World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(REVIEW ARTICLE)

Check for updates

Reducing communication overhead in leaderless consensus algorithms

Kuldeep Deshwal *

Proofpoint Inc, USA.

World Journal of Advanced Research and Reviews, 2025, 26(02), 693-701

Publication history: Received on 27 March 2025; revised on 03 May 2025; accepted on 06 May 2025

Article DOI: https://doi.org/10.30574/wjarr.2025.26.2.1589

Abstract

Leaderless consensus algorithms represent a significant advancement in distributed systems, eliminating single points of failure while enhancing fault tolerance. However, these systems face considerable communication overhead challenges as they scale to include numerous nodes across global networks. This article examines techniques that reduce message traffic while maintaining effective consensus, including quorum-based voting, gossip protocols, message aggregation and compression, asynchronous communication, and partial synchrony approaches. These methods deliver substantial benefits such as improved scalability, reduced latency, lower resource requirements, and enhanced fault tolerance. Despite these advantages, implementation presents several challenges, including consistency-efficiency trade-offs, complex implementations, security vulnerabilities, and parameter tuning difficulties. Looking forward, emerging innovations such as adaptive protocols, network-aware optimizations, hardware acceleration, hybrid approaches, and privacy-preserving techniques promise to further revolutionize communication efficiency in distributed systems.

Keywords: Asynchronous Communication; Consensus Algorithms; Fault Tolerance; Gossip Protocols; Privacy-Preserving Techniques

1. Introduction

In the world of distributed computing systems, leaderless consensus algorithms are gaining traction as a powerful alternative to traditional approaches. Think of distributed systems as networks of computers working together to perform tasks and store information like the systems behind cryptocurrencies, cloud storage, or global databases. Traditionally, these systems often relied on a "leader" node that would coordinate all decisions, similar to how a team might have a manager who approves all decisions. While this works well in many cases, it creates a weakness: if that leader fails or becomes unavailable, the entire system can grind to a halt. The concept of consensus without centralized control was first formalized in Lamport's seminal Paxos algorithm, which established the foundation for fault-tolerant distributed systems [1].

Leaderless consensus algorithms solve this problem by allowing all nodes (computers) in the network to participate in decision-making equally, without designating any single node as the ultimate authority. This is similar to how a group of friends might decide where to eat dinner through discussion and agreement rather than having one person always make the decision. This approach greatly improves fault tolerance, the system's ability to continue operating even when some components fail because there's no single point that can bring everything down. If one node stops working, the others simply continue without it. This makes leaderless systems particularly valuable for applications that need to be highly reliable and available, such as financial systems, critical infrastructure, or global communication networks.

However, as these distributed systems grow to include hundreds or thousands of nodes spread across the globe, they encounter a significant challenge: communication overhead. When every node needs to exchange messages with many

* Corresponding author: Kuldeep Deshwal

others to reach agreement, the network can become flooded with traffic. Imagine if, to decide where to eat, each person in a group of 100 friends had to personally call every other person to discuss options that would result in thousands of phone calls! Similarly, in large distributed systems, this message explosion can overwhelm network capacity, drain computational resources, and significantly slow down decision-making. This article explores innovative techniques that address this communication challenge, making leaderless consensus systems more practical and efficient for real-world applications ranging from blockchain networks and cryptocurrency systems to peer-to-peer file sharing and decentralized cloud services.

2. How This Works

Reducing communication in leaderless consensus systems involves several clever techniques that work together to minimize network traffic while still ensuring all nodes can reach agreement. Each approach tackles the communication challenge from a different angle, allowing system designers to choose the methods that best fit their specific needs.

2.1. Quorum-Based Voting

Quorum-based voting introduces a simple but powerful idea: instead of waiting for every single node to respond, the system only needs to hear from enough nodes to be confident in a decision. Think of it like taking a poll where you stop once you have a clear majority, rather than insisting on interviewing every single person. In technical terms, a quorum is a predetermined number of nodes that must agree before a decision is considered final.

For example, in a network with 100 computers, waiting for all 100 to respond could take a long time, especially if some are slow or temporarily disconnected. With quorum-based voting, the system might only need 67 nodes (a two-thirds majority) to agree before proceeding. This immediately reduces the required message traffic by 33%, which makes a huge difference in large systems. The beauty of this approach is that it maintains security and correctness while significantly reducing the communication burden. The system can be configured to require different quorum sizes depending on how critical the decision is or how much fault tolerance is needed. This approach draws from distributed database systems research that established quorum-based techniques can provide both consistency guarantees and improved performance through reduced communication overhead [2].

