# Reliable Broadcast Networking Stacks

Mariya Koeva, Filip Rezabek*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: mariya.koeva@tum.de, frezabek@net.in.tum.de

***Abstract*—This paper investigates the design, analysis, and potential of robust and reliable broadcast protocols in distributed systems, addressing challenges like fault tolerance, latency, and communication efficiency in both stand-alone and simulation-based frameworks. Existing broadcast protocols are examined, with a focus on evaluating their scalability and resilience under failure conditions. We explore the limitations of traditional transport protocols like TCP and introduce modern alternatives such as QUIC, which offer enhanced performance in high-latency, high-concurrency environments. QUIC's built-in encryption, faster connection setup, and improved multiplexing make it a promising alternative for reliable broadcast communication. The study provides a comparative analysis of TCP and QUIC, highlighting their strengths, weaknesses, and use cases. Additionally, it examines the role of advanced cryptographic techniques like threshold cryptography and Distributed Key Generation (DKG) and discusses open-source implementations, for instance, libp2p. The findings offer valuable insights for future research in the deployment of the QUIC protocol in reliable broadcasts.**

***Index Terms*—reliable broadcast, distributed systems, broadcast protocols, fault tolerance, communication efficiency, scalability, TCP, QUIC, TLS, consensus algorithm, threshold cryptography, distributed key generation (DKG), libp2p**

## 1. Introduction

Reliable broadcast is a fundamental primitive in distributed systems, ensuring that messages are consistently delivered to all intended recipients, even under adverse conditions such as network failures or malicious attacks [1]. These protocols underpin critical applications like blockchain networks, distributed databases, and fault-tolerant systems. Three primary requirements must be met for reliability: **validity, agreement and integrity** [2]. Validity ensures that all correct nodes eventually deliver a message if the sender is correct; agreement guarantees that all correct nodes deliver the same message; and integrity prevents tampering or duplication of messages. These requirements were first formalized in 1983 [3] and remain foundational in distributed systems research.

Traditional transport protocols like TCP have been widely adopted due to their robustness and well-understood reliability mechanisms. Combined with TLS (Transport Layer Security) for encryption, TCP has long been the standard for secure communication. However, these solutions face challenges in scalability and efficiency, particularly in high-latency environments or under heavy network congestion. To address these limitations, modern alternatives like QUIC [4] have emerged. QUIC incorporates built-in encryption, multiplexing, and faster connection setups, presenting a promising alternative for reliable communication, including broadcast scenarios.

This paper explores the current state of reliable broadcast solutions, analyzing existing algorithms and used protocols with a focus on evaluating the feasibility of QUIC as a substitute for TCP. This is achieved through a comparative analysis of these transport protocols, their advantages and disadvantages, and use cases involving advanced cryptographic techniques such as threshold cryptography and Distributed Key Generation (DKG). Furthermore, we investigate open-source implementations, including libp2p, and discuss the possible future developments in reliable broadcasting.

## 2. Background, Related Work and Existing Broadcast Solutions

Ensuring reliable broadcast in distributed systems is a fundamental challenge, especially in environments where communication may be unreliable, and nodes may fail. To address this, various consensus mechanisms and synchronization models have been developed to guarantee message delivery and agreement among nodes. This section explores the principles behind reliable broadcast and consensus, highlighting key algorithms, and examines the role of different synchronization models in achieving reliable communication.

### 2.1. Consensus Algorithms

Consensus algorithms are fundamental to ensuring consistency in distributed systems, especially when dealing with unreliable communication and potential faults. These algorithms enable a set of processes or nodes to agree on a single value, even in the presence of failures, ensuring reliable broadcast of messages. A few notable consensus algorithms are widely used to achieve this reliability:

**2.1.1. Paxos.** Paxos is a foundational consensus algorithm used in partially synchronous to synchronous systems. Paxos ensures that a majority of nodes reach agreement on a single value despite faults (e.g, node crashes). It operates under the assumption that at least one correct process is not faulty, and it guarantees safety and liveness. Safety in this context means that no two nodes will

 doi: 10.2313/NET-2025-05-1_18

decide on different values and liveness means that a decision will eventually be made under specific conditions. While Paxos is a breakthrough in consensus protocols, its implementation is viewed as complex and inefficient in terms of performance due to frequent communication and coordination overheads [5].

