claim higher throughput than traditional BFT consensus protocols by attaching trusted hardware, such as Intel SGX and Keystone, with each replica and employing this trusted hardware during consensus. These TrustBFT protocols argue the following: compared to their crash fault-tolerant (CFT) counterparts, BFT protocols require $3f + 1$ replicas (f more replication than CFT protocols) to guard against f Byzantine replicas, which can equivocate–send conflicting information to participants. TrustBFT protocols force each replica to commit to a single order for each request by having the trusted component sign each message it sends, which allows these protocols to remain safe with $2f+1$ replicas only. However, we uncovered three fundamental issues with existing TrustBFT protocols: (1) limited responsiveness for clients, (2) safety concerns associated with trusted components, and (3) inability to perform multiple consensuses in parallel. We propose a novel suite of consensus algorithms (FlexiTrust) that address the aforementioned limitations. These protocols are always responsive and achieve high throughput as they (1) make minimal use of trusted components (once per client operation and at the primary replica only) and (2) support parallel consensus invocations. Both these properties are made possible by the ability to use large quorums (of size 2f+1) when using 3f+1 replicas. Our techniques can help to convert any TrustBFT protocol into a FlexiTrust protocol.

4. Other Works.

Before designing efficient algorithms for decentralized systems, I spent some time understanding the challenges faced during designing distributed applications. To our surprise, application designers did not have access to any benchmark suite that they could use to test their system. A key reason for this lack of benchmarks is the diverse nature of these distributed applications. We identified nine core distributed computing problems, and IMStuite [11] includes implementation for at least one representative algorithm in each category. The selected algorithms cover the key distributed primitives: communication (broadcast, unicast, or multicast), timing (synchronous, asynchronous, or partially synchronous), and failure. IMSuite implements these algorithms in two task-parallel languages, X10 and HJ, and has been used to create/test new optimizations for distributed applications.

Next, we observed that distributed system designers use parallelism to maximize the hardware utilization to ensure that their replicas are only network bottlenecked. On analyzing popular design practices, we observed that programmers employ the recursive task parallel (RTP) programming model to parallelize their programs. Unfortunately, system designers tend to create embarrassingly parallel RTP programs, which spawn redundant tasks and introduce overheads rather than benefit the system. We present a new optimization DECAF [16] in the context of RTP programs that benefits as follows. (1) It expands the scope of task termination constructs from the procedural definitions to their respective call-sites in a semantics-preserving manner. (2) It extracts useful parallelism by tracking the number of available idle workers at runtime as part of a mixed compiler+runtime-based optimization.

Finally, we analyzed a large number of existing edge applications and observed that they are compute-intensive and rely on cloud providers or dedicated edge servers for computing. This dependence yields undesirable high latencies when an edge device is far from the compute provider. ServerlessBFT [19] acknowledges this tradeoff and offers edge applications flexibility to select different cloud providers based on the location of their users. ServerlessBFT does so by facilitating a serverless-edge model that promotes executing lightweight tasks at the edge and compute-intensive tasks at a serverless cloud. Edge devices collaborate to spawn serverless functions for executing compute-intensive operations. As edge devices may belong to different parties, which may not trust each other, they collaborate by running an existing BFT consensus protocol.

Future Research

Over the next 5-10 years, I envision the overarching goal of my research to expedite the adoption of decentralization primitives into industrial use cases. This goal requires looking beyond improving consensus protocols, as consensus is just one element among many that we encounter while designing an efficient fault-resilient decentralized system.

Chemistry Behind Agreement. In the past 50 years, the database and systems community has introduced numerous agreement protocols, each of which tries to make a set of nodes agree on a value. Unfortunately, agreement protocols come in several flavors: commit, Crash Fault-Tolerant (CFT), and BFT protocols, depending on the types of failures they can handle. This minor disparity has created distinct communities and incompatible algorithmic designs, schematic representations, and proof systems. In our recent CIDR paper [7], we identify this disparity and present a model that unifies all the agreement protocols. We advocate a chemical representation of agreement protocols as atoms (functional properties) and elements (phases). We use these atoms and elements to represent 25 state-of-the-art agreement protocols. However, our chemical representation has revealed the following open questions: How can this model express different synchrony models? How can we argue about properties like totality, validity, consistency, and termination? How can we define various consistency constraints in this model?

Towards Constant Communication. Many organizations today use a replicated state machine (RSM) to provide high performance and reliability for applications ranging from key-value stores to microservices. Notable companies like Uber and Doordash build their infrastructure around replication, using more than 1000 microservices. Similarly, conversations with government agencies reveal that, for security, services cannot be managed across agency borders. Instead, RSMs should communicate any shared information. Breaking down a monolithic RSM into sub-RSMs can also improve performance: GeoBFT and RingBFT, for instance, highlight the benefit of running sub-RSMs locally, minimizing geo-distributed communication. These examples speak of a common need: RSMs must support the ability to efficiently and reliably exchange messages with other RSMs. Existing solutions are either ad-hoc, offer vague (and evolving) guarantees, rely on a trusted third party, or require an expensive all-to-all broadcast. Can we define a primitive that generalizes Reliable Broadcast to enforce safety and liveness guarantees between two RSMs? Can any protocol that follows such a primitive facilitate communication between two arbitrary RSMs with heterogeneous sizes, communication, and fault models? Can such a protocol send only a constant amount of additional metadata in the failure-free case?

