Leveraging Networking Techniques to Improve Data Integration for Business Intelligence Systems

Malvina NIKLEKAJ[1]
https://orcid.org/0009-0005-0506-215X

Jora BANDA[2]
https://orcid.org/0009-0004-8497-2903

Abstract

Integrating heterogeneous data sources for Business Intelligence (BI) systems remains a critical challenge in the digital transformation of organizations. Most of the literature focuses on traditional integration techniques such as ETL/ELT or on the use of advanced architecture such as data lakes and data warehouses, often neglecting the impact that network protocols and transmission policies have on the process. This study proposes a new "networking-aware" approach to data integration in BI, analyzing the role that protocols such as HTTP/2, HTTP/3/QUIC, gRPC, MQTT or WebSockets play in ensuring data freshness, reducing latency and managing network costs.

The methodology followed includes analytical performance modeling through queuing theory (queuing models), formalization of metrics for measuring latency and data staleness, and analysis of typical integration scenarios (CRM–ERP, IoT streaming, and hybrid integration). The predicted results highlight the advantages

[1] Department of Informatics and Technology, Faculty of Engineering, Informatics and Architecture, European University of Tirana, Tiranë, Albania

[2] Department of Informatics and Technology, Faculty of Engineering, Informatics and Architecture, European University of Tirana, Tiranë, Albania

and limitations of different protocols, suggesting optimal combinations for different use cases.

The main contribution of this paper is the creation of a conceptual framework that links network metrics to BI integration, providing practical recommendations for architects and researchers. The conclusions show that choices at the network layer are not only technical issues of data transmission, but essential elements that directly affect the quality, consistency, and cost of business analysis.

Key words: *Data Integration, BI, Network Protocols, CDC, ETL Process.*

Introduction

Background of the problem

In the digital age, organizations increasingly rely on Business Intelligence (BI) systems to make informed decisions and generate competitive advantage. BI is no longer a luxury, but a necessity for managing strategic information, monitoring performance indicators, and responding quickly to market changes. This process depends on the ability to integrate data distributed across different sources such as enterprise resource planning (ERP) systems, customer relationship management (CRM), cloud applications, or IoT sensor networks.

However, the challenge lies not only in the heterogeneity of the sources or the transformation of the data, but also in the network infrastructure that transports this data. A delayed or costly integration directly affects the "freshness" of the information, reducing the validity of BI analyses. This situation makes it necessary to explore a new perspective: how network protocol choices and transmission policies affect the quality of BI integration.

Motivation of the study

In practice, many BI architectures consider the network simply as a neutral "pipe" for transporting data. The network is not neutral at all. The choice between HTTP/2 and HTTP/3, the use of gRPC over REST, or the adoption of lightweight protocols such as MQTT in IoT, has a major impact on latency, throughput, consistency, and cost. For example:

- HTTP/1.1 has problems with head-of-line blocking, while HTTP/2 and HTTP/3 offer multiplexing and latency reduction.
- gRPC, using HTTP/2, enables more efficient communication with binary messages.

- MQTT is used for environments with limited bandwidth but has challenges with message persistence.
- QUIC as a next-generation protocol over UDP significantly reduces connection setup time, facilitating real-time BI.

These characteristics make the choice of protocol a critical element for the success of BI.

Purpose and objectives

This study aims to create an evaluation framework to analyze the impact of network protocols on BI data integration. The main objectives are:

1. To identify key metrics that relate network performance to BI freshness and latency.
2. To model typical integration scenarios (CRM–ERP, IoT streaming, hybrid) through queuing theory and analytical models.
3. To compare the main protocols (HTTP/2, HTTP/3/QUIC, gRPC, MQTT, WebSockets) in terms of performance and costs.
4. To provide practical guidelines (design guidelines) for BI architects and researchers in the field.

Research questions

RQ1: What combinations of protocols and transmission policies minimize staleness for a given network budget?
RQ2: How does the use of HTTP/3/QUIC compared to HTTP/2 affect the freshness and throughput of CDC pipelines?
RQ3: Is the use of delta-sync with compression more efficient than micro-batching when sources are heterogeneous and schemas change frequently?

Contribution of the paper

The main contribution of this paper is the integration of the network perspective into BI processes, building a bridge between the data systems literature and that of network communications. Instead of viewing the network as an immutable channel, this paper proposes that the BI architecture be designed as a "network-aware" system, where the choices in transmission protocols and policies are crucial for the success of real-time and affordable BI.

Related Work

Data integration for Business Intelligence (BI) systems is a widely explored area in the scientific literature, but it has often been treated mainly from the perspective of ETL (Extract–Transform–Load) and ELT (Extract–Load–Transform) processes. Very few works have emphasized the impact of network protocols and their parameters on the quality and freshness of integration. This section analyzes the existing literature on three pillars: (1) data integration techniques and Change Data Capture (CDC), (2) the concept of data freshness and consistency, and (3) network protocols and their role in distributed systems. Through this review, we highlight the current gaps and justify the need for a "networking-aware" approach to BI.

Data Integration and CDC

In the literature, ETL/ELT have been described as the traditional mechanisms for building data warehouses. However, their main limitation is high latency, as data is often updated at periodic intervals (e.g., once a day) and does not serve real-time BI needs (Stonebraker, 2018). To overcome this, the Change Data Capture (CDC) approach was developed, which allows the transmission of changes as soon as they occur in the data source.

A recent study (Kumar & Yadav, 2025) provides a detailed analysis of CDC methods – log-based, trigger-based, and timestamp-based – comparing them in terms of latency, robustness, and scalability. The paper shows that log-based CDC is more efficient for systems with large data volumes, while trigger-based CDC is more suitable for small but frequently changing schemas. However, none of these approaches consider network constraints when transmitting changes.

Similarly, works such as Abadi's (2020) on data management in the cloud highlight the limitations of migrating data to cloud-native environments, where the cost of transferring data over the network (egress cost) becomes a significant factor. This suggests that data integration cannot be seen in isolation from a network perspective.

Data Integration and CDC

In the literature, ETL/ELT have been described as the traditional mechanisms for building data warehouses. However, their main limitation is high latency, as data is often updated at periodic intervals (e.g., once a day) and does not serve the needs of real-time BI (Stonebraker, 2018). To overcome this, the Change Data Capture

(CDC) approach was developed, which allows the transmission of changes as soon as they occur in the data source.

A recent study (Kumar & Yadav, 2025) provides a detailed analysis of CDC methods – log-based, trigger-based and timestamp-based – comparing them in terms of latency, robustness and scalability. The paper shows that log-based CDC is more efficient for systems with large data volumes, while trigger-based CDC is more suitable for small but frequently changing schemas. However, none of these approaches consider network constraints when transmitting changes.

