Deadline-Aware Causal Broadcast: A Survey of Vector-Based Protocols in Real-Time Distributed Systems

Jinho Ahn
Professor, School of AI Computer Science and Engineering, Kyonggi University, Suwon, Gyeonggi-do, Republic of Korea
doi.org/10.64643/IJIRTV12I12-200644-459

***Abstract*—This paper presents a comprehensive survey of vector-based causal order broadcast protocols from a real-time perspective. We propose a taxonomy that classifies existing approaches based on causality enforcement strategies, waiting mechanisms, metadata overhead, deadline awareness, and scalability. Furthermore, we provide a comparative analysis of representative protocols using the five key metrics. Our analysis reveals that traditional vector-based protocols suffer from inherent limitations in real-time environments due to blocking delays, while relaxed and hybrid approaches improve timeliness at the cost of consistency guarantees. In particular, we emphasize the impact of late messages, which may preserve causality but lose their practical value when deadlines are missed. We further highlight the issue of late messages and outline future directions for adaptive, deadline-aware causal broadcast protocols.**

***Index Terms*—distributed systems, causality, message delivery order, timeliness, scalability.**

I. INTRODUCTION

Causal order broadcast is a fundamental communication abstraction in distributed systems that enforces the preservation of causality among events[4]-[7],[14]. Specifically, if one message causally influences another, any process in the system must observe this relationship in the order of message delivery. Vector-based techniques, particularly vector clocks[2],[3],[15] have been widely adopted to capture such causal dependencies due to their ability to represent partial ordering precisely. Due to this feature, while they provide strong correctness guarantees, the techniques inherently rely on waiting for missing dependencies, which can introduce significant latency[23],[31].

This limitation becomes critical in emerging real-time distributed applications, including satellite systems, autonomous mobilities platforms, and robotic systems, which impose strict timing constraints on message delivery [5],[23],[25],[44],[47]. In these systems, the usefulness of a message is not determined solely by its causal correctness, but also by whether it is delivered within a time window where it remains relevant. This research demand leads to shifting toward relaxing strict causality constraints and incorporating timing-awareness into protocol design. Approaches[5],[23],[25],[44],[47] such as Δ-causal broadcast and deadline-aware delivery mechanisms aim to satisfy the requirement of the timing constraint while preserving acceptable consistency.

The main contributions of this paper are as follows:

- We present a comprehensive survey of vector-based causal order broadcast protocols from a real-time perspective, highlighting the fundamental tension between causality and timeliness.
- We propose a novel taxonomy that classifies existing protocols along key dimensions, including causality enforcement, waiting mechanisms, metadata overhead, deadline awareness, and scalability.
- We provide a comparative evaluation of representative approaches based on the five key metrics, revealing inherent trade-offs.
- We identify open challenges and outline future research directions toward adaptive and deadline-aware causal broadcast protocols, with a particular focus on minimizing late messages in real-time systems.

II. BACKGROUND AND SYSTEM MODEL

Causal relationships in distributed systems are commonly defined using the happened-before relation[1], which captures the partial ordering of events based on communication and local executions. To operationalize this concept, vector clocks are widely used as a mechanism for tracking causal

dependencies[2],[3],[15],[33],[36]. More generally, causal ordering and distributed coordination have been extensively studied in classical distributed systems literature[8],[9],[16],[30], providing the theoretical and algorithmic foundations for modern causal broadcast protocols. Instead of relying on a single global notion of time, each process maintains a vector representing its knowledge of event progression across the system.

When a message is transmitted, the sender attaches its current vector timestamp. Upon receiving a message, the receiver compares the attached timestamp with its local state to determine whether all causally preceding events have been observed. If not, the message must be delayed until the required dependencies are satisfied[6],[7],[14].

Although this approach provides precise causality tracking, it introduces two major limitations. First, the size of vector timestamps grows linearly with the number of processes, resulting in increased communication overhead[31]-[42]. Second, the need to wait for missing dependencies can lead to unbounded delays, particularly in asynchronous or unreliable networks[5],[23],[47].

These challenges become especially problematic in real-time environments, where delayed messages may lose their relevance even if they are eventually delivered in a causally correct order[25],[44].

In this survey, we consider a distributed system model consisting of a set of processes that communicate via message passing over non-FIFO and unreliable channels[12],[24]. Messages may experience bounded delays, reordering, or losses due to network dynamics. Furthermore, we assume that each message is associated with a timing constraint, referred to as a deadline, within which it must be delivered to remain useful for the application[23],[47].

Under this model, a message that arrives after its deadline is considered a late message, even if it satisfies causal ordering constraints. Such late messages may degrade system performance or become obsolete in real-time applications, such as autonomous systems and cyber-physical platforms[47].

Therefore, the primary challenge addressed in this paper is to examine causal broadcast protocols that not only preserve causal consistency but also ensure timely delivery under deadline constraints [5],[23],[25],[44],[47], which may introduces a fundamental trade-off between causality enforcement and timeliness.

III. RELATED WORK

Causal broadcast has been extensively studied as a core abstraction for ensuring consistency in distributed systems[13]. Moreover, causal consistency has been widely applied in modern distributed data systems, such as CRDT-based frameworks and eventually consistent storage systems, where preserving causal relationships is essential for correctness without strong synchronization[21].

