International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 12-No.1 (2026) pp. 884-891
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Graph-Based Duplicate Trade Detection and Idempotency Framework Implementation in Distributed Electronic Trading Systems

Iswarya Konasani*

Independent Researcher, USA
* **Corresponding Author Email:** iswaryakonasani5@gmail.com- **ORCID:** 0000-0002-0047-5550

Article Info:

DOI: 10.22399/ijcesen.4940
Received : 12 February 2026
Revised : 15 February 2026
Accepted :19 February 2026

Keywords

Duplicate Trade Detection,
Idempotency Framework,
Blockchain Trade Modeling,
Message Fingerprinting,
Temporal Correlation

Abstract:

To prevent the reprocessing of the same trade message in different distributed financial infrastructures, electronic trading systems must have powerful duplicate trade detection protocols. Redundant messages are a result of network timeouts, TCP retransmission protocols, upstream retry queues, and manual resubmission workflows that are part of heterogeneous trading structures. Idempotency models define message uniqueness by using composite business keys, cryptographic fingerprints using the SHA-256 hashing functions, and deduplication logic on time windows that trades off between accuracy of detection and scalability of computation. Graphed graph frameworks are enhanced with blockchain and deliver distributed data models to specify intricate trade relations in the form of immutable ledger records, smart contract validation logic, and multi-channel designs, which assure information integrity across trading networks. Multi-channel correlation algorithms differentiate between actual trade amendments and replay events based on machine learning classification models and partial fill cases and cross-venue execution strategies. Strategies of implementation are used to optimize parameters of tolerance windows with the use of hierarchical composite key matching, progressive sampled indexing, and container-based pre-fetching strategies. Microsecond-latency duplicate-detection In-memory caching architectures in conjunction with Bloom filter probabilistic structures can achieve duplicate detection at millions of trade messages per day to protect downstream risk management and regulatory reporting systems against position inflation and compliance violations.

1. Introduction to Duplicate Trade Detection in Modern Trading Infrastructure

Millions of trade messages are transmitted by electronic trading systems in modern financial markets in a variety of asset classes, across a variety of execution venues, and communication protocols every day. Duplicate trade detection is an essential control process that detects and eliminates incorrect processing of a trade message by a downstream risk management system, position-keeping system, and accounting system several times. The technical issue of replay is brought about by the distributed nature of modern trading structure with messages traversing across various system boundaries such as order management systems, execution management systems, middleware layers, and connectivity gateways to the venue. The speedy shift to real-time and event-driven microservices has made transactions grow by far, and today, it is estimated that the number of transactions will grow even further, reaching more than 10 trillion by 2025, and that the risk of failing to process a duplicate message will rise dramatically [1].

The impact on the detection controls when trade messages are duplicated, which then pass to the core processing systems, is not just a mere inconvenience in operating the systems. Systematic replays that are undetected overstate trading positions with the creation of artificial long or short exposures that do not reflect the actual market risk of the firm. Repeat executions should not be recorded in the same book of accounts as the same economic activity due to the resultant corruption of the financial profit and loss accounts, resulting in exaggerated or underrepresented performance reports. It has been estimated in research that poor-quality data in financial services leads to a large systemic risk, with an average failed or duplicate payment resulting in an average of 12 dollars in

reprocessing and customer support [1]. Transparency regimes that encompass regulatory reporting frameworks have the risk of submitting the same trade records to the supervisory authorities, which could lead to the impact of investigations and financial penalties related to the same.

The complexity of the technical aspects of duplicate detection increases in conditions with diverse system integration patterns. Trading companies usually have hybrid frameworks in which old mainframes are present alongside contemporary microservices, FIX connections are present alongside RESTful APIs, and real-time streaming data combines with batch file transfer. Baseline levels of 0.7% duplicates of requests into the network are introduced by network unreliability, and systems must respond within 100 milliseconds or less to satisfy consumer requirements [1]. Both communication channels have different failure modes that may cause retransmission of messages, and in this situation, different detection logics that have the ability to identify replays irrespective of their route to the receiver and their timing distributions with regard to processing windows are required.

Modern strategies to replicate trade detection are becoming more and more based on the use of unique identifiers of transactions, operationalized using idempotency keys, to provide exact-once semantics in distributed contexts [1]. It is these mechanisms that allow systems to recognize repeat requests and provide corresponding original responses rather than perform new financial operations, which ultimately forms a sort of magic shield that shields downstream processing (conversely) against the underlying anarchy of distributed communication networks.