Similarly, works such as Abadi's (2020) paper on data management in the cloud highlight the limitations of migrating data to cloud-native environments, where the cost of transferring data over the network (egress cost) becomes a significant factor. This suggests that data integration cannot be viewed in isolation from a network perspective.

Data Freshness and Consistency

An important dimension of integration for BI is data freshness. According to Materialize (2023), freshness is measured as the time interval between the creation of a record and the moment it is available for analysis. In operational BI applications, such as sales monitoring or IoT analytics, a delay of even seconds can reduce the value of data (Zhao et al., 2022).

Studies on freshness (Elementary Data, 2024) suggest metrics such as data staleness and freshness lag, which are closely related to network latency and streaming policies. For example, a pipeline that uses micro-batching every 5 minutes cannot provide freshness better than this interval, regardless of processing power. This shows that the network and streaming rate are natural constraints for BI.

From a consistency perspective, Carbone et al. (2017) discuss exactly-once, at-least-once, and at-most-once approaches in streaming systems such as Apache Flink. These approaches are influenced by network policies and how lost packets or retransmissions are handled. Thus, even consistency guarantees cannot be achieved without considering network behavior.

Network Protocols and Modern Architectures

HTTP/2 and HTTP/3 (QUIC)

Several papers have compared HTTP/2 and HTTP/3 over QUIC. For example, Chellappa et al. (2022) demonstrated that HTTP/3 performs significantly better than HTTP/2 in networks with packet loss, thanks to the multiplexing and congestion control mechanisms built into QUIC. Similarly, Peng et al. (2023) compared QUIC with TCP in cloud environments, showing that QUIC significantly reduces connection setup time and improves throughput under various conditions.

However, these studies have been conducted mainly in the context of streaming media or web applications and have not addressed their implications for data integration for BI. This is a gap that our paper aims to fill.

gRPC and WebSockets: gRPC, which uses HTTP/2 and binary serialization with Protobuf, has been praised for its efficiency in applications requiring low latency (Google, 2021). On the other hand, WebSockets have been used for real-time bidirectional communication, often in BI applications requiring live visualization (e.g., dashboards). Existing literature (Liu et al., 2021) acknowledges that these protocols are more efficient than REST, but a systematic analysis of their impact on freshness and integration costs is lacking.

MQTT for IoT: In IoT contexts, MQTT is used due to its ease of use and minimal bandwidth usage. A study by Saif & Matrawy (2021) suggests that an alternative with HTTP/3 can offer better performance than MQTT-over-QUIC in networks with limited resources. This shows the potential of modern protocols for applications where data integration has high freshness requirements.

Merging Perspectives

Although the literature on BI, data freshness, and network protocols is rich, very few papers bring these perspectives together. Each field is treated almost in isolation:

- BI and CDC focus on semantic and architectural aspects but often assume an "ideal" network.
- Papers on QUIC, HTTP/3, and MQTT deal with network performance, but without analyzing the impact on BI pipelines.
- Papers on data freshness talk about metrics but are not closely related to protocols or transmission policies.

This gap is precisely where this paper contributes by proposing a unified framework that links protocol choices and network parameters to BI freshness, latency, and cost.

Research Methodology

Methodological Approach

The methodological approach of this study combines the development of a conceptual model and "network-aware" architecture with analytical performance analysis. Specifically, a modeling and evaluation strategy like Design Science

research is followed, where the creation of artifacts (conceptual model and proposed architecture) is accompanied by their rigorous evaluation. Initially, based on the existing literature and research objectives, the main factors that link network performance to the quality of data integration in BI are identified. Subsequently, these factors are formalized through analytical modeling, where metrics such as network latency, throughput, data staleness, etc., are defined in mathematical terms. For this purpose, queuing theory is used, which allows modeling of integration scenarios as systems where data is "queued" for transmission or processing. This analytical approach enables the prediction of latency and data freshness under different network conditions without the initial need for physical implementation.

To ensure practical relevance, the study focuses on three typical integration scenarios identified in literature:

(i) Enterprise systems synchronization scenario (e.g. CRM–ERP), where transactional data is migrated from an operational system to a central repository.
(ii) IoT streaming scenario, where sensors continuously generate data that needs to be integrated in real time; and
(iii) Hybrid cloud on premises scenario, where data sources are distributed between on-premises and cloud environments.

These scenarios represent a wide range of use cases and allow us to test our approach under different conditions. For each scenario, an abstract model is built that includes the data source, the transmission network and the receiver (BI system), as well as the main parameters (e.g., update frequency, message size, network capacity, etc.). By substituting the values of these parameters into the queuing theory models, performance metrics are calculated for different protocols and configurations. This allows for analytical comparison of alternatives without performing experimental implementation.

Selection of literature and scientific articles

Our methodology started with a comprehensive literature search to establish the theoretical foundations of the study and identify existing gaps. Initially, several key keywords related to the topic were defined, such as "real-time data integration", "ETL latency", "BI data freshness", "HTTP/2, HTTP/3, MQTT, gRPC network protocols", "Change Data Capture (CDC)", "BI streaming analytics", etc. The search was conducted in academic databases such as IEEE Xplore, ACM Digital Library, SpringerLink and Google Scholar, focusing mainly on publications from recent years (2017–2024) to reflect the latest developments (e.g. HTTP/3/QUIC protocols, cloud-native applications, modern streaming technologies). In total,

dozens of papers were identified, which were then subjected to filtering with certain inclusion/exclusion criteria.

The inclusion criteria for the literature were: (1) Papers that deal with data integration for BI or data warehouses and address performance issues (e.g. latency, data delay, throughput, scaling); (2) Studies on real-time integration techniques such as CDC, stream processing, incremental ETL, etc., especially those that report data freshness metrics or update times; (3) Articles from the field of computer networks that analyze communication protocols (e.g. HTTP/2 vs HTTP/3, MQTT, WebSocket) and their performance (latency, loss, etc.) in distributed systems; (4) Literature on data quality in BI (e.g. consistency, pipeline reliability) where the impact of infrastructure can be alluded to; (5) Sectoral sources (whitepapers, technical blogs of companies e.g. Materialize, Elementary Data) for modern definitions of concepts such as "data freshness" or best practices in pipeline monitoring. The exclusion criteria excluded studies that narrowly focused only on isolated aspects (e.g. only transformation algorithms without discussing processing times at all, or strictly theoretical network works unrelated to data applications), as well as old classical literature (before 2010) in cases where newer data/studies existed on the same topic.