Early works[4],[6],[7],[14],[17] primarily focused on enforcing strict causal ordering, often relying on blocking mechanisms that delay message delivery until all dependencies are satisfied. However, while such approaches guarantee correctness, they may introduce significant latency, limiting their applicability in time-sensitive environments[5],[23]. Subsequent research has explored techniques to improve scalability and efficiency. These include effective logical clock variants, hierarchical dependency tracking and compressed representations of causal metadata[18],[19],[26],[27],[32]-[45]. In parallel, gossip-based approaches have been proposed to enhance scalability and robustness[10],[22],[46]. These probabilistic techniques reduce communication overhead and improve fault tolerance but often sacrifice strict causal ordering guarantees.

In addition, Tree-based and topology-aware causal delivery mechanisms have also been proposed to improve efficiency and scalability in structured environments[39].

In real-time distributed systems, the limitations of traditional causal broadcast protocols become more pronounced[5],[23]. Further research has focused on hybrid and adaptive approaches that aim to balance causality and timeliness[25],[44],[47].

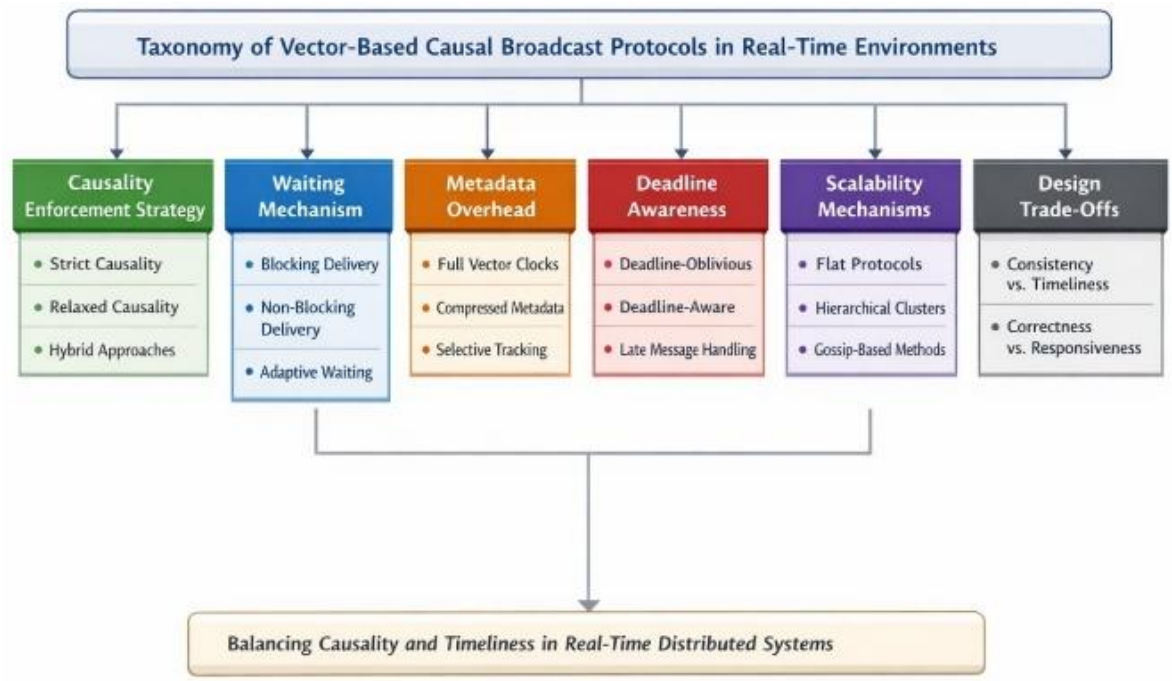

Fig. 1. Taxonomy of vector-based causal broadcast protocols under real-time constraints.

These methods introduce deadline-awareness, selective dependency tracking, and dynamic adjustment of waiting strategies to improve performance in real-time environments.

Despite these advances, existing studies and surveys primarily emphasize causality preservation and scalability, with limited attention to deadline-constrained environments and the impact of late messages[47]. Unlike them, this work emphasizes the impact of real-time constraints on causal broadcast protocols. In particular, we highlight the problem of late messages, which may still preserve causality but lose their practical value due to missed deadlines[47]. This perspective shifts the focus from correctness alone to usefulness in real-time decision-making systems, such as autonomous mobility and cyber-physical applications.

IV. TAXONOMY OF VECTOR-BASED PROTOCOLS

To systematically analyze vector-based causal order broadcast protocols in real-time environments, we propose a taxonomy that classifies existing approaches along several key design dimensions like in Fig. 1. These dimensions capture the fundamental trade-offs between causality enforcement and delivery timeliness, which are critical in real-time distributed systems.

A. Causality Enforcement Strategy

Vector-based protocols differ in how strictly they enforce causal ordering constraints, and this design choice fundamentally determines the trade-off between consistency guarantees and delivery latency. Strict causality enforcement requires that a message be delivered only after all its causal predecessors have been delivered[6],[7]. This approach preserves full causal consistency equivalent to Lamport's happened-before relation[1], ensuring correctness under all conditions. In practice, such guarantees are typically achieved using vector clock–based dependency tracking and acknowledgment-based delivery mechanisms[4],[6],[7],[14],[17]. However, it may incur significant delays due to blocking on missing dependencies, especially in networks with variable latency or loss[5],[23].