2. Message Replay Scenarios and Root Cause Analysis

Various technical failure conditions arising from distributed computing architecture and the network communication protocol give rise to message replay events in trading systems. Network timeout conditions are the most common replay trigger, where the upstream systems send trade messages but do not obtain acknowledgment messages within set time limits. The encrypted deduplication systems have shown that the deterministic encryption exposes underlying frequency distributions with experiments in the real world showing that 99.8 percent of chunks are replicated less than 100 times, and the experiment shows that 30 out of 41 million chunks have a frequency greater than 10,000 [2]. In normal retry logic, the originating system believes it has lost the message and sends the same message, while the downstream system may have consumed the original message without receiving the acknowledgment packet.

The TCP-level retransmission systems, though necessary to guarantee reliable data delivery, add more complexity to replay detection. In case message streams are interrupted by network congestion or routing failures, the lower protocol layers will automatically retransmit sequences of packets without knowledge to the application layer. Connection health applications can even start their own recovery retry sequences at the same time, leading to multiple copies of a message being received via automated protocol recovery and explicit application retry logic. State-of-the-art detection systems have processing rates of up to 150,000 transactions per second with less than 3 ms latency, indicating the time scales within which replay detection must be done [3].

Persistent retry queue Persistent retry queues are often used in order management and execution management systems to make sure messages are delivered eventually, even if there is a temporary loss of connection. Multi-layer-based detection frameworks achieve data ingestion-layer processing latency of 0.5-1.2 ms, throughput of 100,000 events, and model computation-layer processing latency of 50,000 events/s [3]. These queuing systems can hold the trade messages longer and re-transmit them when connectivity is restored, even after the same trade is transmitted to the downstream systems by other communication routes.

Another major cause of duplicates of trade messages is through manual resubmission workflows, especially when undertaking operations exception handling processes. The financial market surveillance systems with GAN-based architectures achieve a detection accuracy of 94.76 when identifying abnormal patterns, which is 15.5 percent higher than traditional systems, but low rates of false positives of 0.3 percent are still hard to achieve [3]. The locality-based attack algorithm shows that up to 23.2 percent of backup data can be inferred through frequency analysis with the assistance of recent previous backups, highlighting the challenges in recognizing frequency patterns that differentiate between legitimate and actual retransmissions [2].

3. Idempotency Framework and Message Fingerprinting Techniques

Idempotent message processing is the design principle on which the trading systems rely to allow duplicate messages to be received without

corrupting the downstream state. The point of an idempotent operation is that its effect is the same on a system when it is performed once or more with the same input parameters, and the message replays are apparently neutralized. To implement idempotency, unique message identifiers that are determined to be constant during n retransmissions must be defined, database operations must be created to identify and avoid duplicate processing, and adequate historical metadata is necessary to identify replays sent by a reentrant within a long time span [1].

The key composition of the business is the major method of creating uniqueness of the message in the trade in the lifecycle events. A coming business key is normally a combination of a number of attributes, such as the trade identifier, the execution venue identifier, the order identifier, the instrument identifier, and the execution timestamp at microsecond accuracy. The choice of the essential elements should strike a balance between the need to be unique and the probability of false negativity, where the legitimate trade amendments are wrongly identified as duplicates. Financial data architecture studies have also shown that SQL systems have extremely low memory utilization of 0.1406 MB and query times of 3.8927 seconds for 1 million records and can therefore be highly utilized in the quick key search of the business key [4]. Nevertheless, distributed processing models such as Apache Spark demonstrate better scalability with semi-structured data, achieving a query response time of 5.5915 seconds for equal quantities of records, which implies architectural compromises in key composition approaches [4].

Another common technique that complements business key methods is cryptographic fingerprinting, which produces deterministic hash values that are calculated over the normalized message payload as a whole. SHA-256 and SHA-3 algorithms create consistent-length fingerprints from trade messages that can vary in size, allowing for quick comparisons of messages, no matter how complex they are. The normalization procedure has to deal with the differences in the field ordering, inconsistencies in whitespaces, and the presence of optional fields to ensure that the same economic trade produces the same fingerprints at all times. The logic of implementation in fixed-income markets indicates that the market participants retain embedded relationships with up to 40 favored customers, with the salespeople making 35 calls every day to a specific customer to provide proprietary information [5]. Such an interaction pattern at high frequency requires fingerprinting systems that can accommodate thousands of messages per second whilst being deterministic in the generation of hash values in the presence of variations in message format.