2.2. Gossip Protocols

Gossip protocols take inspiration from how information naturally spreads in human communities. Instead of every node broadcasting messages to the entire network (which would create enormous traffic), each node periodically selects a few random neighbors and shares information only with them. These neighbors then do the same, causing information to spread exponentially throughout the network, similar to how gossip or rumors spread through social groups.

This approach is remarkably efficient. If each node talks to just a few others in each round of communication, information can still reach the entire network quickly. For instance, in a network of 1,000 nodes, information can typically reach everyone within 10-15 communication rounds, even if each node only contacts 3-4 others each time. This vastly reduces the total number of messages compared to having each node directly contact all 999 others. Gossip protocols work especially well in large, dynamic networks where nodes may join or leave frequently, as they don't require maintaining a complete view of the network structure. Research showed that epidemic (gossip) protocols can reliably propagate information with logarithmic communication complexity, making them ideal for large distributed systems [3].

2.3. Message Aggregation and Compression

Message aggregation and compression address the overhead problem by focusing on message efficiency rather than reducing the number of communication partners. With aggregation, nodes collect multiple pieces of information over a short period and then send them together in a single, larger message rather than sending many small messages separately. This is similar to how you might save up several errands to run them all in one trip instead of making separate trips for each task.

For example, instead of immediately forwarding each vote or transaction as it arrives, a node might wait for a brief period (perhaps a few seconds) to collect several items, then bundle them together in one transmission. This dramatically reduces the overhead associated with each message, such as network headers and connection establishment. Compression techniques can then be applied to these aggregated messages, further reducing their size by eliminating redundancy in the data. Together, these techniques can reduce network traffic by 50-80% in many scenarios, allowing the system to process more actual work with the same bandwidth. Kempe's work on gossip-based

computation demonstrated that aggregation techniques can reduce communication costs while maintaining accuracy of distributed computations [4].

2.4. Eventual Consistency

Eventual Consistency allows nodes to operate at their own pace without strict coordination. In traditional synchronous systems, nodes often wait for responses from others before proceeding, creating bottlenecks when some nodes are slow. Eventual protocols remove these dependencies by allowing nodes to send messages and continue their work without waiting for immediate responses.

This approach is particularly well-suited to real-world networks where connection speeds vary and occasional delays are inevitable. By allowing faster nodes to progress independently, the system as a whole can maintain good performance even when some components are slower. Nodes can process information as it arrives and make decisions based on the best currently available data, rather than being held back by the slowest participants. This reduces both the direct communication overhead and the indirect costs of coordination, allowing for more natural adaptation to varying network conditions. This reduces both the direct communication overhead and the indirect costs of coordination, allowing for more natural adaptation to varying network conditions. The Cassandra database system demonstrates successful implementation of eventual consistency principles, enabling highly available distributed operation without strict coordination requirements [5].

2.5. Partial Synchrony

Partial synchrony represents a middle ground between fully synchronous and fully asynchronous approaches. This technique divides system operation into alternating time periods with different communication rules. During more structured synchronous periods, the system operates with stricter timing guarantees, allowing for efficient decision-making with fewer messages. During more flexible asynchronous periods, the system relaxes these constraints to accommodate network fluctuations.

This hybrid approach allows the system to get the best of both worlds: the efficiency and simplicity of synchronous operation when network conditions are favorable, and the flexibility and resilience of asynchronous operation when conditions deteriorate. For example, a distributed database might use synchronous communication for critical operations like financial transactions when the network is performing well, then switch to asynchronous mode during periods of network congestion or when some nodes become temporarily unreachable. This adaptive behavior allows the system to maintain both reliability and efficiency across varying conditions. Dwork's research on consensus in partial synchrony conditions established theoretical foundations for systems that can operate across varying network conditions [6].

3. Advantages

The techniques for reducing communication overhead in leaderless consensus algorithms deliver numerous benefits that make distributed systems more practical and effective in real-world applications. These advantages extend beyond just saving bandwidth, creating ripple effects that improve the entire system's capabilities.

3.1. Improved Scalability

When we talk about scalability in distributed systems, we're addressing a fundamental question: how well does the system grow? Communication-efficient leaderless consensus algorithms excel at handling growth because they don't suffer from the message explosion problem that plagues many traditional approaches.