After filtering, the selected literature was grouped by topic to structure the review. First, works on data integration techniques were examined from traditional ETL to modern approaches such as CDC. Here it was clearly seen that periodic ETL suffers from high latency and does not fit the contemporary needs for real-time BI. To overcome this limitation, the literature suggests mechanisms such as CDC with real-time change delivery, which have been extensively analyzed in contemporary works. A 2025 study (Kumar & Yadav) compared CDC variants in terms of latency, updating consistency, and scalability, finding relative advantages of each approach. These results helped us identify key metrics for our analysis framework. The literature in this area also highlighted that, regardless of the technology used for integration, the network infrastructure can become the "weakest link": For example, Abadi (2020) notes that in cloud environments, the "egress cost" (the cost of transmitting data outside the cloud) can become a dominant factor in the total cost – a clear indication that data integration and the network should be addressed together.

Second, the literature on data freshness and consistency was covered. These studies (e.g. Materialize, 2023; Elementary Data, 2024; Zhao et al., 2022) provide definitions and metrics for measuring freshness, such as data staleness or freshness lag. These concepts are closely related to network latency and transmission policies: e.g., as argued in the literature, a pipeline with micro-batch every 5 minutes can never achieve better than 5-minute freshness no matter how powerful the processing is simply because it is limited by the delivery interval. Such a finding from the literature confirmed our hypothesis that the network and the transfer

rate are natural constraints for BI and therefore should be analyzed carefully. Also, regarding data consistency, works on streaming systems (such as Apache Flink) that handle delivery semantics (exactly-once, at-least-once, at-most-once) were reviewed. Carbone et al. (2017) show that achieving exactly-once depends not only on synchronization protocols, but also on network behavior – e.g. how lost packets are handled, their retransmission. These references guided us to include transmission reliability as an evaluation dimension in our framework.

Third, the literature on modern network protocols (HTTP/2, HTTP/3, gRPC, MQTT, WebSockets) and their implementations were analyzed. Comparative experimental studies were found, e.g. Chellappa et al. (2022) showed that HTTP/3 (over QUIC) outperforms HTTP/2 under packet-lossy network conditions, while Peng et al. (2023) reported that QUIC significantly increases throughput compared to TCP in cloud environments. Also, industry papers (e.g. Google, 2021) highlight the efficiency of gRPC (over HTTP/2) for low-latency applications, and other studies suggest the use of WebSockets for real-time two-way communication (e.g. for live dashboards) instead of traditional HTTP. In the IoT context, the literature notes the dominance of MQTT due to its simplicity and low overhead, however, alternative proposals also emerge: Saif & Matrawy (2021) experimented with a replacement of MQTT with HTTP/3 for resource-constrained IoT scenarios, achieving better performance than MQTT-over-QUIC. This literature data helped us understand which protocols are worth testing in our scenarios and what potential advantages each has.

At the end of the literature review, knowledge from different fields was consolidated and it was observed that most of the existing works treat BI integration, freshness metrics and network aspects separately. As highlighted by the literature itself, BI/CDC works mainly at the logical level and assumes an ideal network. Network studies focus on protocol performance without analyzing the impact on BI pipelines, and freshness studies rarely deal with protocol details. This multidisciplinary gap clearly identified by other authors justifies the contribution of this paper. This determined the direction of our methodology: to build a unifying framework and propose an architectural solution that integrates the network perspective with that of BI, supported by the theoretical and empirical evidence of the best existing literature.

Analysis Framework

To systematically assess the impact of networking techniques on data integration for BI, we have built an analytical assessment framework with five main dimensions. These dimensions are derived from the literature and best practices and represent key aspects of the performance and quality of an integration system. Below are these dimensions and the importance of each:

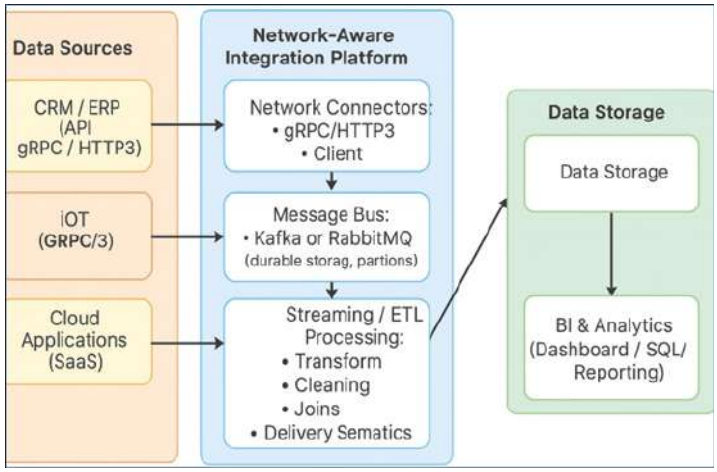
- D ata transmission latency: The time it takes for a unit of data to travel from the source to the BI system (repository or dashboard). Latency is critical because even delays of seconds can reduce the operational value of the data – e.g., in sales monitoring or financial fraud detection, data received a few seconds late may be too late for action. The objective is for the proposed architecture to minimize end-to-end latency, addressing network transport delays (e.g., avoiding slow handshakes, head-of-line blocking, etc.).
- Data Freshness: The extent to which the integrated data is up to date with the latest state of the sources. Formally, freshness is measured as the time interval between the moment a record is created/modified in the source and the moment it becomes available for analysis. A high freshness (short interval) ensures that decisions made by BI reflect current reality. In our framework, freshness is closely related to latency: the lower the latency, the fresher the data item is when it arrives. We also use metrics such as staleness and freshness lag from the literature to evaluate scenarios with periodic deliveries can achieve.
- Throughput and processed volume: The speed at which the system can integrate data (e.g. the number of messages or records transmitted per second) and the total volume that can be processed within a given time. Throughput determines whether a solution can scale to large data sizes without being blocked. In our context, throughput is affected by network choices: e.g. the use of the QUIC protocol (HTTP/3) has been shown to increase throughput in network conditions with latency or loss thanks to blocking avoidance and congestion control optimizations. In our evaluation, we will compare how different protocols affect throughput.
- Scalability: The ability of the architecture to maintain performance (or minimal degradation) as the system scale increases – whether in terms of the amount of data, number of resources, or frequency of updates. A scalable solution can handle a growth from thousands to millions of messages per hour without losing freshness or dramatically increasing latency. The literature suggests that different integration techniques have different scalability, e.g., log-based CDC often excels at high transaction volumes, while trigger-based CDC works better for small resources with frequent changes. In our framework, we assess scalability by analyzing how metrics respond when traffic intensity increases, and how proposed network protocols or architectures handle this load.
- Integration reliability and consistency: This dimension encompasses the assurance that transferred data arrives correctly, completely, and without duplication, and that the source and destination systems remain consistent during integration. This includes the concepts of delivery guarantees (exactly-once vs. at-least-once), fault tolerance, and proper loss handling. It is critical

that a BI system does not lose data during transfer or include duplicated data, otherwise decisions may be wrong. Reliability is directly affected by network behavior and protocols: for example, packet loss or connection interruptions can cause data inconsistencies or the need for retransmission, which affects consistency. In our framework we evaluate how different protocols address these issues (e.g. MQTT has QoS that ensure at least once or exactly once; HTTP/3 avoids reestablishing new connections for each message; gRPC provides end-to-end verification with acknowledgements, etc.). Furthermore, eventual coherence will be considered in cases where small delays are tolerated but final integrity must be guaranteed.