Relaxed causality enforcement, in contrast, intentionally weakens the ordering guarantees by allowing bounded violations of causal order. For example, Δ-causal broadcasting[5] permits a message to be delivered even if some causal predecessors are missing, as long as the violation remains within a predefined temporal or logical bound (Δ). The key characteristic of relaxed approaches is that the relaxation policy is fixed and predefined: the system always operates under the same weakened consistency model regardless of runtime conditions. As a result, latency is reduced and timeliness is improved, particularly in environments with high delay variability or

intermittent connectivity, but at the cost of potentially observable causal inconsistencies[25],[47].

Hybrid approaches[23],[25],[44],[47] go a step further by introducing adaptive enforcement mechanisms that dynamically adjust the level of causality enforcement at runtime. Unlike relaxed approaches, hybrid protocols do not commit to a single fixed relaxation policy. Instead, they switch between strict and relaxed behaviors (or interpolate between them) based on system conditions such as network delay, message loss, buffer occupancy, or application-level deadlines. For instance, a hybrid protocol may enforce strict causality under normal conditions but temporarily relax ordering constraints when delays exceed a threshold, or when deadlines are at risk of being violated. This enables the system to maintain strong consistency when feasible while still providing low-latency delivery under adverse conditions.

The critical distinction is therefore that relaxed approaches provide static, bounded weakening of causality, whereas hybrid approaches provide dynamic, context-aware adaptation of causality enforcement. This difference has important implications: relaxed protocols offer predictability and simpler reasoning about violations, while hybrid protocols offer greater flexibility and performance optimization at the cost of increased system complexity.

This dimension directly impacts both correctness and responsiveness in real-time systems, making it a central consideration in the design of vector-based causal broadcast protocols.

B. Waiting Mechanism

Most vector-based protocols rely on waiting conditions to ensure causal consistency by delaying message delivery until all required dependencies are satisfied[23],[47]. The design of this waiting mechanism directly affects both latency and the ability to meet real-time deadlines. While blocking delivery guarantees correctness, it may result in unbounded waiting time, particularly under network delays, message loss, or straggler effects[25],[44].

Non-blocking delivery fundamentally changes this paradigm by eliminating or minimizing waiting at delivery time[48]. Messages are delivered as soon as they arrive, even if some causal dependencies are not yet satisfied. To handle the resulting inconsistencies, non-blocking approaches typically rely on post-delivery compensation mechanisms. This design achieves very low latency and high responsiveness but may introduce additional complexity in maintaining consistency after the fact.

Adaptive waiting strategies, in contrast, do not eliminate waiting altogether but instead dynamically control how much to wait before delivering a message[23],[25],[44],[47]. Unlike non-blocking approaches, adaptive strategies still treat waiting as a core mechanism for preserving causality, but they selectively relax waiting conditions based on runtime factors such as message deadlines, network congestion, or buffer occupancy[47]. For example, a protocol may wait for missing dependencies up to a certain deadline, after which it delivers the message even if some dependencies are unresolved. In this sense, adaptive waiting can be seen as a controlled and context-aware relaxation of blocking behavior, rather than a complete shift to post-hoc correction. The critical distinction is therefore that non-blocking delivery removes waiting from the critical path and shifts consistency handling to after delivery, whereas adaptive waiting strategies retain waiting as a primary mechanism but adjust its duration and strictness dynamically.

The choice of waiting mechanism thus plays a crucial role in determining both delivery timeliness and the complexity of maintaining causal consistency in real-time systems.

C. Metadata Overhead

Vector clocks provide a precise mechanism for representing causal dependencies, but they introduce communication and storage overhead proportional to the number of processes in the system[8],[9],[30]. In large-scale distributed environments, this metadata cost becomes a major scalability bottleneck, making metadata reduction a central design concern in vector-based protocols[15],[16],[26].

Full vector clocks allow exact evaluation of delivery conditions, but incur O(N) metadata overhead per message, where N is the number of processes[15],[16]. As the system grows, both the message size and the local state required for dependency tracking increase linearly, which can significantly degrade scalability.

Compressed representations[15],[39],[42] aim to reduce this overhead by encoding essentially the same broad dependency space in a more compact form. Instead of explicitly listing every process's logical time, these approaches exploit structural regularities in causality information. Examples include interval-based vector clocks, which summarize contiguous logical histories as

intervals[42], and hierarchical vector clocks, which aggregate dependency information according to process groups or communication structure[15],[39]. The key idea is that compressed representations still attempt to preserve a system-wide view of causality, but they do so through more efficient encoding. In other words, the set of potentially relevant dependencies remains broad, yet its representation is compacted to reduce storage and transmission cost[15],[39],[42]. This may preserve exactness in some cases, or only partial dependency information in others, depending on the compression scheme.

Selective dependency tracking, by contrast, reduces metadata not primarily by compressing the encoding of dependency information, but by tracking fewer dependencies in the first place[40],[46]. Rather than maintaining causal relations with respect to all processes, the protocol records only those dependencies deemed relevant to correctness or application semantics. Relevance may be determined by communication locality, group membership, topic-based interaction, data sharing patterns, or application-defined influence relations. For example, if a process only causally interacts with a limited subset of other processes, selective tracking may ignore unrelated processes entirely[40]. Thus, unlike compressed representations, which retain a broad causality scope but encode it more efficiently, selective dependency tracking shrinks the scope of causality being represented.