Querying with Bounded Storage. Most decentralized systems, post-ordering a block (batches) of client transactions, make the Merkle root (the root of the Merkle tree constructed from transactions in a block) publicly available as proof that the block was ordered. If a client sends a query to determine if its request is in a specific block, they must adopt one of the following approaches. (1) Sacrifice storage by storing each Merkle tree in the disk and retrieving it from the disk to generate the proof. (2) Sacrifice performance by generating Merkle trees and proofs on the fly in response to a client query. (3) Sacrifice decentralization by depending on a third-party organization to answer the client's queries. This makes us believe that existing decentralized systems need to remodel their storage structures to support efficient querying about past transactions. Does this remodeling need to preserve the Merkle root generation, or can we design a new proving scheme? Can we disallow generating proof structures on the fly and reduce the disk storage required at each replica by at least 50%? Can this new proof structure allow replicas to leave/join the system without any loss of data?

Truly Immutable Ledgers. Decentralized systems employ BFT protocols to offer clients access to an immutable ledger that logs their transactions. Yet, recent works show that any ledger constructed with the help of a BFT protocol is vulnerable to a long-range attack where a malicious party aims to rewrite the ledger. In these BFT protocols, adding a new block to the ledger is computationally inexpensive. Each block added to the ledger includes the digital signatures of a quorum of participants, which proves that this quorum agreed to add this block to the blockchain. To perform a long-range attack, an adversary needs to compromise the private keys of the honest participants. One popular way to eliminate long-range attacks is to append PoW consensus to the impacted system. This solution is undesirable as PoW-based consensus, in practice, yields low throughput, high latencies, wastes energy, and causes temporary ledger inconsistencies (forks). This raises the following open questions: can we design a protocol that yields high throughput, low latency, consumes less energy, does not rely on long-term external safe-keeping of private keys, and is computationally expensive for an attacker to rewrite the ledger? Can such a protocol differ from PoW protocol? Can it prevent participants from competing with each other to add a new block to the ledger?

References

[1] Apache Incubator. *ResilientDB Project Incubation Status.* Oct. 2023. URL: `https://incubator.apache.org/projects/resilientdb.html`.

[2] Tara Siegel Bernard and Ron Lieber. *Banks Are Closing Customer Accounts, With Little Explanation.* Apr. 2023. URL: `https://www.nytimes.com/2023/04/08/your-money/bank-account-suspicious-activity.html`.

[3] Junchao Chen et al. “Power-of-Collaboration: A Sustainable Resilient Ledger Built Democratically”. In: *IEEE Data Eng. Bull.* 45.2 (2022), pp. 25–36.

[4] Harish Dattatraya Dixit et al. “Silent Data Corruptions at Scale”. In: *CoRR* abs/2102.11245 (2021).

[5] Suyash Gupta. “Resilient and Scalable Architecture for Permissioned Blockchain Fabrics”. In: *Proceedings of the VLDB 2020 PhD Workshop co-located with the 46th International Conference on Very Large Databases (VLDB 2020), ONLINE, August 31 - September 4, 2020.* Vol. 2652. CEUR Workshop Proceedings. CEUR-WS.org, 2020. URL: `https://ceur-ws.org/Vol-2652/paper13.pdf`.

[6] Suyash Gupta. “Resilient and Scalable Architecture for Permissioned Blockchain Fabrics”. PhD thesis. University of California, Davis, USA, 2021. URL: `https://www.escholarship.org/uc/item/6901k4tj`.

[7] Suyash Gupta, Mohammad Javad Amiri, and Mohammad Sadoghi. “Chemistry behind Agreement”. In: *13th Conference on Innovative Data Systems Research, CIDR 2023, Amsterdam, The Netherlands, January 8-11, 2023.* www.cidrdb.org, 2023.

[8] 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, October 14-18, 2019, Budapest, Hungary.* Vol. 146. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 44:1–44:3. DOI: `10.4230/LIPIcs.DISC.2019.44`.

[9] Suyash Gupta, Jelle Hellings, and Mohammad Sadoghi. *Fault-Tolerant Distributed Transactions on Blockchain.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2021. DOI: `10.2200/S01068ED1V01Y202012DTM0`

[10] Suyash Gupta, Jelle Hellings, and Mohammad Sadoghi. “RCC: Resilient Concurrent Consensus for High-Throughput Secure Transaction Processing”. In: *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021.* IEEE, 2021, pp. 1392–1403. DOI: `10.1109/ICDE51399.2021.00124`.