These five dimensions – latency, freshness, throughput, scalability, reliability – form the basis of our assessment. With such a framework in mind, in the following sections we will build the conceptual model and proposed architecture, and analyze them precisely along these dimensions, to answer whether and how networking techniques can improve data integration for BI.

Proposed Architecture for Improving Data Integration in BI

FIGURE 1. Proposed "network-aware" architecture for data integration in BI systems, illustrating multiple data sources, integration layer with different protocols, and repository for analysis.



The developed architecture, shown in Figure 1, is designed to leverage networking techniques to improve data integration performance. It follows an event-driven architecture paradigm, where data is captured and transmitted as it is generated, rather than being collected in periodic batches. This ensures a near-real-time flow of information from the sources to the BI repository. Furthermore, the architecture is "network-aware" – i.e. each communication block is designed

considering the network characteristics and choosing the optimal protocol for the respective scenario.

Data Sources: The left side of the architecture shows some typical types of data sources that a BI system might integrate: (1) Traditional on-premise systems (e.g., a CRM and an ERP) within the organization's infrastructure; (2) IoT devices/ sensors, which continuously generate data streams; and (3) Cloud Applications (SaaS), such as a cloud order management application, which provide APIs for accessing the data. Each source connects to the integration layer through a specific, tailored mechanism/protocol:

- For on-premises CRM/ERP systems, the architecture proposes using an advanced API with gRPC or HTTP/3 instead of traditional REST APIs. This is because gRPC (over HTTP/2) allows for more efficient communication (binary messages, multiplexing) that reduces latency and overhead compared to REST. A CDC service within these systems can be configured to capture changes (e.g., new changes to the ERP database) and push them immediately to the integration platform over this high-throughput connection. HTTP/3 (QUIC) can also be used, especially for high-frequency deliveries, as it avoids connection establishment latency and improves response time compared to HTTP/1.1/2. This is especially useful when transmitting a continuous stream of changes (e.g., hundreds of messages per second) – HTTP/3 can maintain a stable connection without blocking, minimizing the waiting time for each message.
- For IoT devices, the architecture foresees the use of a dedicated MQTT broker, which acts as an intermediary between the sensors and the integration platform. MQTT was chosen due to its ease and efficiency in low-bandwidth environments – it operates with a publish/subscribe model where sensors send messages to the broker, and the platform receives them from the broker. This ensures robust communication even in unstable networks, thanks to features such as MQTT's QoS (Quality of Service) that enables exactly once or at least once delivery. The broker also preserves messages in case of interruptions, increasing reliability. However, being "network-aware", our architecture is not limited to MQTT: we also consider the possibility for IoT sensors to communicate directly with HTTP/3 in specific cases. Recent research shows that an alternative HTTP/3 approach can compete with or outperform MQTT in some IoT scenarios – for example, when the number of devices is not too large and a QUIC connection can efficiently serve for secure, low-latency communication. Therefore, the proposed architecture allows for this flexibility: critical IoT devices can also send events directly via an HTTP/3 API (bypassing the broker), depending on the application requirements (e.g. for maximum freshness in ultra-low-latency industrial applications).

- For cloud applications (SaaS), which often offer streaming APIs or webhooks, the architecture leverages a persistent streaming connection to the integration platform. This can be achieved with WebSockets or similar mechanisms (e.g. Server-Sent Events) that allow the cloud application to "push" new data to us as it happens. By maintaining a persistent WebSocket connection, the overhead of creating new HTTP sessions for each message is avoided and bi-directional communication is ensured in real time. For example, if a sales cloud application generates events for each new order, these events are immediately sent to our integration platform via WebSocket, without waiting for periodic polling requests. This increases both freshness (events are received immediately) and efficiency (less overhead than HTTP).

Network-Aware Integration Platform: At the heart of the architecture is the integration platform, which is the layer where all data from the sources arrive, is transformed and prepared for storage. The platform is designed with modularity, having separate connectors for each of the protocols/sources. These connectors are software components that know how to communicate with the respective source (e.g. a gRPC/HTTP3 connector that calls the CRM/ERP API whenever there are changes or waits for stream messages; an MQTT Client connector that connects to the MQTT broker and listens to sensor topics; a WebSocket client connector for the cloud application, etc.). The connectors act as an interface between the network and the backend of the platform: they take the raw data from the network and place it in a shared messaging system within the platform.

Within the integration platform, at its heart, lies a message bus or in concrete terms can be implemented as a modern message broker. All events received by connectors are published to this central bus in a secure queue. The rationale for including a message broker is multiple:

First, it provides decoupling between data producers (sources) and consumers (BI repository), increasing scalability we can add new sources or change the production rate without directly impacting the consumer.

Second, the broker offers reliability guarantees: messages are stored persistently and can be re-delivered in case of failures, enabling at-least-once delivery. Technologies like Kafka also offer exactly-once guarantees with certain configurations, which is essential to avoid duplication in the integrated data.

Third, a message bus allows for parallelized and real-time processing: we can have several microservices subscribing to different data topics that transform/filter the flow in real time. These microservices leverage stream processing frameworks to perform ETL on the flight, minimizing the time data is spent in the system.

Warehousing and BI layer: After the data passes through the integration and transformation phase, it is sent to the storage layer – which is usually a central Data Warehouse, Data Lake or Data Lakehouse. In our architecture, this step is illustrated

by the "Data Warehouse/Lake" box in the figure, where the data is integrated and stored for analytical use. Depending on the nature of the data, a Lambda model (where data is processed in parallel as a real-time stream and in batches for ultimate accuracy) or Delta model (where streaming updates are immediately reflected in the warehouse tables with versioning techniques) can be applied. Defining the details of the warehouse is not the focus of this paper, however, the architecture is flexible enough to work with both transactional repositories that are constantly updated, as well as traditional repositories (e.g. a data warehouse where streaming data is entered into staging tables and consolidated periodically). It is important that the data reaches this layer with minimal latency and in a consistent manner from the previous steps.

The last layer is the BI presentation and analysis layer, where business intelligence accesses the data warehouse. Here our architecture does not intervene directly, but its effects are felt since the data in the warehouse is fresher, more consistent, and arrives at a higher speed, any BI query or report reflects information much closer to real-time reality. BI tools typically connect to the warehouse through SQL queries or analytical APIs and users get a near real-time view of the business.