**2.1.2. Raft.** Raft was designed as a more understandable alternative to Paxos, focusing on clarity and ease of implementation. Raft divides the consensus process into three key components: leader election, log replication, and safety. A leader is elected to manage the consensus process, and the leader's log is replicated across follower nodes. Raft ensures that all changes to the system are coordinated through the leader, simplifying the process and improving efficiency in comparison to Paxos. [6].

**2.1.3. Byzantine Fault Tolerant (BFT) Algorithms.** In more adversarial environments, where nodes may behave arbitrarily, BFT algorithms are used. For example, Practical Byzantine Fault Tolerance (PBFT) is designed to tolerate up to one-third of the nodes being compromised by a Byzantine adversary. The algorithm works by having nodes exchange messages in multiple rounds to achieve agreement on a transaction, even in the presence of faulty or malicious participants. [7].

## 2.2. Sybil resistance approaches

While Consensus algorithms determine how nodes agree on a single state or value in the system, sybil resistance determines who gets to participate in the network and ensures that participation is fair, preventing adversaries from gaining excessive influence. A Sybil attack occurs when an adversary controls a large number of nodes, allowing them to disrupt the network by censoring messages, invalidating transactions, or influencing consensus outcomes. In the context of reliable broadcast, Sybil resistance ensures that an adversary cannot gain disproportionate control over message dissemination.

Decentralized systems implement Sybil-resistant mechanisms to counterfeit sybil attacks by imposing economic or computational barriers to prevent adversaries from cheaply creating multiple identities. Proof of Stake (PoS): Forces participants to stake cryptocurrency as collateral, ensuring economic penalties for dishonest behavior.

Sybil resistance mechanisms often incorporate consensus techniques to validate participation. For example, PoS uses consensus among staked participants to finalize blocks while preventing Sybil attacks by requiring economic commitment. Sybil resistance is a prerequisite for secure consensus in decentralized networks. Without it, consensus protocols become vulnerable to 51% attacks (in PoW) or stake concentration attacks (in PoS), where an attacker gains the majority influence and disrupts agreement.

Moreover, the efficiency and scalability of consensus mechanisms are influenced by the choice of Sybil resistance methods. PoW offers strong Sybil resistance but suffers from high energy consumption, whereas PoS reduces energy usage but introduces new challenges, such as stake centralization risks. Some systems combine multiple mechanisms, such as PoS with reputation-based Sybil resistance, to balance security, efficiency, and decentralization.

**2.2.1. Proof of Work (PoW).** In this approach, miners compete to solve complex mathematical puzzles using computational power. The first to succeed earns the right to create a new block and receive rewards. PoW ensures security by making mining computationally expensive, deterring Sybil attacks and fraudulent transactions. An attacker would need to control more than 51% of the network's total computational power to alter transactions, which is highly costly and impractical. However, PoW's high energy consumption raises questions regarding its scalability and environmental impact. Despite this, it remains widely used in networks like Bitcoin, Litecoin, and Monero, where strong security and decentralization are prioritized [8].

**2.2.2. Proof of Stake (PoS).** PoS offers a more energy-efficient alternative by selecting validators based on the number of coins they have staked as collateral rather than computational work. This system ensures security by making dishonest behavior costly, as malicious validators risk losing their staked assets [9]. Unlike PoW, PoS significantly reduces energy consumption while maintaining Sybil resistance, as attackers would need to control a majority of the staked assets to manipulate the network. It also supports faster transaction finality, making it more scalable for high-volume applications. PoS is widely adopted in blockchain platforms like Ethereum 2.0.

## 2.3. Synchronization Models in Reliable Broadcasting

In the context of reliable broadcasting, the synchronization model of a system plays a crucial role in determining the guarantees provided by the underlying communication protocols. These synchronous, partially synchronous, and asynchronous models define the expectations regarding timing and network delays, and they influence the design and robustness of reliable broadcast mechanisms.