In conventional systems, adding more nodes often leads to a quadratic increase in message traffic doubling the number of nodes can quadruple the number of messages. This quickly becomes unsustainable as systems grow. With communication-reduction techniques like gossip protocols and quorum-based voting, the relationship between system size and message volume becomes much more manageable. A network can expand from hundreds to thousands of nodes without drowning in its own communication traffic.

This scalability advantage is particularly important for public blockchain networks, global database systems, and other applications where the number of participants might grow unpredictably over time. For instance, a cryptocurrency network using efficient communication protocols could potentially support millions of nodes, allowing for truly global participation without requiring enormous bandwidth from each participant. This democratizes access to the network and enhances its decentralized nature, as participation doesn't require industrial-grade infrastructure. Bailis and

Ghodsi's work on eventual consistency highlights how relaxing strict consistency requirements can dramatically improve system scalability [7].

3.2. Reduced Latency

Latency is the time delay between initiating an action and seeing its effect is critical in many distributed applications. Every millisecond matters in financial transactions, online gaming, or emergency response systems. Communication-efficient consensus mechanisms dramatically reduce latency by clearing congestion from the network.

When fewer messages need to travel between nodes, each message faces less competition for network resources. This means information propagates faster throughout the system, allowing decisions to be reached more quickly. Additionally, techniques like message aggregation reduce the overhead associated with processing many small messages, further speeding up the system.

The real-world impact of this reduced latency can be substantial. Payment processing systems can confirm transactions in seconds rather than minutes. Distributed databases can provide faster query responses. Interactive applications remain responsive even under heavy load. This improved performance translates directly to better user experiences and enables new classes of applications that weren't previously practical on distributed infrastructure.

3.3. Lower Resource Requirements

Not every node in a distributed system has access to high-end hardware or unlimited bandwidth. By reducing communication overhead, these optimized consensus mechanisms democratize participation by lowering the bar for entry.

With less communication traffic to process, each node needs less computational power, memory, and network bandwidth to participate effectively. This means that consensus protocols can run on a wider variety of hardware, from powerful data center servers down to modest edge devices or even IoT sensors in some cases.

This accessibility has profound implications for system design and deployment. Organizations can deploy distributed systems using their existing infrastructure without massive upgrades. Networks can include participants from regions with less developed internet infrastructure. Systems can extend to mobile and edge computing scenarios that were previously impractical due to resource constraints. The result is more inclusive, diverse networks that can operate across a broader range of environments and use cases.

3.4. Better Fault Tolerance

The primary purpose of distributed systems is often to provide reliability even when components fail. Leaderless consensus algorithms already excel at fault tolerance by eliminating single points of failure, but communication efficiency techniques enhance this resilience even further.

When systems use techniques like gossip protocols, information has multiple potential paths to travel through the network. If some nodes or network links fail, messages can simply route around the damage, similar to how water finds new paths when its usual channel is blocked. Quorum-based approaches mean the system can make progress even when significant portions of the network are unavailable, as decisions only require a subset of nodes to participate.

These techniques also make the system more resilient to network quality issues like packet loss, congestion, or varying connection speeds. By reducing reliance on perfect communication conditions, they allow systems to maintain operation even in challenging environments where traditional approaches might fail. For example, a distributed database using these techniques could continue processing transactions during partial network outages, providing continuous service where conventional systems would become unavailable.

This enhanced fault tolerance translates to higher uptime, more consistent performance under stress, and greater overall system reliability critical factors for mission-critical applications in finance, healthcare, infrastructure, and other domains where failures have serious consequences. Schneider's state machine replication approach provides a systematic framework for building fault-tolerant services that remain available despite component failures [8].

4. Challenges

While communication-reduction techniques offer substantial benefits for leaderless consensus systems, they also introduce several significant challenges and trade-offs that system designers must navigate carefully. Understanding these challenges is crucial for building systems that not only perform well but also meet reliability, security, and consistency requirements.

4.1. Consistency vs. Efficiency Trade-offs

One of the most fundamental challenges in distributed systems is maintaining data consistency while optimizing for efficiency. When we reduce communication between nodes, we inherently limit how much information each node has about the overall system state, which can affect consistency guarantees.

In traditional distributed systems, strong consistency often relies on extensive communication to ensure all nodes have the same view of data at all times. When we implement techniques like gossip protocols or asynchronous communication, information propagates more gradually through the system. This means that for brief periods, different nodes might have different views of the system's state. For some applications, like social media feeds or content delivery networks, this temporary inconsistency might be acceptable or even unnoticeable to users. However, for applications like financial systems or medical records, even momentary inconsistencies could have serious consequences.