Below, we summarize the step-by-step flow of this architecture, to highlight the key stages and the role of the network components:

1. Extracting events from sources: Source systems generate events that are immediately captured by dedicated connectors (e.g. CDC module with gRPC for ERP, MQTT client for IoT sensors, webhook for cloud application). Each connector ensures that communication with the source is done with the optimal protocol, minimizing latency and losses.
2. Network data transmission: Captured events are transmitted over the network to the hub. This is where networking techniques come into play: using low-latency protocols (HTTP/2, HTTP/3) and avoiding bottlenecks. The goal is for data to travel quickly and with as little overhead as possible over the network. For example, the ERP connector can aggregate several small changes into a larger message to better utilize HTTP/3 throughput, rather than sending 100 small messages individually.
3. Receiving and buffering at the integration platform: The integration platform receives incoming messages through its network gateways. Immediately, the messages are placed in the shared queue (broker), where it is ensured that no data is lost. This layer acts as an elastic buffer between production and consumption, allowing, for example, that even if a large flow of events arrives at a moment (burst traffic), the system can gradually absorb them without loss, thus increasing elasticity and scalability.
4. Data processing and transformation: Services within the platform process the incoming data. Here, transformations can be performed, and business

logic can be applied before storage. These services are read from the queue when there are messages ready and can be scaled horizontally. Thanks to the event-driven architecture, processing occurs in streaming as soon as an event arrives, its transformation starts without waiting for other events. This maintains freshness: data is not collected untouched but flows immediately towards the final step.

5. Loading (Load) into the BI repository: After processing, the data is incorporated into the BI repository. Depending on the technology, this can be done either by writing directly to the tables of a Data Warehouse or by storing files in a Data Lake (e.g. formats like Parquet in cloud storage) which are then refreshed at the query level. In the proposed architecture, both models are supported: for high-criticality events, we can have a "live updates" table in the data warehouse where every change is inserted as soon as it arrives; at the same time, a batch consolidation process can be executed in the background at certain intervals to integrate these changes into the final tables. This resembles the Lambda Architecture (stream + batch) to provide a balance between freshness and full historical accuracy. The key is that, thanks to the above optimized flow, this entire cycle from source to repository happens very quickly – the system aims for end-to-end latency in the order of seconds or sub-minutes, in contrast to the hours of a traditional integration.

With this architecture, we expect significant improvements in all dimensions of our analysis framework. The overall latency is reduced due to the elimination of batches and the use of faster protocols. The freshness is increased as data is refreshed as it changes (event-driven architecture) instead of waiting for fixed schedules. The maximum throughput of the system is increased by using parallel communication channels (e.g. Kafka connectors and partitions) and protocols that fully utilize bandwidth (e.g. HTTP/2 multiplexes several messages on a single connection). Scalability is ensured by the decoupled design: more broker instances or transformation processes can be added as the volume increases, without the need for resource changes. Reliability is strengthened by delivery safety mechanisms (MQTT QoS, automatic gRPC retries, storing events on disk by the broker) – thus, the system can tolerate partial network failures without losing data.

The proposed architecture is the result of a synthesis of theoretical research and practical requirements. It addresses the gaps identified in the literature by combining the best techniques from different fields: it simultaneously exploits modern network communication protocols and embraces modern data engineering principles (streaming, decoupling, observability). In the evaluation chapter (Results), we will analyze how this architecture performs precisely

according to the mentioned dimensions, comparing alternative configurations (e.g. what happens if HTTP/1.1 is used instead of HTTP/3, or batch vs streaming, etc.) to highlight the advantages and limitations. Before moving on to the results, however, it is important to also discuss the limitations of our methodological approach and possible alternatives which are presented in the following section.

Limitations of the Methodological Approach and Alternatives

While our methodological and architectural approach aims to be innovative, it is important to acknowledge its limitations and address potential alternatives. We discussed some of them below:

- Abstractions and assumptions of analytical modeling: Our theoretical model and queuing theory analysis rely on several simplifying assumptions (e.g., average distribution of event arrival times, treatment of the network as a stable average of latency, etc.). These assumptions help to achieve general results but may not fully capture the complexity of real-world situations. For example, in typical queuing analysis, a Poisson distribution of event arrivals or i.i.d. of service times is often assumed – in reality, the integration traffic may be bursty (e.g., many events after a quiet hour) or have heavy-tail distributions that do not fit standard models. This means that our theoretical predictions for latency or throughput may be somewhat optimistic or generalized.
- Lack of final experimental validation: Up to this point, our study has produced "predicted" results based on modeling and literature. We have not yet performed a full prototype implementation of the architecture to test in a production environment or a simulation with real traffic. This is an important limitation, as at the end of the day, real-world performance can be affected by unforeseen factors. For example, Chellappa et al. (2022) demonstrated the benefits of HTTP/3 under certain lossy network conditions – in our real system, scenarios with other types of bottlenecks could be encountered (e.g. networks with very low latency but limited bandwidth, or with packets being lost not because of the network but because the source applications fail, etc.). A final experiment (either with a laboratory testbed or with a network simulator) will be needed to validate and calibrate our results. This was beyond the scope of this chapter but is planned as future work.
- Limited focus on selected protocols: We have selected a specific subset of protocols (HTTP/2, HTTP/3, gRPC, MQTT, WebSocket) as the most promising based on the literature. There are other communication technologies that can influence data integration, which we did not examine in detail. For example, industry-specific protocols such as OPC-UA for data

integration from production machines or catching technologies and CDNs (Content Delivery Networks) for bringing data closer to the point of use. Also, configuration variants within the same protocol were not extensively explored. These constitute other optimization parameters that in practice may be as important as the choice of the protocol itself. Our limitation is that we have not covered the entire possible space of solutions; we have chosen to focus on the most prominent ones according to literature and the objectives of the study. An extended work could also consider these additional options.