Synchronous systems operate under strict timing assumptions, where a known finite upper bound ($\Delta$) ensures that messages are delivered within a specified timeframe [10]. This model simplifies protocol design by providing predictable communication, making it suitable for applications requiring high reliability and determinism. Examples include high-frequency trading systems and real-time control networks. However, the reliance on accurate $\Delta$ value presents challenges: a conservatively large $\Delta$ may degrade performance due to long timeouts, while a small $\Delta$ risks safety violations in real-world conditions.

The partially synchronous model bridges the gap between synchrony and complete asynchrony. It assumes an unknown finite upper bound ($\Delta$) on message delays, which holds only after a Global Stabilization Time (GST) [11]. Before GST, the system behaves asynchronously. This model is highly applicable in real-world distributed systems where networks experience transient disruptions but eventually stabilize.

 doi: 10.2313/NET-2025-05-1_18

Asynchronous systems place no bounds on message delivery times, making them highly flexible but challenging for achieving consensus. The lack of timing guarantees means that protocols must ensure eventual message delivery and consensus without relying on temporal assumptions. This model is essential for environments with unpredictable delays, such as highly decentralized peer-to-peer networks.

# 3. Comparative Analysis: TCP vs. QUIC for Reliable Broadcast

Reliable broadcast depends heavily on the underlying transport protocol, which influences factors such as latency, security, and message delivery efficiency. This section explores the strengths and weaknesses of TCP and QUIC protocols and compares them.

## 3.1. TCP as the traditional transport layer

TCP (Transmission Control Protocol) [12] is the cornerstone of reliable data transmission in modern networking. It offers reliability through mechanisms such as retransmissions, in-order delivery, and congestion control. These guarantees make TCP essential for applications like file transfers, web browsing, and distributed databases like Google Spanner. TCP's acknowledgment system ensures data integrity, while its flow control mechanisms [13] aim to prevent network congestion.

**3.1.1. Challenges of TCP.** TCP suffers from inherent latency due to its three-way handshake and sequential acknowledgment system. The Round-Trip Time (RTT) for a TCP connection establishment is at least one RTT for the handshake, and additional RTTs are incurred for data transfer, especially with larger datasets. Moreover, the sequential acknowledgment of packets introduces head-of-line blocking, where a single lost packet can delay the entire stream, a significant drawback for time-sensitive applications [14].

The inefficiency of TCP becomes more pronounced in large-scale systems, particularly where high throughput and low-latency communication are required, such as in real-time media streaming or large distributed systems.

**3.1.2. Deployment in Real-World Broadcast Systems.** TCP is integral to traditional, reliability-sensitive systems like distributed databases (e.g., Apache Cassandra, MongoDB) where data consistency and integrity are critical. The established ecosystem around TCP and different versions of TLS guarantees secure and reliable connections in transactional systems. However, the latency and lack of multiplexing are limitations in dynamic, distributed environments, where faster connection establishment and handling of multiple streams are required [15].

## 3.2. QUIC as an Alternative

QUIC is a modern transport protocol designed by Google to address some of the TCP limitations, especially in terms of connection setup time, security, and multiplexing. Key Features of QUIC are:

**Built-in Encryption:** QUIC integrates Transport Layer Security (TLS) 1.3 directly into the protocol, eliminating the need for a separate security layer as required by protocols like HTTP/2 over TLS.

**Reduced handshake:** This built-in encryption also reduces connection setup time by consolidating the transport(previously TCP) and security(TLS) handshakes into a single phase. QUIC establishes secure connections more efficiently compared to TCP, as the TLS handshake occurs alongside the initial connection establishment, reducing latency significantly [4].

**Fast Connection Establishment:** QUIC also supports 0-RTT (zero round-trip time) connection establishment, allowing data to be sent in the initial packet even before the handshake is complete. This is particularly beneficial in environments where latency is a concern, such as high-frequency trading or live-streaming applications. [4]

**Multiplexing:** QUIC improves the efficiency of multiplexing by allowing multiple independent data streams within a single connection [4]. Unlike TCP, where head-of-line blocking can occur, QUIC allows independent streams to proceed without interference. This is particularly important in modern applications where multiple data types are transmitted simultaneously.