The crucial distinction is therefore that compressed representations answer the question "how can we encode dependency information more compactly?", whereas selective dependency tracking answers the question "which dependency information is actually necessary to track?". As a result, the former is primarily an encoding optimization, while the latter is a semantic or scope reduction strategy.

This distinction has important implications. Compressed representations typically retain stronger compatibility with traditional causality semantics, but may still face limitations in very large or highly dynamic systems. Selective dependency tracking can achieve much greater metadata reduction, especially in sparse communication patterns, but may sacrifice completeness of causal knowledge outside the selected dependency set. Therefore, the choice between these approaches reflects a fundamental trade-off between causal precision, scalability, and application relevance.

D. Deadline Awareness

A key distinguishing factor in real-time environments is whether a protocol explicitly incorporates timing constraints into its delivery logic. In such systems, correctness is not determined solely by causal consistency, but also by whether messages are delivered early enough to remain useful to the application[5],[23],[25],[44],[47].

Deadline-oblivious protocols[4],[6],[7],[14],[17] focus exclusively on preserving causal relationships without considering message deadlines or temporal validity. Their primary objective is to ensure that messages are delivered in a causally correct order, regardless of whether the delivery occurs before or after the application's useful time window. As a result, such protocols may preserve logical correctness while still failing to meet the practical requirements of real-time applications, where late delivery can be functionally equivalent to message loss[5],[23].

Deadline-aware protocols[5],[23] explicitly incorporate timing constraints into delivery decisions. In addition to checking causal dependencies, these protocols consider whether a message can still meet its deadline if it continues to wait for missing predecessors. Based on this evaluation, the protocol may prioritize urgent messages, drop messages that are unlikely to arrive in time, or relax waiting conditions when strict causal delivery would cause deadline violations[28],[44].

The defining property of deadline-aware approaches is that deadline information is part of the protocol logic itself. However, in many such protocols, the way deadlines affect behavior is still governed by fixed rules or predetermined policies. For example, a message may be dropped whenever its slack time becomes negative, or delivered early whenever the remaining time falls below a predefined threshold[5]. In this sense, deadline-aware protocols are timing-sensitive, but their response to deadlines is often static and rule-based.

Adaptive deadline-aware approaches[25],[44],[47] extend this idea by introducing dynamic, context-sensitive adjustment mechanisms that change protocol behavior at runtime in response to both timing constraints and system conditions. Rather than merely applying fixed deadline rules, these approaches attempt to optimize the usefulness of delivered messages under changing environments, such as varying network delay, message contention, system load, or fluctuations in causal dependency resolution time[44]. For instance, an adaptive

deadline-aware protocol may tighten causal enforcement when the system is lightly loaded and deadlines are easily met, but progressively relax waiting conditions, change prioritization policies, or alter drop thresholds when congestion increases or when many messages approach expiration.

The critical distinction is therefore that deadline-aware protocols incorporate deadlines into decision making, whereas adaptive deadline-aware approaches dynamically change how deadlines are handled in order to maximize effective or useful delivery under runtime uncertainty[25],[44],[47]. Put differently, deadline-aware protocols answer the question, "Should timing constraints influence delivery decisions?"; adaptive deadline-aware approaches answer the further question, "How should the protocol continuously adjust its behavior as timing conditions and system state evolve?"

This distinction is important because simply being aware of deadlines does not guarantee robust real-time performance. A fixed deadline-aware policy may work well under expected conditions but degrade under highly dynamic workloads. Adaptive deadline-aware approaches seek to overcome this limitation by making the protocol responsive to environmental variation, thereby improving deadline satisfaction, reducing wasted transmissions, and increasing the practical value of delivered messages[25],[44],[47].

This dimension is particularly important in real-time causal broadcast, because it determines whether the protocol merely recognizes message deadlines or actively adapts its causality enforcement and delivery policy to preserve application-level usefulness under timing uncertainty.

E. Scalability Mechanisms

As the number of processes increases, scalability becomes a critical concern in vector-based causal broadcast protocols[8],[9]. The cost of maintaining, transmitting, and interpreting causal metadata grows with system size, and coordination overhead may become prohibitive in large or highly dynamic distributed environments[19],[25]. Accordingly, scalability mechanisms determine not only performance efficiency but also the practical applicability of a protocol to wide-area or large-scale systems[16],[30].

Flat vector-based protocols[5],[23],[44],[47] maintain causal dependency information across all processes in a single logical space. Although this design offers a direct and precise representation of causality, it scales poorly because each process must conceptually remain aware of the global process set. As the system grows, both metadata overhead and dependency management complexity increase significantly.

Hierarchical approaches[19],[25] address this limitation by structuring the system into multiple levels of organization, typically by grouping processes into clusters, regions, or domains. Within each local group, causality may still be tracked with relatively fine granularity, while inter-group communication is summarized, aggregated, or mediated through higher-level coordinators[37]. The central idea is to improve scalability by reducing the scope of direct dependency management at each level through explicit structural organization. In other words, hierarchical approaches preserve a relatively systematic model of causality, but they do so by partitioning the system and managing dependencies differently at local and global levels[38]. This can substantially reduce metadata size, communication overhead, and coordination complexity, especially in systems where communication is naturally clustered.