- Interdisciplinary nature of the solution and complexity of implementation: The proposed architecture is quite complex, as it integrates components from different fields (network, data engineering, stream systems, etc.). The practical implementation of such a system requires extensive expertise and coordination between teams (e.g. network teams must work with database teams). This can be challenging in real organizational conditions a "social/ organizational" limitation for our solution. Some organizations may prefer simpler and more well-known solutions, even if with slightly lower performance, due to the high cost of complexity. Also, new technologies (such as QUIC/HTTP3) may still have adoption barriers. This study focuses on the theoretical and practical benefits but recognizes that the path to widespread adoption may be gradual.
- Trade-offs between different metrics: An important finding from both the literature and our analysis is that optimizing one metric can negatively impact another. For example, to achieve maximum freshness, the system should transmit data as often as possible, but this may reduce the effective throughput or increase the network load to the point of congestion. Theoretical studies on the Information Age confirm that there is not always a policy that simultaneously minimizes the average latency and the age of the information often a trade-off between freshness and throughput is needed. Our architecture is configured for balancing: e.g. it allows both continuous streaming and micro-batching, and the choice of the optimal interval may depend on priorities (freshness vs. efficiency). This flexibility is useful, but at the same time shows the limitation that there is no universal "silver bullet" maximum performance is achieved by balancing parameters according to the specific requirements of the scenario.
- Methodological alternatives: Instead of analytical modeling, an alternative would be a full experimental approach: e.g. building a small-scale prototype of the architecture (with some resources and an integration platform) and empirically measuring the metrics in a controlled environment. This would provide concrete results and capture real-world nuances. Another methodological alternative would be to develop a specialized simulator for

data pipelines, where protocol changes could be tested with thousands of synthetic loads such a tool could explore the parameter space wider than our manual analyses. We chose analytical modeling because it allowed us to explore concepts and variables very quickly and identify key relationships, thus reducing the search space for a future experiment. However, the next logical step of this work will be precisely the experimental validation of the findings: setting up a pilot environment where to implement the architecture with real components (e.g. a Kafka broker, a gRPC server, some simulated MQTT sensors) and directly measure latencies, throughput, freshness, etc., under different conditions. This will strengthen the conclusions and address the limitations.

In conclusion, we are aware that our approach, although promising on paper, has its limits. However, identifying these limitations allows us to be transparent about the validity of the results and to set directions for further work. The alternatives discussed (practical experimentation, advanced simulations, expanding the range of protocols) offer opportunities to further refine and verify the conclusions of this study. In the following chapters, we will present the results obtained from our analysis, considering these limitations when interpreting them, and we will suggest how these results can be implemented or further tested in the future.

Background on Current BI Data Integration Approaches

Data integration is the foundation of any business intelligence (BI) system. It ensures that information distributed across different operating systems, separate applications, and external sources is collected, transformed, and stored in a centralized manner for later analysis by users and decision makers. Traditionally, this integration has been accomplished through ETL (Extract, Transform, Load) and later ELT (Extract, Load, Transform) processes, which have been the industry standard for decades. However, the development of new technologies, the increase in data volume and velocity, and the demand for real-time analysis have exposed the limitations of these classic approaches and paved the way for new methods, such as streaming ETL and Change Data Capture (CDC).

Traditional Approaches: ETL and ELT

The ETL Process

The ETL (Extract, Transform, Load) model emerged in the 1980s to consolidate data distributed across different operational systems into a data warehouse. This process is divided into three main steps:

- Extract: retrieving data from multiple sources, such as relational databases, ERP, CRM applications, or external files.
- Transform: cleaning, normalizing, and harmonizing data into a unified format.
- Load: loading the processed data into a Data Warehouse for analysis.

This model became very popular, as it provided a structured and centralized way to build historical reports and support in-depth analysis.

The shift to ELT

With the emergence of modern Data Warehouses and Data Lakes that offered large processing capacities, the practice evolved towards ELT (Extract, Load, Transform). In this model, data is first loaded into the warehouse, and transformation is performed within it by leveraging the processing power of the database or cloud environment. ELT offered greater flexibility and reduced preparation time, but still remains limited when it comes to applications that require real-time analysis.

Limitations of ETL/ELT

Despite its popularity, ETL/ELT approaches have some notable weaknesses:

- High latency: updates are often performed once a day or at hourly intervals, resulting in delayed reports.
- Batch processing: processing occurs in large blocks, not in real time.
- High complexity: creating ETL pipelines requires a lot of custom code and maintenance.
- Infrastructure cost: requires powerful servers for large transformations.
- Lack of flexibility: changes in sources or data structure often break existing pipelines.

Modern approaches: Streaming ETL and Change Data Capture (CDC)

Streaming ETL

Streaming ETL differs fundamentally from the classic process: instead of waiting for a batch to be collected, data is processed as it arrives (event-driven). This is enabled by platforms such as Apache Kafka, Apache Flink or Spark Streaming, which treat events as continuous streams. For example, a new sales transaction is immediately inserted into the stream, transformed and made available for analysis without waiting for the nightly cycle.

Change Data Capture (CDC)

CDC is a technique that tracks changes in operational databases and continuously transmits them to an analytical repository. There are several implementation mechanisms:

- Log-based CDC (tracing transaction logs in the database).
- Trigger-based CDC (functions that trigger events when a change occurs).
- Query-based CDC (comparing previous versions with current ones).

CDC has gained popularity because it allows for near-real-time synchronization of BI systems with operational databases.

Benefits and challenges

These modern approaches bring several key benefits:

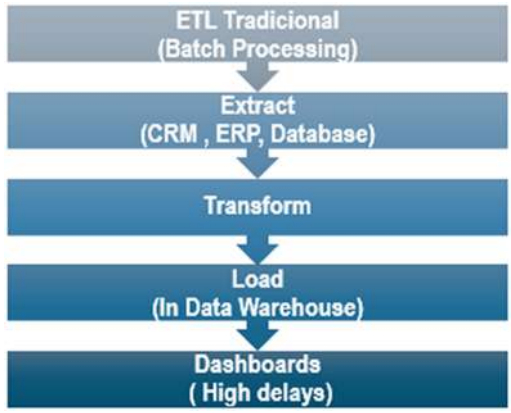
- Much lower latency (seconds or minutes, not hours/days).
- Possibility of real-time analysis and interactive dashboards.
- Better scalability thanks to event-driven architecture.

However, they also have challenges:

- High implementation complexity.
- Need for efficient protocols and networks to handle high volumes.
- Eventual consistency: in some cases, BI must accept that data is not always 100% synchronous.

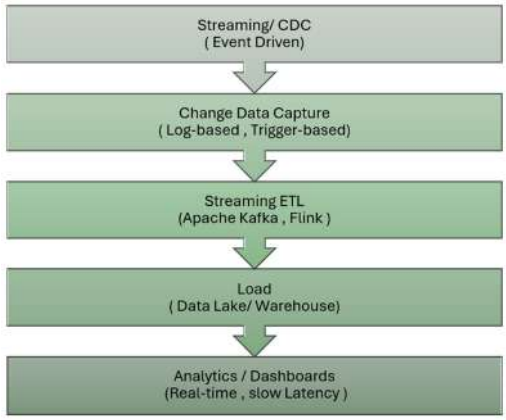
Typical Use Cases

FIGURE 2. Traditional ETL process for data integration in Business Intelligence.