[11] Suyash Gupta and V. Krishna Nandivada. “IMSuite: A benchmark suite for simulating distributed algorithms”. In: *J. Parallel Distributed Comput.* 75 (2015), pp. 1–19. DOI: `10.1016/j.jpdc.2014.10.010`.

[12] Suyash Gupta, Sajjad Rahnama, and Mohammad Sadoghi. “Permissioned Blockchain Through the Looking Glass: Architectural and Implementation Lessons Learned”. In: *40th IEEE International Conference on Distributed Computing Systems, ICDCS 2020, Singapore, November 29 - December 1, 2020.* IEEE, 2020, pp. 754–764. DOI: `10.1109/ICDCS47774.2020.00012`.

[13] Suyash Gupta and Mohammad Sadoghi. “Blockchain Transaction Processing”. In: *Encyclopedia of Big Data Technologies.* Springer, 2019. DOI: `10.1007/978-3-319-63962-8\_333-1`.

[14] Suyash Gupta and Mohammad Sadoghi. “EasyCommit: A Non-blocking Two-phase Commit Protocol”. In: *Proceedings of the 21st International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018.* OpenProceedings.org, 2018, pp. 157–168. DOI: `10.5441/002/edbt.2018.15`.

[15] Suyash Gupta and Mohammad Sadoghi. “Efficient and non-blocking agreement protocols”. In: *Distributed Parallel Databases* 38.2 (2020), pp. 287–333. DOI: `10.1007/s10619-019-07267-w`.

[16] Suyash Gupta, Rahul Shrivastava, and V. Krishna Nandivada. “Optimizing recursive task parallel programs”. In: *Proceedings of the International Conference on Supercomputing, ICS 2017, Chicago, IL, USA, June 14-16, 2017.* ACM, 2017, 11:1–11:11. DOI: `10.1145/3079079.3079102`.

[17] Suyash Gupta et al. “Dissecting BFT Consensus: In Trusted Components we Trust!” In: *Proceedings of the Eighteenth European Conference on Computer Systems, EuroSys 2023, Rome, Italy, May 8-12, 2023.* ACM, 2023, pp. 521–539. DOI: `10.1145/3552326.3587455`.

[18] Suyash Gupta et al. “Proof-of-Execution: Reaching Consensus through Fault-Tolerant Speculation”. In: *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021.* OpenProceedings.org, 2021, pp. 301–312. DOI: `10.5441/002/edbt.2021.27`.

[19] Suyash Gupta et al. “Reliable Transactions in Serverless-Edge Architecture”. In: *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023.* IEEE, 2023, pp. 301–314. DOI: `10.1109/ICDE55515.2023.00030`.

[20] Suyash Gupta et al. “ResilientDB: Global Scale Resilient Blockchain Fabric”. In: *Proc. VLDB Endow.* 13.6 (2020), pp. 868–883. DOI: `10.14778/3380750.3380757`.

[21] Peter H. Hochschild et al. "Cores That Don't Count". In: *Proceedings of the Workshop on Hot Topics in Operating Systems.* HotOS '21. Ann Arbor, Michigan: Association for Computing Machinery, 2021, pp. 9–16. ISBN: 9781450384384. DOI: 10.1145/3458336.3465297.

[22] Thamir Qadah, Suyash Gupta, and Mohammad Sadoghi. "Q-Store: Distributed, Multi-partition Transactions via Queue-oriented Execution and Communication". In: *Proceedings of the 23rd International Conference on Extending Database Technology, EDBT 2020, Copenhagen, Denmark, March 30 - April 02, 2020.* OpenProceedings.org, 2020, pp. 73–84. DOI: 10.5441/002/edbt.2020.08.

[23] Sajjad Rahnama et al. "RingBFT: Resilient Consensus over Sharded Ring Topology". In: *Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29 - April 1, 2022.* OpenProceedings.org, 2022, 2:298–2:311. DOI: 10.48786/edbt.2022.17.

[24] Sajjad Rahnama et al. "Scalable, Resilient and Configurable Permissioned Blockchain Fabric". In: *Proc. VLDB Endow.* 13.12 (2020), pp. 2893–2896. DOI: 10.14778/3415478.3415502.

[25] Catherine Thorbecke. *Silicon Valley Bank collapse sends tech startups scrambling.* Mar. 2023. URL: https://www.cnn.com/2023/03/10/tech/silicon-valley-bank-tech-panic/index.html.

[26] Shaobu Wang et al. "Understanding Silent Data Corruptions in a Large Production CPU Population". In: *Proceedings of the 29th Symposium on Operating Systems Principles.* SOSP '23. Koblenz, Germany: Association for Computing Machinery, 2023, pp. 216–230. ISBN: 9798400702297. DOI: 10.1145/3600006.3613149.