**3.2.1. Potential Advantages in Reliable Broadcasting.** QUIC's design introduces new possibilities for use in reliable broadcasting scenarios, particularly in distributed systems where rapid and fault-tolerant message delivery is crucial.

QUIC's low-latency connection setup and inherent support for multiplexing make it suitable for distributed systems requiring rapid message dissemination. This, combined with its resistance to head-of-line blocking, ensures that broadcast messages can be delivered quickly and efficiently, even in environments with unreliable network conditions, where network changes are a possibility. This is due to the fact that QUIC relies on a Connection ID to identify connections, and not on the IP Addresses and Sockets of the participants.

**3.2.2. Limitations and Open Questions.** Despite its many advantages, QUIC presents certain challenges that may affect its adoption in all contexts.

One significant challenge to QUIC's adoption is its limited compatibility with legacy systems, particularly those that rely on TCP-based communication stacks. QUIC requires modern infrastructure and support for UDP (User Datagram Protocol), making it less suitable for environments where legacy protocols are deeply integrated.

# 4. Analysis and open-source implementations

In this section we explain existing implementation approaches and make assumptions about possible enhancements through future work.

## 4.1. Implementations focusing on TCP-based solutions

One notable example where TCP outperforms QUIC in reliable broadcast scenarios is its role in PBFT consensus. A study evaluating the performance of QUIC in

PBFT-based blockchain networks found that TCP achieves better execution times due to its optimized message replication and congestion control mechanisms [16]. PBFT relies on frequent and structured message exchanges across multiple communication rounds, requiring consistent, in-order delivery and efficient retransmission of lost packets. TCP, with its persistent connections and built-in reliability mechanisms, efficiently handles this high message volume, ensuring that consensus operations proceed with minimal delays. In contrast, QUIC must implement reliability and congestion control at the transport layer, leading to additional processing overhead. QUIC reduces connection establishment latency and supports multiplexing, but the study showed that these benefits do not improve PBFT execution times. QUIC's congestion control mechanisms, such as BBR and New Reno, don't consistently outperform TCP due to their handling of packet loss and congestion feedback in high-frequency message replication scenarios. The simulation results indicate that TCP remains the preferred choice for PBFT and similar consensus protocols, where timely, ordered message delivery and efficient network congestion handling are crucial. However, the study suggests that QUIC may become more viable with further optimizations in congestion control for large-scale distributed systems. Protocols designed for partial synchrony, such as Paxos or Raft [17], can be implemented over TCP/TLS, especially when message delivery times are uncertain but the system can eventually stabilize. These protocols ensure that once a system reaches GST, reliable communication can be achieved, with TCP/TLS offering secure and reliable transport. However, when low-latency and high-throughput are required, QUIC may be favored.

## 4.2. QUIC use cases

Asynchronous Byzantine Fault Tolerant (ABFT) algorithm is an example of an asynchronous protocol that can be implemented over QUIC in cases where low latency is critical, especially in decentralized or mobile networks. QUIC offers distinct advantages over TCP in asynchronous systems due to its reduced handshake overhead, which allows for faster message exchanges. However, those are mainly assumptions that haven't been consolidated by by research.

In asynchronous environments, TCP can still be used, but it is typically less efficient compared to QUIC due to the higher latency introduced by TCP's handshake and slower connection re-establishment after packet loss. TCP and TLS are more suitable for environments with moderate to low network instability, while QUIC performs better in volatile networks(e.g, mobile environments or global-scale decentralized systems).

**4.2.1. Potential of QUIC for Reliable Broadcast in Decentralized Transactions.** QUIC has shown promise in decentralized transaction systems, particularly in peer-to-peer Bitcoin transactions and payment channels. Studies highlight that QUIC's low-latency handshake, built-in encryption, and multiplexing make it well-suited for fast and secure financial transactions in blockchain networks. QUIC Bitcoin Transactions leverage QUIC's Connection ID mechanism to enable direct, unpublished transaction exchanges between peers, reducing reliance on intermediaries and mitigating security risks such as man-in-the-middle (MITM) attacks [18]. Additionally, QUIC Bitcoin Channels introduce efficient payment channels that maintain state even during network switches, allowing seamless transactions between users and machines. These capabilities suggest that QUIC could be a valuable alternative to TCP for high-speed, trustless financial exchanges, particularly in use cases requiring lightweight, adaptable, and encrypted communication channels in decentralized networks.