This figure depicts a classic data processing model with an ETL (Extract, Transform, Load) approach, which uses batch processing as the main mechanism. The process starts with the extraction of data from various operational sources such as CRM, ERP or transactional databases. After collecting, the data undergoes the transformation phase, where it is cleaned, normalized and harmonized to be suitable for further analysis. In the next step, the data is loaded (Load) into a Data Warehouse, which serves as a central repository for storage and management. The final phase is the creation of analytical dashboards, which provide reports and visualizations based on the collected data. However, this process is characterized by high delays, since batch processing is performed at certain time intervals (e.g., daily or weekly schedules), and not in real time.

FIGURE 3. Real-time data integration process for Business Intelligence



This figure illustrates a scalable data integration pipeline that combines advanced processing and analysis techniques to improve the efficiency of Business Intelligence systems. The process begins with Streaming/Change Data Capture (CDC), where new data or changes to the source are captured in real time through log-based or trigger-based approaches. This data is then sent to a Streaming ETL layer, typically implemented with platforms such as Apache Kafka or Apache Flink, which enables data transformation and enrichment on the fly.

CRM and ERP

The integration of CRM (Customer Relationship Management) and ERP (Enterprise Resource Planning) systems is a classic case where ETL has been the main method. However, slow updates cause delays in reflecting the real state of sales or inventory. With CDC and advanced APIs over HTTP/2/HTTP/3, these delays can be significantly reduced.

IoT and sensors
IoT devices generate data continuously, often in real time. The batch method is unsuitable, while streaming ETL and MQTT become the natural choice for integration. For example, sensors in a factory that measure the temperature of machines need to transmit events in real time for immediate monitoring.

Cloud Applications
Many SaaS services offer webhooks or event streaming. Their integration requires an event-driven approach and efficient communication protocols (e.g. WebSockets).

From traditional to modern approaches
The review of literature and practices clearly shows that ETL/ELT, although still widely used, are not suitable for real-time BI. Instead, companies are increasingly adopting streaming ETL and CDC combined with network protocols that offer low latency. This transition forms the basis on which this paper proposes the use of network techniques to improve BI integration.

Networking Techniques for Enhanced BI Integration

The development of network technologies has created new opportunities for improving data integration in business intelligence (BI) systems. While traditional data processing approaches have relied primarily on ETL processes and central storage mechanisms, modern approaches are increasingly oriented towards the use of advanced communication protocols and network techniques that provide low latency, high throughput, and better security in the transmission of information. These techniques do not only replace the integration process but serve as a basic layer that determines the quality of real-time analytical experience.

One of the most important developments is the transition from traditional communication protocols, such as HTTP/1.1, to more efficient protocols such as HTTP/2 and HTTP/3 (QUIC). HTTP/2 introduced the concept of multiplexing, which allows for the parallel sending of many requests over the same TCP connection. This avoids problems known as "head-of-line blocking" and significantly increases the speed of data integration from multiple sources. For example, a BI system consuming data from several different CRM or ERP APIs can process it much faster using a single multiplexed channel. However, TCP remains a limitation due to its linear nature in handling errors and packet losses. This limitation is addressed by HTTP/3, which uses the QUIC protocol built on top of UDP. QUIC provides not only reduced latency due to more efficient

error recovery, but also built-in security with end-to-end encryption. For BI data integration, this means that reports and dashboards that depend on fast updates can benefit from an optimized and protected communication channel.

Another networking technique that is gaining traction in BI is the use of gRPC, a communication framework developed by Google that is based on HTTP/2 and uses Protobuf for data serialization. Due to its efficiency in reducing overhead and support for bidirectional streaming, gRPC is particularly well-suited for scenarios where BI integration requires frequent synchronization between different data sources. For example, in the case of a company that collects data from a cloud application, an on-premises database, and a third-party analytics service, using gRPC can provide a more consistent integration that is less affected by network latency.

In the field of IoT and data sources that generate continuous events, the use of MQTT (Message Queuing Telemetry Transport) has become the de facto standard. This lightweight protocol is designed for environments with limited bandwidth and low latency, making it ideal for integrating sensor data into BI systems. For example, a network of sensors in a manufacturing plant that transmits temperature, humidity, and energy consumption can use MQTT to ensure that this data arrives in real time to the analytics platform, where it can be translated into alerts or process optimizations. Due to its lightweight nature, MQTT also reduces the load on servers and increases scalability. Another technique that is being widely used is WebSockets, which provides a continuous, full-duplex connection between the client and the server. Unlike HTTP, which is based on request-response, WebSockets allows messages to be sent in both directions independently. For BI systems, this means that dashboards and visualizations can be automatically updated as soon as there are changes in the underlying data, without the need for constant refreshes. In a company where decision makers monitor sales performance in real time, architecture based on WebSockets ensures that any new transaction or change in inventory is immediately reflected in the analytical screens.

Another essential aspect of network techniques is the optimization of data flow through compression and micro-batching. Although BI often requires the freshest data possible, in some cases efficiency can be achieved by aggregating several small messages into a single packet, reducing communication overhead. Micro-batching is a compromise between traditional batching and pure streaming, balancing freshness with transmission efficiency. Likewise, data compression techniques, integrated into modern protocols, help reduce transfer time and network bandwidth consumption.

In addition to performance, transmission security remains a critical element. New protocols like QUIC and gRPC offer built-in security with forced encryption, eliminating the need for additional complex configurations. For organizations

that handle sensitive data, such as financial or healthcare data, this is an additional guarantee that the BI integration process does not compromise the privacy and integrity of information.

On a broader level, network techniques should be seen not simply as technological solutions, but as architectural elements that can transform the way BI platforms are built. The combination of modern protocols, event-driven communication, and transmission optimization creates a new paradigm, often called network-aware BI integration. This paradigm no longer sees the network as a neutral barrier or channel, but as an intelligent layer that directly impacts the quality of analysis and decision-making.

Comparative Analysis and Discussion

Comparison Table

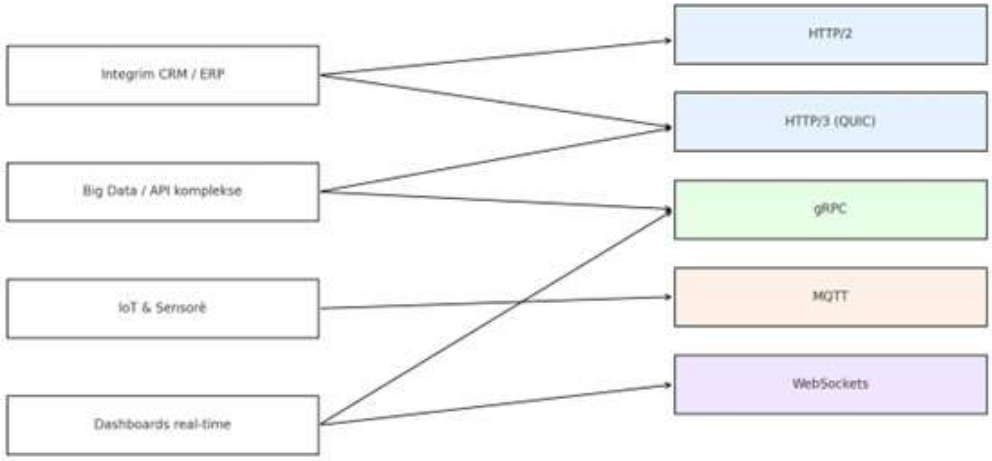
Table 1 presents a summary of the performance of network protocols against key dimensions of data integration.