Decentralized and gossip-based techniques[46],[48], in contrast, improve scalability not primarily through explicit structural layering, but through distributed dissemination and local interaction rules. Rather than relying on a well-defined hierarchy or cluster coordinators, these approaches spread dependency or state information incrementally across the system through peer-to-peer exchanges, epidemic dissemination, or probabilistic propagation[10]. Their key advantage is that they avoid centralized bottlenecks and can remain robust under churn, partial failures, and dynamic membership[22]. However, this benefit often comes at the cost of weaker guarantees, such as delayed convergence, incomplete global knowledge, redundant message propagation, or only probabilistic assurance that all relevant dependency information will eventually reach the necessary processes[22],[46].

From a taxonomy perspective, these two classes should not be treated as minor variations of the same idea. Although both aim to improve scalability beyond flat vector-based designs, they do so through fundamentally different design philosophies: one through architectural structuring, and the other through distributed self-organization. Consequently, they differ not only in overhead characteristics, but also in fault tolerance, dependency visibility, and the strength of the guarantees they can provide.

F. Summary of Design Trade-offs

The proposed taxonomy highlights that no single protocol can simultaneously optimize all design dimensions. Instead, vector-based causal broadcast protocols must navigate a set of inherent trade-offs spanning causality enforcement, waiting mechanisms, metadata overhead, deadline awareness, and scalability mechanisms.

A particularly important insight emerges from the interaction between causality enforcement and deadline awareness. Ensuring strict causal ordering often conflicts with timely message delivery, especially under real-time constraints. While deadline-aware protocols incorporate timing considerations into delivery decisions, adaptive deadline-aware approaches further attempt to dynamically balance causal consistency and timeliness based on runtime conditions. Nevertheless, this balance remains fundamentally constrained: a message that is perfectly ordered but delivered too late may have no practical value, whereas a timely message may require relaxing causal guarantees.

Taken together, these dimensions illustrate that protocol design in real-time causal broadcast is not about achieving absolute optimality, but about carefully selecting trade-offs that align with application requirements, such as consistency sensitivity, latency tolerance, system scale, and fault tolerance needs.

This taxonomy therefore serves as a foundation for the comparative analysis presented in the following section, enabling a structured and multidimensional evaluation of vector-based causal broadcast protocols under real-time constraints.

V. PROTOCOL ANALYSIS

This section presents a comprehensive analysis of representative vector-based causal order broadcast protocols under real-time constraints. Building upon the taxonomy introduced in the previous section, we evaluate existing approaches along the five key design dimensions.

Recent advances in real-time distributed systems[28],[29]—particularly in edge computing, IoT, and cyber-physical systems—have introduced stringent latency requirements, dynamic network conditions, and resource constraints. Consequently, recent protocols increasingly incorporate relaxed consistency models, adaptive control mechanisms, and lightweight metadata representations to balance correctness and timeliness[5],[23],[25],[44],[47].

A. Strict Causality-Based Protocols

Traditional vector clock-based protocols [4],[6],[7],[14],[17] prioritize correctness by ensuring that all dependencies are satisfied before delivery with the following features.

Analysis:

- Causality Enforcement: Strict
- Waiting Mechanism: Blocking
- Metadata Overhead: O(N)
- Deadline Awareness: Absent
- Scalability: Limited

While these protocols guarantee strong consistency, this often results in significant delays, particularly in environments with unpredictable network behavior [23],[31]. In real-time environments, such delays frequently result in deadline violations, as messages are held back waiting for missing dependencies[5],[23].

Recent studies indicate that strict causality enforcement is often impractical in time-sensitive applications such as real-time streaming, vehicular networks, and interactive systems, where delayed messages rapidly lose their utility[25],[47].

B. Δ-Causal Protocols

To mitigate latency, recent research[5] has introduced relaxed causality models, most notably Δ-causal broadcast, which permits bounded violations of causal order, with the following features.

Analysis:

- Causality Enforcement: Relaxed (bounded violations)
- Waiting Mechanism: Deadline-bounded waiting
- Metadata Overhead: O(N)
- Deadline Awareness: Explicit
- Scalability: Moderately improved

In these approaches, a message can be delivered if its missing dependencies are expected to arrive within a bounded time window Δ. If dependencies do not arrive within Δ, the system may proceed with delivery to preserve timeliness. This significantly reduces delivery latency and improves deadline satisfaction[25],[44], making such protocols well-suited for multimedia streaming systems, real-time

collaborative applications, and edge computing environments.
However, these benefits come at the cost of weakened consistency guarantees, as causal ordering may be temporarily violated[23].

C. Adaptive Deadline-Aware Protocols
Recent work[23],[25],[44],[47] increasingly focuses on adaptive protocols that dynamically adjust causality enforcement and waiting strategies based on runtime conditions.

Analysis:
- Causality Enforcement: Hybrid (dynamic)
- Waiting Mechanism: Adaptive (blocking ↔ non-blocking)
- Metadata Overhead: Moderate
- Deadline Awareness: Central
- Scalability: Moderate to high

These protocols monitor system conditions such as network delay, message loss rate, and workload intensity. Based on these observations, they switch between strict and relaxed modes as follows.
- Under stable conditions → strict causality is enforced
- Under high latency or congestion → relaxed delivery is allowed

This adaptability enables a more effective trade-off between consistency and timeliness, which is critical in modern applications such as autonomous systems, smart mobility, and industrial IoT. These approaches reflect a critical shift toward value-aware delivery, where the usefulness of information depends on its timeliness rather than strict ordering correctness.