System designers must therefore make careful decisions about what level of consistency their application requires and select communication reduction techniques accordingly. They might implement hybrid approaches that use more communication-intensive methods for critical operations while using lighter-weight approaches for less sensitive tasks. This balancing act requires deep understanding of both the application domain and the distributed systems principles, making it one of the most challenging aspects of system design.

4.2. Implementation Complexity

Many communication-efficient consensus mechanisms are conceptually elegant but fiendishly difficult to implement correctly. The asynchronous nature of distributed systems, combined with the inherent complexity of coordinating multiple independent nodes, creates numerous edge cases and potential failure modes that must be addressed.

Gossip protocols require careful design to ensure information reliably reaches all nodes without creating hotspots or leaving some nodes isolated. Determining optimal message patterns, handling node failures, and maintaining efficiency as network conditions change all add layers of complexity. Quorum systems demand mechanisms to track which nodes have participated, handle conflicting votes, and manage membership changes. Asynchronous protocols must deal with unbounded message delays and out-of-order delivery while still ensuring progress and correctness.

This complexity extends to testing and verification as well. Distributed systems are notoriously difficult to test thoroughly because many failure modes only emerge under specific timing conditions or rare combinations of events. Formal verification of these protocols is an active research area but remains challenging for complex real-world implementations. As a result, developing robust implementations often requires specialized expertise and significant investment in validation and testing infrastructure.

4.3. Security Implications

Optimizing for communication efficiency can sometimes inadvertently create new security vulnerabilities or exacerbate existing ones. These security considerations add another dimension to the already complex design space for distributed consensus systems.

Quorum-based systems present a particularly interesting security challenge. By design, they allow decisions to be made when only a subset of nodes agree. While this improves efficiency, it also means an attacker needs to compromise fewer nodes to influence system decisions. For example, in a system with 100 nodes that requires 67 for a quorum, an attacker who controls 34 nodes could potentially block consensus, while controlling 67 nodes could allow them to force incorrect decisions. This represents a security threshold that's lower than in systems requiring near-unanimous agreement.

Gossip protocols introduce different security considerations. Their randomized communication patterns make them relatively robust against targeted attacks on specific communication paths. However, they can be vulnerable to eclipse attacks, where an attacker isolates certain nodes by controlling all their communication partners. Message aggregation

creates opportunities for message tampering if cryptographic protections aren't properly implemented, as modifying a single aggregated message could affect multiple pieces of information at once.

System designers must incorporate security analysis into their communication optimization strategies, often implementing additional protections like cryptographic verification, byzantine fault tolerance mechanisms, or reputation systems to mitigate these risks. Eyal and Sirer's research revealed how consensus protocols can be vulnerable to strategic manipulation by attackers controlling even a minority of resources [9].

4.4. Parameter Tuning

Communication-efficient consensus systems involve numerous configurable parameters that significantly impact their performance, and finding optimal settings for these parameters presents a major operational challenge. These systems rarely have a one-size-fits-all configuration that works well across all deployment scenarios.

For gossip protocols, parameters include how frequently nodes gossip, how many peers they contact each round, and how they select those peers. Quorum systems must determine appropriate quorum sizes that balance efficiency against fault tolerance. Message aggregation systems need policies for how long to collect messages before sending and maximum aggregation sizes. Each of these parameters affects system behavior in complex, interconnected ways that may not be immediately obvious.

Making matters more complicated, the optimal settings often depend on dynamic factors like network size, topology, traffic patterns, and failure rates. A configuration that works perfectly for a stable network of 100 nodes might perform poorly when the network grows to 1,000 nodes or experiences increased node churn. This necessitates extensive testing under various conditions, sophisticated monitoring systems, and sometimes adaptive algorithms that automatically adjust parameters based on observed performance.

Organizations deploying these systems must invest in performance testing, monitoring infrastructure, and operational expertise to ensure their systems remain well-tuned as conditions evolve. This operational complexity represents a significant hidden cost of implementing communication-efficient consensus mechanisms, beyond the initial development effort.

5. Future Directions

The field of leaderless consensus is not standing still. Researchers and engineers are actively exploring innovative approaches to further reduce communication overhead while addressing current limitations. These emerging techniques promise to make distributed systems even more efficient, secure, and powerful in the coming years.