**4.2.2. Threshold Cryptography and Distributed Key Generation (DKG).** Threshold cryptography, particularly Distributed Key Generation (DKG), plays a fundamental role in decentralized security mechanisms. As outlined in Das and Ren's work [19], DKG ensures that signing keys are securely distributed among multiple participants rather than being held by a single entity. Their scheme employs adaptively secure BLS threshold signatures, which maintain security even if an adversary selects corrupted nodes dynamically. The protocol minimizes overhead compared to prior DKG approaches, ensuring efficient key management in distributed environments. This decentralized key-sharing mechanism is particularly relevant for secure broadcast systems, where maintaining integrity and fault tolerance in adversarial conditions is crucial.

QUIC, on the other hand, integrates cryptographic handshakes with transport-layer encryption, minimizing round-trip times and ensuring secure data transmission [4]. While QUIC itself does not inherently implement DKG, its built-in encryption and low-latency communication may offer advantages in scenarios where distributed key agreement protocols like DKG are deployed. Specifically, QUIC's multiplexed streams and rapid reconnection capabilities could potentially reduce the overhead associated with key exchange and cryptographic signing in decentralized networks. However, no existing research—including that of Das and Ren—has explicitly evaluated QUIC's interaction with DKG. Further studies would be required to determine whether QUIC's performance benefits align with the security guarantees of threshold cryptographic protocols.

**4.2.3. DAG-Based Consensus Mechanisms:.** QUIC's low-latency and efficient communication protocols align well with Directed Acyclic Graph (DAG) structures used in systems like IOTA [20]. DAGs enable parallel transaction processing without the bottleneck of sequential blocks typical of traditional blockchain systems. QUIC's quick message propagation and connection setup are beneficial for the high-throughput and low-latency requirements of DAG-based consensus, where real-time message delivery is essential for scalability and network efficiency.

**4.2.4. libp2p.** In libp2p, reliable broadcast is essential for efficient message delivery across peers. TCP and QUIC—play key roles in this functionality.

TCP's head-of-line blocking can be a significant drawback in real-time applications. Additionally, TCP's reliance on middleboxes for header inspection can create challenges for deploying new protocol features [21]. QUIC avoids head-of-line blocking which makes it ideal

for real-time and high-concurrency environments. WebTransport, built on QUIC, offers an alternative to WebSockets by enabling bidirectional communication over stream multiplexing [22]. Unlike WebSockets, which use a single connection, WebTransport allows multiple streams to operate in parallel, improving performance. It also enables browsers to connect to libp2p nodes securely using self-signed certificates, addressing WebSocket's limitations in peer-to-peer networks. A standard WebSocket connection is conducted through 6 RTTs:

- 1 RTT for TCP handshake.
- 1 RTT for TLS 1.3 handshake.
- 1 RTT for WebSocket upgrade.
- 1 RTT for multistream security negotiation (Noise or TLS 1.3).
- 1 RTT for security handshake (Noise or TLS 1.3).
- 1 RTT for multistream muxer negotiation (mplex or yamux).

In comparison, WebTransport only requires 3 RTTs:

- 1 RTT for QUIC handshake.
- 1 RTT for WebTransport handshake.
- 1 RTT for libp2p handshake; one for multistream and one for authentication(with a Noise handshake) [22].

## 5. Conclusion and future work

In conclusion, while the exploration of TCP and QUIC for reliable broadcast in distributed systems has provided valuable insights, it is clear that further research is needed, particularly regarding QUIC's evolving capabilities and potential for broader adoption. QUIC's inherent advantages in latency reduction, multiplexing, and built-in encryption make it an increasingly promising candidate for high-performance applications, especially in real-time communication, streaming, and large-scale decentralized systems. QUIC is still undergoing development and its full potential in diverse distributed environments remains an area for continued investigation. QUIC continues to mature and gain support across the industry. In my opinion it is expected that QUIC's deployment will expand, offering new opportunities for optimizing reliable broadcasting in environments where speed, scalability, and fault tolerance are critical. Future research should focus on refining QUIC's interoperability with legacy systems, exploring hybrid solutions that combine the strengths of both TCP and QUIC, and further integrating advanced cryptographic techniques to enhance the security and efficiency of reliable broadcast protocols. As the adoption of QUIC accelerates, it is anticipated that it will play an increasingly integral role in shaping the next generation of communication protocols, driving innovation in distributed systems and beyond.