D. Metadata Optimization Techniques
A major limitation of vector-based protocols is the O(N) metadata overhead, which restricts scalability in large systems[15],[42]. Recent works[40],[46] aim to reduce metadata size through various optimization techniques.

Analysis:
- Causality Enforcement: Varies
- Waiting Mechanism: Varies
- Metadata Overhead: Reduced (often sub-linear)
- Deadline Awareness: Partial
- Scalability: Significantly improved

In particular, selective tracking maintains only essential dependency information.

E. Scalability-Oriented Approaches
Scalability is a crucial requirement in large-scale distributed systems.

1) Hierarchical Approaches[19],[25]
These organize processes into multi-level structures, where causality is managed within and across clusters.

2) Decentralized and Gossip-Based Approaches[46]
These rely on peer-to-peer dissemination without centralized coordination, often providing probabilistic causality guarantees.

Therefore, hierarchical approaches provide stronger consistency but introduce structural overhead whereas gossip-based approaches achieve superior scalability and low latency but offer weaker, probabilistic guarantees.

Table 1. Key characteristics of different categories of vector-based protocols

Protocol Category	Causality Enforcement	Waiting Mechanism	Metadata Overhead	Deadline Awareness	Scalability
Strict Protocols	Strict	Blocking	O(N)	No	Low
Δ-Causal Protocols	Relaxed	Deadline-bounded	O(N)	Yes	Medium
Adaptive Protocols	Hybrid	Adaptive	Moderate	Yes	Medium–High
Metadata-Optimized Protocols	Varies	Varies	Reduced	Partial	High

F. Summary of Protocol Trends

Recent trends in vector-based real-time causal broadcast protocols can be summarized as follows:

- A shift from strict → relaxed → adaptive causality enforcement
- Increasing importance of deadline-aware design
- Emphasis on metadata reduction for scalability
- Preference for timeliness over strict consistency in real-time environments

Overall, there is a fundamental paradigm shift: ensuring timely and meaningful message delivery is often more critical than preserving perfect causal ordering. This observation is particularly important in real-time systems, where late messages may be causally correct but practically useless.

VI. COMPARATIVE ANALYSIS

This section provides a comparative evaluation of vector-based causal order broadcast protocols under real-time constraints. Based on the taxonomy and protocol analysis presented in the previous sections, we systematically compare representative approaches across the key design dimensions. In addition, we analyze their effectiveness in achieving timely message delivery and minimizing late message occurrences.

A. Comparison Across Design Dimensions

Like in Table 1, several key observations emerge. First, strict protocols[4],[6],[7],[14],[17] provide the strongest consistency guarantees, but their blocking nature results in high latency and poor deadline satisfaction[5]. This makes them unsuitable for real-time environments where timeliness is critical[23].

Second, Δ-causal protocols[5] significantly improve timeliness by relaxing causality constraints. However, their performance depends heavily on the choice of Δ, and inappropriate parameter settings may either degrade consistency or fail to meet timing requirements.

Third, adaptive deadline-aware protocols [23],[25],[44],[47] offer a balanced approach by dynamically adjusting their behavior according to system conditions. This enables better performance across varying workloads and network environments, making them highly suitable for recent deadline-constrained networked systems. In addition, they can significantly reduce the number of late or obsolete messages. However, those often come at the cost of relaxed consistency guarantees and moderate metadata overhead.

Lastly, metadata-optimized protocols[40],[46] focus on reducing communication overhead by compressing or selectively maintaining dependency information. While this improves scalability and efficiency, it may introduce variability in causality enforcement and waiting behavior, and often provides only partial support for deadline awareness.

B. Limitations of Existing Approaches

In real-time environments, two performance metrics, message delivery timeliness and late message ratio are particularly critical[47]. Existing approaches primarily focus on reducing waiting time, relaxing causality constraints, and incorporating deadline awareness[5],[23],[25],[44]. Despite the significant progress, current protocols exhibit several limitations as follows. First, due to lack of proactive delivery optimization, most protocols focus on reducing waiting time rather than actively advancing delivery timing[47]. Second, late messages are typically handled after they occur (e.g., by discarding), rather than being prevented[47]. Third, many approaches rely on fixed parameters (e.g., Δ), which may not adapt well to dynamic environments[23],[25],[44]. Lastly, existing designs often operate at fixed points in the consistency–timeliness spectrum, limiting flexibility.

C. Future Research Directions

The comparative analysis highlights several important directions for future protocol design:

- Incorporating proactive delivery mechanisms to reduce latency beyond simple waiting reduction.
- Minimizing late messages through predictive or adaptive techniques.
- Designing protocols that dynamically balance consistency and timeliness.
- Reducing metadata overhead while preserving sufficient causality information.

These observations motivate the need for a new class of vector-based real-time causal broadcast protocols that go beyond existing approaches.

VII. CONCLUSION

This survey highlights a fundamental shift in the design of causal broadcast protocols for real-time

systems. While traditional approaches emphasize strict correctness, modern applications require a more nuanced perspective that accounts for timeliness and practical usefulness.

Our analysis shows that no single protocol can simultaneously optimize all design dimensions. Instead, effective solutions must carefully balance trade-offs between causality, latency, scalability, and deadline-awareness.

In particular, the notion of late messages introduces a critical challenge: a message that is causally correct but delivered too late may have no value in real-time applications and degrade system performance.