5.1. Adaptive Protocols

Today's consensus protocols typically use fixed communication patterns and decision rules, regardless of changing network conditions. The future lies in adaptive systems that can dynamically adjust their behavior based on real-time observations of the network environment.

Imagine a distributed system that monitors factors like network congestion, node response times, and current workload, then automatically tunes its communication strategy accordingly. During periods of light activity, it might use more communication-intensive protocols to achieve faster consensus. When the network becomes congested or when many nodes are slow to respond, it could seamlessly switch to more communication-efficient approaches that prioritize throughput over latency. This adaptability would allow systems to maintain optimal performance across widely varying conditions.

Advanced adaptive protocols might even learn over time, using machine learning techniques to recognize patterns in network behavior and predict optimal configurations before problems arise. For example, a system might notice that certain times of day consistently show higher network congestion and proactively adjust its parameters in anticipation. This self-tuning capability would significantly reduce the operational burden of maintaining these complex systems while improving their overall efficiency.

5.2. Network-Aware Optimizations

Current consensus protocols often treat the underlying network as a black box, with limited understanding of its actual structure and characteristics. Future approaches will likely incorporate sophisticated awareness of network topology to make smarter routing decisions.

By understanding the physical and logical arrangement of nodes which are physically close, which have fast connections between them, where bottlenecks exist consensus algorithms could strategically plan communication paths. Rather than using randomized communication patterns, messages could follow optimized routes that minimize overall network load and avoid congestion points. Nodes could preferentially communicate with well-connected neighbors that can efficiently propagate information to the rest of the network.

This network awareness might extend to understanding higher-level infrastructure as well. Systems could identify when nodes are in the same data center, region, or cloud provider, and use this information to minimize expensive cross-region traffic while ensuring sufficient geographic diversity for fault tolerance. The result would be distributed systems that achieve consensus with far less overall network traffic, enabling larger and more efficient deployments.

5.3. Hardware Acceleration

As distributed systems become more critical to global infrastructure, we're likely to see increased investment in specialized hardware designed specifically to accelerate consensus operations. This represents a shift from purely software-based solutions to hardware-software co-design approaches.

Consensus algorithms involve computationally intensive operations like cryptographic signature verification, hash calculations, and message processing. Specialized hardware accelerators either dedicated chips or features integrated into general-purpose processors could perform these operations orders of magnitude faster than software implementations. For example, custom ASICs (Application-Specific Integrated Circuits) could verify hundreds of digital signatures simultaneously, dramatically reducing the processing time needed for each consensus round.

Hardware acceleration would be particularly valuable for resource-constrained environments like edge computing, IoT networks, or mobile devices. By reducing the computational burden of participation, it would enable leaderless consensus protocols to run efficiently on a much wider range of devices. This could expand the reach of decentralized systems into new domains where they were previously impractical due to performance limitations.

5.4. Hybrid Approaches

Rather than viewing leader-based and leaderless approaches as competing alternatives, future systems will increasingly combine elements of both to leverage their complementary strengths. These hybrid approaches aim to get the best of both worlds: the communication efficiency of leader-based systems during normal operation and the resilience of leaderless systems during failures.

A promising hybrid design pattern involves using a leader-based protocol as the primary consensus mechanism during stable periods, with fast fallback to a leaderless approach when leaders fail or become unavailable. The system continuously maintains the infrastructure for both approaches, allowing for rapid switching without disruption. This provides both the performance benefits of centralized coordination and the fault tolerance of decentralized decision-making.

More sophisticated hybrid systems might employ multiple consensus mechanisms simultaneously for different types of operations. Critical, high-value transactions might use more robust, communication-intensive protocols, while routine operations use lighter-weight approaches. The system could dynamically assign incoming requests to the appropriate consensus channel based on their requirements. This multi-tiered architecture would optimize resource usage while still providing strong guarantees where needed.

5.5. Privacy-Preserving Techniques

Traditional consensus protocols require nodes to share considerable information with each other transaction details, votes, system state which creates both communication overhead and potential privacy concerns. Advanced cryptographic techniques are emerging that allow nodes to reach agreement while sharing significantly less data.

Zero-knowledge proofs represent one of the most promising approaches in this area. These cryptographic constructs allow one party to prove to others that a statement is true without revealing any additional information beyond the

validity of the statement itself. Applied to consensus systems, they could allow nodes to verify that others have followed protocol rules correctly without sharing the underlying data. For example, rather than broadcasting complete transaction details, nodes could share zero-knowledge proofs that the transactions are valid, significantly reducing message sizes.