## References


[1] H. Zhu, F. Bao, and R. H. Deng, "Robust and reliable broadcast protocols in the stand-alone and simulation-based frameworks," *2008 IEEE International Conference on Communications*, pp. 1635–1641, 2008.

[2] A. Kshemkalyani and M. Singhal, "Chapter 14: Consensus and Agreement," pp. 510–566, 2012.

[3] D. Dolev and H. R. Strong, "Authenticated algorithms for byzantine agreement," https://www2.imm.dtu.dk/courses/02220/2015/L12/DolevStrong83.pdf, 1983.

[4] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9000

[5] "Paxos consensus algorithm," 2024. [Online]. Available: https://www.geeksforgeeks.org/paxos-consensus-algorithm/

[6] S. Aggarwa, "Raft and paxos : Consensus algorithms for distributed systems," Aug. 2023. [Online]. Available: https://medium.com/@mani.saksham12/raft-and-paxos-consensus-algorithms-for-distributed-systems-138cd7c2d35a

[7] R. Behnke, "What is practical byzantine fault tolerance in blockchain?" Oct. 2023. [Online]. Available: https://www.halborn.com/blog/post/what-is-practical-byzantine-fault-tolerance-in-blockchain

[8] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[9] "Understanding sybil attacks and consensus mechanisms in blockchain," 2023. [Online]. Available: https://www.nervos.org/knowledge-base/sybil_attacks_consensus_mechanisms_(explainCKBot)

[10] I. Abraham, "Synchrony, asynchrony and partial synchrony," Jun. 2019. [Online]. Available: https://decentralizedthoughts.github.io/2019-06-01-2019-5-31-models/

[11] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the Association for Computing Machinery*, vol. 35, no. 2, pp. 288–323, Apr. 1988. [Online]. Available: https://groups.csail.mit.edu/tds/papers/Lynch/jacm88.pdf

[12] "Transmission Control Protocol," RFC 793, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc793

[13] E. Blanton, D. V. Paxson, and M. Allman, "TCP Congestion Control," RFC 5681, Sep. 2009. [Online]. Available: https://www.rfc-editor.org/info/rfc5681

[14] A. Varshney, "Head-of-line (hol) blocking in http/1 and http/2," Jun. 2021. [Online]. Available: https://engineering.cred.club/head-of-line-hol-blocking-in-http-1-and-http-2-50b24e9e3372

[15] "Latency in distributed system," Aug. 2024. [Online]. Available: https://www.geeksforgeeks.org/latency-in-distributed-system/

[16] T. L. Habibi and R. F. Sari, "Performance evaluation of quic protocol in message replication overhead in pbft consensus using ns-3," *International Journal of Electrical, Computer, and Biomedical Engineering*, 2023.

[17] P. Sapkota, "Consensus algorithms: Paxos and raft," Sep. 2023. [Online]. Available: https://pragyasapkota.medium.com/consensus-algorithms-paxos-and-raft-1b3af91bed80

[18] A. Pagani, "Quic bitcoin: Fast and secure peer-to-peer payments and payment channels," 10 2022, pp. 578–584.

[19] S. Das and L. Ren, "Adaptively secure BLS threshold signatures from DDH and co-CDH," Cryptology ePrint Archive, Paper 2023/1553, 2023. [Online]. Available: https://eprint.iacr.org/2023/1553

[20] "An obvious choice: Why dags over blockchains?" 2023. [Online]. Available: https://blog.iota.org/dags-over-blockchains-iota20/

[21] "Quic." [Online]. Available: https://docs.libp2p.io/concepts/transports/quic/

[22] "Webtransport." [Online]. Available: https://docs.libp2p.io/concepts/transports/webtransport/