By restricting replay detection to time windows that can be configured, temporal window-based deduplication logic trades off false positive reduction with memory usage and lookup speed. Fixed-window applications archive the entire message history over defined timeframes, and Python-based systems have been shown to consume 488.6354 MB of memory to process one million structured records and a total time of 164.5462 seconds to do the entire task [4]. Sliding window methods dynamically drop old entries, which ensure more predictable memory utilization patterns, which are found in Spark implementations, which require 163.0830 seconds total time and 488.6288 MB memory consumption with the same workloads [4]. The windowing time should be able to accommodate the maximum delays observed between the processing of original messages and possible replay arrival, with safety margins being provided to cover operational conditions such as longer periods of system maintenance.

Exactly-once delivery semantics Message queue architectures with deduplication infrastructure Message queue architectures with deduplication infrastructure at the infrastructure level offer deduplication capabilities that are complementary to application-layer deduplication logic. The fixed-income market structure demonstrates that over 90 percent of the trading volume is dependent on telephone-based embedded relationships between institutional investors and market makers, and the parties exchange private information via strong ties to take advantage of information asymmetry [5]. Nevertheless, in smaller transactions and liquid instruments, alternative trading systems (ATSs) exhibit arm's-length relationship patterns in which infrastructure-level guarantees are more relevant. Fingerprinting on the application layer is still required to provide end-to-end replay protection across heterogeneous system boundaries, as there is a large difference between the CPU utilization of SQL systems, with 3.8872%, and near-zero CPU utilization in Python and Spark implementations that make use of GPU acceleration, with 2.0 and 5.0 percent utilization of each type of data, respectively [4].

4. Blockchain-Enhanced Trade Relationship Modeling

Trade lifecycle management based on graph architecture has also been developed with a strong level of integration of blockchain technology, offering distributed systems to model complex

trading relationships and guarantee the integrity and non-repudiation of data. Contemporary applications use the multi-channel design of Hyperledger Fabric to establish single-channel supervision streams of various trading relationships and throughput with a rate of about 211-328 transactions per second (tps) and a transaction success rate of more than 99% [6]. In contrast to the old-fashioned centralized databases where there are high risks of information manipulation and the absence of clear audit systems, the models of trade relations based on blockchain provide a clear idea of trading organizations and the manner of their interactions as an unchangeable record of operations with cryptographic verification.

The network graph of the blockchain trading systems consists of various levels of hierarchy and participants of the trading system. Enterprise nodes carry trader identification credentials, organizational affiliations, and digital signatures approved by Certificate Authority (CA) systems. Trading channel nodes are independent communication channels between particular country commodities or bilateral trade agreements; each channel has its own ledgers that execute between 150-212 tps based on configuration parameters like block size (optimal 128 KB) and logging levels [6]. Transaction nodes hold information about the trades, including how much was exchanged, exact timestamps down to the millisecond, codes for tracking the data, and addresses that can be verified to make sure the information in the distributed ledger is clear and trustworthy.

Smart contract relations encode business logic to validate trade and regulatory compliance checks. Combining machine learning into these frameworks has shown accuracy of prediction of agricultural commodities of 69-88%, far outpacing traditional USDA forecast accuracy of less than 35% [7]. The hybrid storage architecture is an implementation that integrates on-chain cryptographic hashes (only a small block header block of about 80 bytes is needed) with off-chain rich transaction records to minimize blockchain storage requirements while preserving verification capabilities. This implementation has a traceability query response time (average 2.58 seconds) at 500 requests and an average transaction latency of 4.03 seconds at 500 requests/s [6].

High-confidence relationship (>90) association rules in blockchain trade networks have determined that when predictor-consequence relationships are considered in 96 commodity classifications, 67% of trade transactions are predictable when an antecedent-consequence relationship exists between the flow of commodities [7]. The validation of the distributed consensus mechanism is performed by several peer nodes before block commitment, and the system can handle network loads of up to 600 requests per second, after which performance degrades. The analysis of feature importance in these models shows that distance, GDP indicators, and population statistics are the major predictors of trade, accounting for 6.38, 6.22, and 3.83 percent of the variance in predicting black swan disruption phenomena, respectively. These factors can be used to formulate evidence-based policy in both routine business operations and during black swan disruption incidents.