Secure multi-party computation offers another avenue for privacy-preserving consensus. This technique allows multiple parties to jointly compute functions over their inputs while keeping those inputs private. In distributed systems, this could enable nodes to collectively reach consensus decisions without revealing their individual votes or data. By reducing the amount of information that needs to be exchanged, these techniques not only enhance privacy but also substantially decrease communication overhead.

The integration of these emerging cryptographic techniques into consensus protocols is still in its early stages, but the potential impact on both communication efficiency and privacy is enormous. As these methods mature and become more practical for real-world deployment, they could fundamentally transform how distributed systems operate, enabling new applications in privacy-sensitive domains.

By continuing to advance these frontier technologies, the distributed systems community is working toward a future where highly efficient, scalable consensus can be achieved with minimal communication overhead. This will enable the next generation of decentralized applications that can operate at global scale while maintaining performance, security, and privacy. The ongoing innovation in this space promises to expand the reach of distributed systems into new domains and use cases that were previously impractical due to communication limitations. Abraham's work on incentive-compatible consensus demonstrates how cryptographic techniques can align economic incentives with protocol compliance while preserving privacy [10].

6. Conclusion

Reducing communication overhead in leaderless consensus algorithms transforms distributed systems from theoretical concepts into practical, scalable solutions for real-world applications. By implementing techniques like quorum-based voting, gossip protocols, and message aggregation, systems can achieve consensus efficiently even at massive scale. These approaches not only reduce bandwidth usage but fundamentally improve scalability, performance, accessibility, and reliability. While challenges exist in balancing consistency with efficiency and addressing security concerns, the field continues to evolve rapidly. Emerging technologies like adaptive protocols, network-aware optimizations, and privacy-preserving cryptographic techniques point toward a future where distributed systems can support increasingly sophisticated applications with minimal communication overhead. This ongoing innovation expands the practical reach of decentralized architectures, enabling new categories of applications that require both robust consensus and efficient operation at global scale, ultimately democratizing access to reliable distributed computing across diverse environments and use cases.

References

[1] Leslie Lamport, "The part-time parliament," ACM Transactions on Computer Systems, vol. 16, no. 2, pp. 133-169, 1998. [Online]. Available: https://lamport.azurewebsites.net/pubs/lamport-paxos.pdf

[2] Omar Mohammed Bakr and Idit Keidar, "On the Performance of Quorum-Based Systems over the Internet," Electrical Engineering and Computer Sciences , University of California at Berkeley, 2008. [Online]. Available: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-141.pdf

[3] Alan Demers, et al., "Epidemic Algorithms For Replicated Database Maintenance," in Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing (PODC '87), 1987, pp. 1-12. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/41840.41841

[4] David Kempe, et al., "Gossip-Based Computation of Aggregate Information," in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS '03), 2003, pp. 482-491. [Online]. Available: https://www.cs.cornell.edu/johannes/papers/2003/focs2003-gossip.pdf

[5] Avinash Lakshman and Prashant Malik, "Cassandra — A Decentralized Structured Storage System," ACM SIGOPS Operating Systems Review, 2010. [Online]. Available: https://www.researchgate.net/publication/220624179_Cassandra_-_A_Decentralized_Structured_Storage_System

[6] Cynthia Dwork, et al., "Consensus in the Presence of Partial Synchrony," Journal of the ACM, vol. 35, no. 2, pp. 288-323, 1988. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/42282.42283

[7] Peter Bailis and Ali Ghodsi, "Eventual consistency today: Limitations, extensions, and beyond," Communications of the ACM, vol. 56, no. 5, pp. 55-63, 2013. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/2460276.2462076

[8] Fred B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," ACM Computing Surveys, Vol. 22, No. 4, December 1990. [Online]. Available: https://www.cs.cornell.edu/fbs/publications/SMSurvey.pdf

[9] Ittay Eyal and Emin Gun Sirer, "Majority is not enough: Bitcoin mining is vulnerable," Communications of the ACM, Volume 61, Issue 7, 2018. [Online]. Available: https://www.cs.cornell.edu/~ie53/publications/btcProcFC.pdf

[10] Ittai Abraham, et al., "Solida: A Blockchain Protocol Based on Reconfigurable Byzantine Consensus," arXiv preprint arXiv:1612.02916, 2017. [Online]. Available: https://arxiv.org/pdf/1612.02916