Future research should therefore focus on adaptive and value-aware protocols that dynamically optimize delivery decisions under varying system conditions.

REFERENCES

[1] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," Communications of the ACM, vol. 21, No. 7, July 1978, pp. 558-565.

[2] F. Mattern, "Virtual Time and Global States of Distributed Systems," in Parallel and Distributed Algorithms, M. Cosnard and P. Quinton, eds., North-Holland, Amsterdam, 1989, pp. 215-226.

[3] C. Fidge, Dynamic Analysis of Event Orderings in Message-Passing Systems, doctoral dissertation, Australian Nat'l Univ., 1989.

[4] R. Baldoni, A. Mostefaoui, and M. Raynal, "A positive acknowledgment protocol for causal broadcasting," IEEE Trans. Computers, vol. 47, no. 12, pp. 1341–1350, 1998.

[5] R. Baldoni, A. Mostefaoui, and M. Raynal, "Causal delivery of messages with real-time data in unreliable networks," Real-Time Systems, vol. 10, no. 3, pp. 245–262, 1996.

[6] K. Birman and T. Joseph, "Reliable communication in the presence of failures," ACM Trans. Computer Systems, vol. 5, no. 1, pp. 47–76, 1987.

[7] K. Birman, "The process group approach to reliable distributed computing," Communications of the ACM, vol. 36, no. 12, pp. 37–53, 1993.

[8] M. Raynal, Distributed Algorithms for Message-Passing Systems, Springer, 2013.

[9] G. Coulouris et al., J. Dollimore, T. Kindberg, and G. Blair, Distributed Systems: Concepts and Design, 5th ed. Addison-Wesley, 2011.

[10] A. Demers, D. Greene, C. Houser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," ACM SIGOPS Operating Systems Review, vol. 22, no. 1, 1988, pp. 8-32.

[11] S. Floyd, V. Jacobson, C. Liu, S. McCanne and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," IEEE/ACM Trans. Networking, vol. 5, no. 6, 1997, pp. 784–803.

[12] V. Hadzilacos and S. Toueg, "Fault-tolerant broadcasts and related problems," Distributed Systems, 2nd Ed., 1993, pp. 97-145.

[13] N. Lynch, Distributed Algorithms. Morgan Kaufmann, 1996.

[14] A. Schiper, J. Eggli, and A. Sandoz, "A new algorithm to implement causal ordering," Proc. of the 3rd International Workshop on Distributed Algorithms, LNCS, vol 392. Springer, Berlin, Heidelberg 1989, pp. 219–232.

[15] M. Singhal and A. Kshemkalyani, "An efficient implementation of vector clocks," Information Processing Letters, vol. 43, no. 1, 1992, pp. 47–52.

[16] A. Kshemkalyani and M. Singhal, Distributed Computing: Principles, Algorithms, and Systems. Cambridge Univ. Press, 2008.

[17] K. Birman, A. Schiper, and P. Stephenson, "Lightweight causal and atomic group multicast," ACM Transactions on Computer Systems, vol. 9, no. 3, 1991, pp. 272–314.

[18] I. Muñoz-Fernández, S. Arévalo-Viñuales, and P. de-las-Heras-Quirós, "Timestamp system for causal broadcast communication," Journal of Supercomputing, 2024, vol. 80, no. 13, pp. 18728-18760.

[19] T. Hou, C. Yang, and K. Chen, "Adaptive message scheduling for supporting causal ordering in wide-area group communications," Journal of Systems and Software, vol. 72, no. 3, 2004, pp. 321-333.

[20] M. Letia, N. Preguiça, and M. Shapiro, "Consistency without concurrency control in large, dynamic systems," Proc. of the 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware, Big Sky, MT, USA, 2009, pp. 29-34.

[21] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, “Conflict-free replicated data types,” Proc. of the 13th international conference on Stabilization, safety, and security of distributed systems, 2011, pp. 386-400.

[22] P. Eugster, R. Guerraoui, A. Kermarrec, and L. Massoulié, “Epidemic information dissemination in distributed systems,” IEEE Computer, vol. 37, no. 5, 2004, pp. 60–67.

[23] L. Rodrigues, R. Baldoni, E. Anceaume, and M. Raynal, “Deadline-constrained causal order,” Proc. of the Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2000, pp. 234-241.

[24] T. Chandra and S. Toueg, “Unreliable failure detectors for reliable distributed systems,” J. ACM, vol. 43, no. 2, 1996, pp. 225–267.

[25] A. Benslimane and A. Abouaissa, “Dynamical grouping model for distributed real time causal ordering,” Computer Communications, vol. 25, no. 3, 2002, pp. 288-302.

[26] C. Benzaid and N. Badache, "An Optimal Causal Broadcast Protocol in Mobile Dynamic Groups," 2008 IEEE International Symposium on Parallel and Distributed Processing with Applications, Sydney, NSW, Australia, 2008, pp. 477-484.

[27] S. Kulkarni, M. Demirbas, D. Madappa, and B. Avva, M. Leone, “Logical Physical Clocks,” Proc. of the International Conference on Principles of Distributed Systems, Lecture Notes in Computer Science, vol. 8878. Springer, Cham, 2014, pp. 17–32.