5. Multi-Channel Trade Correlation and Lifecycle Event Disambiguation

A heterogeneous channel of trade correlation is a critical technical problem in the systems of duplicate detection that support contemporary fragmented trading activities. Trade confirmations are sent to financial institutions using many concurring methods, such as real-time connections using the FIX protocol, batch file transmission, RESTful API polling, and messaging queue subscriptions. The complexity of the European equity markets are reflected in the number of stocks that trade in primary markets as well as three alternative systems (BATS, Chi-X, and Turquoise) that execute around 10.5 million trades and 456 million messages each month [9]. Temporal correlation makes the tolerance windows have values of 50 to 500 milliseconds based on observed clock correlation accuracy [8].

Correlation algorithms have to derive correlation relationships even when there are variations in identity and formatting between channels. Ghost liquidity indicators show that after trading 100 shares in one venue, traders cancel about 19 shares in rival venues in 10-millisecond periods, and cancellation rates go up to 42 percent of the trade size in stocks of the UK when scaled by trade size [9]. The European venues market members (388) implement coordinated order management policies, conditionally cancelling duplicated orders across other executions, with ghost liquidity averaging 4.04% of the consolidated depth of alternative and primary venues [9].

Advanced temporal sequencing examination is required to distinguish between legitimate trade amendments and replay events. Production applications of machine learning classification models that are trained on historic patterns attain a disambiguation accuracy of more than 99.7% [8], using feature sets such as counts of attribute differences and timestamp delta distributions. Persistent-set selective search algorithms reduce the

validation state search by 27%, nearly an order of magnitude, across various application cases [8].

There are also partial fill situations that are further complicated in that several valid executions are required to differentiate between two reports. The graph traversal queries permit cumulative quantity analysis, which identifies inconsistency. Cross-venue correlation considers those situations when aggressive order routing policies implement parts of single logical orders in multiple venues at the same time. The proportion of global members trading in more than one venue is 72.81 percent of total traded volumes and accounts for 96.02 percent of volumes in alternative venues [9]. This indicates complex multi-venue execution policies that require highly developed correlation frameworks for accurate duplicate detection and lifecycle events.

6. Detection Logic Implementation and Tolerance Window Optimization

To actually implement duplicate detection logic, one should consider tolerance windows that are necessary to reduce the number of false positives and to minimize the likelihood of undetected replay. Timestamp tolerance windows establish time proximity within which arriving trades are detailed and compared to be possibly labeled as duplicates. Narrow windows are very specific and minimize false positive rates in situations where venues trade different trades in quick sequence but can miss delayed replays due to lengthy network failures or system recovery actions. The fault-tolerant container replication of HyCoR has a sub-millisecond added delay and less than one-second recovery latency, which proves that the tight tolerance windows may be compatible with high-speed recovery functionality [10].

Optimization techniques for tolerance windows use statistical results of the distribution of replay latency in production systems. Empirical research has shown that the replay latencies of network-induced duplicates are normally between milliseconds and seconds, whereas system recovery situations create delays of up to several hours. The performance overhead of the epoch-processing performance of HyCoR is within acceptable thresholds of real-world benchmarking [10]. Multi-tier window implementations use aggressive short windows to do first real-time detection, with periodic batch processes that run long historical windows to detect delayed replays that are past real-time limits.

Composite key matching strategies apply hierarchical comparison logic through successively-discriminative-characteristic identifier components of a most-discriminative attribute-to-least-discriminative-attribute hierarchy. One way to do this is to use deduplication systems with progressive sampled indexing: with a memory capacity of 64 GB and 11.6 TB at a 100 percent sampling rate, systems would deduplicate 97.9 percent at 1/86 sampling rates when scaled to 1,000 TB [11]. Container-based pre-fetching in 16 MB containers achieves a mark operation rate of 26 GB/sec and a backup throughput of 950 MB/sec of unique and 6 GB/sec of duplicate data, respectively [11].

Performance optimization methods would become important when trading volumes rise to millions of messages a day. Memory caching allows direct lookups in milliseconds at high throughput times, and deduplication systems have been demonstrated to support 123 billion objects with 500 TB of system memory out of only 25 GB [11]. Bloom filter probabilistic data structures offer space-efficient first-stage filtering, rapidly identifying messages that are unquestionably not duplicates prior to the application of costly exact matching logic to remaining candidates.

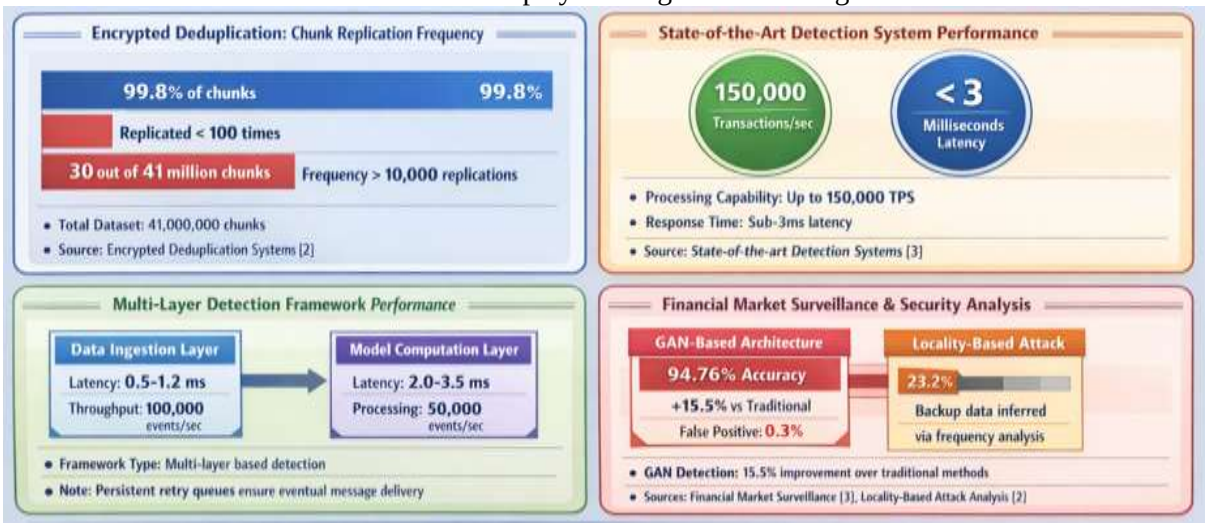

Figure 1: *Performance Metrics and Analysis Framework for Trade Deduplication and Financial Market Surveillance Systems [1]*

Figure 2: *Multi-Venue Execution Dominance: Trading Volume Distribution Across Market Participant Types [8, 9]*

Table 1: *Idempotency Framework and Message Fingerprinting Techniques [2, 3]*

Component	Description	Key Characteristics
Idempotent Message Processing	Duplicate message handling principle	Prevents state corruption; neutralizes replays; consistent system effects
Business Key Composition	Primary uniqueness identifier	Combines trade ID, venue, order, instrument, and microsecond timestamp; balances uniqueness with false negative prevention
Cryptographic Fingerprinting	Hash-based message identification	SHA-256/SHA-3 algorithms; deterministic output; handles message variations through normalization
Temporal Window Deduplication	Time-bounded replay detection	Fixed-window (complete history) vs. sliding-window (dynamic pruning); balances memory usage and lookup speed
SQL vs. Distributed Processing	Data architecture comparison	SQL: low memory, fast queries; Spark: better scalability with semi-structured data
Message Queue Architecture	Infrastructure-level deduplication	Exactly-once delivery semantics; complements application-layer fingerprinting
Market Structure Patterns	Trading relationship models	Fixed-income: embedded relationships with high-frequency calls; ATS: arm's-length transactions for liquid instruments
Processing Resource Utilization	System performance metrics	Variable CPU consumption across SQL, Python, and Spark; GPU acceleration impact

Table 2: *Blockchain-Enhanced Trade Relationship Modeling Framework [6, 7]*

Component	Description	Key Characteristics
Blockchain Platform Architecture	Distributed ledger system for trade lifecycle management	Multi-channel design; immutable transaction records; cryptographic verification, and high transaction success rate
Enterprise Nodes	Organizational participant identifiers	Trader credentials; organizational affiliations, and CA-approved digital signatures
Trading Channel Nodes	Dedicated communication pathways	Country-specific or bilateral agreement channels; independent ledgers; configurable block parameters; variable logging levels
Transaction Nodes	Individual trade record elements	Exchange quantities; millisecond-precision timestamps; tracking codes, and verifiable addresses for distributed ledger transparency
Smart Contract Relations	Automated business logic enforcement	Trade validation; regulatory compliance verification; machine learning integration for commodity prediction
Hybrid Storage	Dual-layer data management	On-chain cryptographic hashes (compact header