[28] J. Zhang, H. Shi, Y. Cui, F. Qian, W. Wang, and K. Zheng, “To Punctuality and Beyond: Meeting Application Deadlines with DTP,” Proc. of the IEEE 30th International Conference on Network Protocols (ICNP), Lexington, KY, USA, 2022, pp. 1-11.

[29] G. Daneels, E. Municio, K. Spaey, G. Vandewiele, A. Dejonghe, F. Ongenae, S. Latré, and J. Famaey, “Real-Time data dissemination and analytics platform for challenging IoT environments,” Proc. of the Global Information Infrastructure and Networking Symposium (GIIS), Saint Pierre, France, 2017, pp. 23-30.

[30] H. Attiya and J. Welch, Distributed Computing: Fundamentals, Simulations, and Advanced Topics, 2nd ed. Wiley, 2004.

[31] D. Wilhelm, Causal broadcast algorithms for dynamic distributed systems, Distributed, Parallel, and Cluster Computing [cs.DC], Sorbonne Université, 2023.

[32] M. Tyulenev, A. Schwerin, A. Kamsky, R. Tan, A. Cabral, and J. Mulrow, “Implementation of Cluster-wide Logical Clock and Causal Consistency in MongoDB,” Proc. of the 2019 International Conference on Management of Data, 2019, pp. 636-650.

[33] A. Kshemkalyani, M. Shen, and B. Voleti, “Prime Clock: Encoded Vector Clock to Characterize Causality in Distributed Systems,” Journal of Parallel and Distributed Computing, vol. 140, 2020, pp. 37-51.

[34] S. Kulkarni, G. Appleton, and D. Nguyen, “Achieving Causality with Physical Clocks,” Proc. of the 23rd International Conference on Distributed Computing and Networking, ICDCN 2022, New York, USA, 2022, pp. 97-106.

[35] A. Kshemkalyani, A. Khokhar, and M. Shen, “Vector Clock: Using Primes to Characterize Causality in Distributed Systems,” Proc. of the 19th International Conference on Distributed Computing and Networking. Varanasi India, 2018, pp. 1-8.

[36] A. Kshemkalyani and A. Misra, “The Bloom Clock to Characterize Causality in Distributed Systems,” Advances in Networked-Based Information Systems. Ed. by Leonard Barolli et al. 1264. Cham: Springer International Publishing, 2021, pp. 269-279.

[37] G. Evropeytsev, E. Domínguez, S. Hernandez, M. Trinidad, and J. Cruz, “An efficient causal group communication protocol for P2P hierarchical overlay networks,” Journal of Parallel and Distributed Computing, vol. 102, 2017, pp. 149-162.

[38] G. Evropeytsev, E. Domínguez, S. Hernandez, and J. Cruz, “An Efficient Causal Group Communication Protocol for Free Scale Peer-to-Peer Networks,” Applied Sciences, vol. 6, no. 9, 2016, p. 234.

[39] S. Blessing, S. Clebsch, and S. Drossopoulou, “Tree topologies for causal message delivery,” Proc. of the 7th ACM SIGPLAN International Workshop on Programming Based on Actors, Agents, and Decentralized Control, 2017, pp. 1–10.

[40] V. Santos and L. Rodrigues, “Localized Reliable Causal Multicast,” Proc. of the IEEE 18th International Symposium on Network

Computing and Applications (NCA), Cambridge, MA, USA, 2019, pp. 1-10.

[41] J. Araujo, L. Arantes, E. Júnior, L. Rodrigues, and P. Sens, "A Communication-Efficient Causal Broadcast Protocol," Proc. of the 47th International Conference on Parallel Processing, 74, 2018, pp. 1-10.

[42] P. Almeida, C. Baquero, and V. Fonte, "Interval Tree Clocks: A Logical Clock for Dynamic Systems," Proc. of the 12th International Conference on Principles of Distributed Systems, vol. 5401, 2008, pp. 259-274.

[43] M. Costea, R. Ciobanu, R. Marin, C. Dobre, C. Mavromoustakis, and G. Mastorakis, "Causal and Total Order in Opportunistic Networks," IGI Global. chapter Emerging Innovations in Wireless Networks and Broadband Technologies, 2016, pp. 221–262.

[44] F. Guidec, P. Launay, and Y. Mahéo, "Causal and Δ-causal broadcast in opportunistic networks," Future Generation Computer Systems, vol. 118, 2021, pp. 142-156.

[45] A. Mostefaoui, M. Perrin, M. Raynal, and C. Jiannong, "Crash-Tolerant Causal Broadcast in O(n) Messages," Information Processing Letters, vol. 151, 105837, 2019.

[46] A. Mostefaoui, and S. Weiss, "Probabilistic Causal Message Ordering," Proc. of the 14th International Conference on Parallel Computing Technologies (PaCT 2017), Springer, Nizhni Novgorod, Russia, 2017, pp. 315–326.

[47] Jinho Ahn, "Large-scale Deadline-Constrained Causal Order Broadcast Algorithm for Enhancing Message Delivery Success Rate Performance," International Journal of Control and Automation, vol. 11, no. 4, 2018, pp.1-10.

[48] B. Nédelec, P. Molli and A. Mostéfaoui, "Breaking the Scalability Barrier of Causal Broadcast for Large and Dynamic Systems," Proc. of the IEEE 37th Symposium on Reliable Distributed Systems (SRDS), Salvador, Brazil, 2018, pp. 51-60.