Architecture	system	blocks); off-chain detailed transaction records; optimized storage with verification preservation
Performance Metrics	System efficiency indicators	Traceability query response times; transaction latency measurements; request handling capacity, and network load thresholds
Association Rule Mining	Predictive relationship analysis	High-confidence commodity flow patterns; antecedent-consequence trade relationships, and commodity classification systems
Distributed Consensus Validation	Multi-peer verification mechanism	Pre-commitment peer node validation; scalable request processing, and defined performance degradation thresholds
Feature Importance Analysis	Predictive variable ranking	Distance factors; economic indicators; demographic statistics, and black swan event disruption modeling

7. Conclusions

Duplicate trade detection is a basic control system that defends current electronic trading infrastructure against systematic risks linked to events of message replay. This mix of idempotency solutions, cryptographic fingerprinting, and relationship modeling using blockchain creates strong defenses that can identify duplicate messages across different types of communication channels. The temporal window optimization approaches create a trade-off between detection precision and computational efficiency by using multi-tier methods that combine real-time narrow-window filtering with long-run batch scanning. The classification models in machine learning are very good at telling the difference between real trade changes and repeated actions, doing so accurately while keeping false alarms to a minimum. Distributed ledger architecture and graph-based architecture offer scalability in terms of representation of complex trade lifecycle relationships and the ability to generate highly complex correlation algorithms to support multi-venue execution strategy and partially filled scenarios. In-memory caching, probabilistic filter order structures, and progressive indexing methods are used to optimize performance and achieve microsecond latency when trading large volumes of data in high-frequency trading environments. These combined technology structures provide financial institutions with resilience against position inflation, corruption of profits, and loss reporting transgressions and offer solid bases on which they can effectively transact business in more and more intricate and interlinked international businesses.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.
- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

References

[1] Ayanniyi Mariam and Tanvir Khalid, "The Role of Idempotency Keys in Preventing Double-Settlement Errors: Why unique transaction identifiers are the backbone of accurate financial data," ResearchGate, 2025. https://www.researchgate.net/profile/Ayanniyi-Mariam/publication/399952183

[2] Jingwei Li et al., "Information Leakage in Encrypted Deduplication via Frequency Analysis: Attacks and Defenses," ACM Digital Library, 2019. https://dl.acm.org/doi/pdf/10.1145/3365840

[3] Yuexing Chen et al., "Real-Time Detection of Anomalous Trading Patterns in Financial Markets Using Generative Adversarial Networks," Preprints, 2025. https://www.preprints.org/frontend/manuscript/8f48a127381adf149a5751b753b89792/download_pub

[4] Sergiu-Alexandru Ionescu et al., "Engineering Sustainable Data Architectures for Modern Financial Institutions," Electronics, 2025. https://www.mdpi.com/2079-9292/14/8/1650

[5] Ali Reza Montazemi, John J. Siam, and Akbar Esfahanipour, "Effect Of Network Relations On The Adoption Of Electronic Trading Systems,"

McMaster eBusiness Research Centre (MeRC), 2007. https://prod-ms-be.lib.mcmaster.ca/server/api/core/bitstreams/16f5675b-7bba-4812-a5df-0def64424f3b/content

[6] Jianxiong Zhang et al., "Distributed Supervision Model for Enterprise Data Asset Trading Based on Blockchain Multi-Channel in Industry Alliance," Sensors, 2022. https://www.mdpi.com/1424-8220/22/20/7842

[7] Feras A. Batarseh et al., "Public policymaking for international agricultural trade using association rules and ensemble machine learning," Machine Learning with Applications, Volume 5, 2021. https://www.sciencedirect.com/science/article/pii/S2666827021000232

[8] Patrice Godefroid et al., "Using Partial-Order Methods in the Formal Validation of Industrial Concurrent Programs," ACM, 1996. https://dl.acm.org/doi/pdf/10.1145/226295.226324

[9] Hans Degryse et al., "Cross-Venue Liquidity Provision: High Frequency Trading and Ghost Liquidity," HAL Science, 2021. https://hal.science/hal-03338259v1/file/Degryse%2CDeWinne%2CGresse%2CPayne2019.pdf

[10] Diyu Zhou and Yuval Tamir, "HyCoR: Fault-Tolerant Replicated Containers Based on Checkpoint and Replay," arXiv:2101.09584v1, 2021. https://arxiv.org/pdf/2101.09584

[11] Fanglu Guo and Petros Efstathopoulos, "Building a High-performance Deduplication System." https://www.usenix.org/legacy/events/atc11/tech/final_files/GuoEfstathopoulos.pdf