TABLE 1. Comparison of network protocols for BI integration

Protocol	Latency	Throughput	Data freshness	Scalability	Credibility
HTTP/2	Average (improvement over HTTP/1.1 with multiplexing)	High in stable connections	Good freshness, but not always real-time	Good scalability for traditional APIs	High, thanks to TCP, but affected by head-of-line blocking
HTTP/3 (QUIC)	Very low, thanks to the elimination of head-of-line blocking	High even in environments with packet loss	Very good freshness, close to real-time	Greater scalability for cloud-native services	Very high, built-in encryption, fast recovery
gRPC	Very low, due to Protobuf and HTTP/2	Too high for binary transmission	Very high refresh rate, supports bidirectional streaming	Strong scalability for microservices and complex APIs	Very high, corporate standard with TLS security
MQTT	Very low, designed for IoT	Limited due to small messages	Very high freshness (event-driven)	Good scalability for millions of IoT devices	High, but depends on the broker's preferences
WebSockets	Very low, continuous connection	High for two-way communication	Very high refresh rate (real-time push)	Good scaling, but requires connection management	High, but depends on the servers that store the sessions

This table illustrates that each protocol has its own strengths and limitations, and the choice depends on the context of the BI integration. For example, HTTP/3 and gRPC are better suited for complex, high-volume integrations, while MQTT is indispensable in IoT environments. WebSockets, on the other hand, are important for continuous interactivity of dashboards.

FIGURE 4. Mapping of Application Use Cases to Communication Protocols



Comparison Analysis and Interpretation

Latency

One of the key factors in BI is latency. The faster data arrives from the source to the repository or dashboard, the more valuable it is for decision-making. HTTP/2 brought a significant improvement over HTTP/1.1 but still suffers from the limitations of TCP. HTTP/3 and QUIC, using UDP and eliminating head-of-line blocking, offer significant latency reduction. In comparison, gRPC is even more efficient than HTTP/3 in many cases, due to binary serialization and support for bidirectional streaming. MQTT and WebSockets offer the lowest possible latency, as they are specifically designed for real-time communication. From this analysis, BI that requires real-time dashboards should rely on technologies such as gRPC, MQTT or WebSockets.

Throughput

Throughput is the dimension related to the amount of data that can be transmitted in a given time. HTTP/2 and HTTP/3 offer high throughput for API scenarios. gRPC is particularly efficient for integrating complex services, where complex data structures are transmitted. MQTT, although with more limited throughput, is sufficient for small messages from millions of sensors. WebSockets offer

good throughput for two-way communication, but in large systems require optimizations to manage multiple connections. For BI that handles Big Data or integrates multiple enterprise sources, the combination of HTTP/3 and gRPC is the most suitable.

Data freshness
BI increasingly needs high data freshness, i.e. immediate updates. Protocols such as gRPC, MQTT and WebSockets ensure that events are transmitted without delay, enabling real-time analysis and alerts. HTTP/2 and HTTP/3 offer satisfactory freshness but are not always built for event driven. This suggests that for critical applications such as industrial or healthcare equipment monitoring, BI should use protocols with an event-driven nature.

Scalability
Scalability is essential for organizations operating in complex environments with hundreds or thousands of data sources. HTTP/3 and gRPC scale well in cloud-native environments, supporting microservices architectures. MQTT is designed specifically for horizontal scaling, enabling millions of sensor connections. WebSockets offer good scalability, but managing open connections remains a challenge for servers. This means that there is no single protocol for all cases: the choice depends on the nature of the data and resources.

Reliability
Reliability is a prerequisite for BI integration, especially when data is used for critical decisions. TCP-based protocols (HTTP/2, gRPC) provide a high level of security and consistency. HTTP/3 strengthens this aspect with built-in encryption and fast error recovery. MQTT, while reliable, depends on broker configuration and network quality. WebSockets are reliable for interactive communication but require additional mechanisms to preserve sessions in the event of an outage. This suggests that for organizations with strict security requirements, the combination of HTTP/3 and gRPC offers the best balance between performance and integrity.

Practical examples

In the case of CRM and ERP, the use of gRPC and HTTP/3 ensures that sales and financial data are updated with minimal latency. For IoT, MQTT is essential to transmit sensor data in real time. For SaaS applications, WebSockets enable changes to be reflected immediately in BI dashboards. This combination of protocols shows that BI architecture should be designed based on the characteristics of data sources, not only on storage technologies.

Gaps in literature and future research

Although there is a significant increase in interest in the use of network techniques in BI, the literature still does not provide a unified model for how these protocols should be combined. Many studies focus on the technical performance of individual protocols, but there is a lack of analysis that directly relates their impact on BI quality. Similarly, few papers address the data security implications when BI is extended to multi-cloud environments. These gaps open opportunities for further research, especially on creating network-aware frameworks for data integration in BI.

Conclusion and Future Work

This paper examined how advanced network techniques can improve data integration in business intelligence (BI) systems. Starting from an analysis of traditional approaches based on ETL and ELT, it was shown that these methods, although still widely used, are limited when it comes to real-time analytical needs. The increase in data volumes, diversity of sources and the demand for fast decision-making have driven the need for more dynamic and flexible alternatives. This is where the role of new network protocols and techniques comes in, which see the network not simply as a transmission channel, but as an active element that directly affects the quality of the analysis.

The comparison made between HTTP/2, HTTP/3 (QUIC), gRPC, MQTT and WebSockets showed that each protocol carries specific advantages and limitations. HTTP/3, using QUIC, reduces latency and improves security, while gRPC offers highly efficient transmission and flexibility in communication between services. MQTT, on the other hand, is indispensable in IoT environments due to its simplicity and scalability, while WebSockets provide continuous and interactive updates for BI dashboards. The analysis clearly showed that there is no universal protocol, but that a hybrid approach, based on data and resource characteristics, is more suitable to guarantee low latency, high freshness and reliability.

The main contribution of this study lies in the direct correlation between the performance of network protocols and the quality of data integration in BI. In literature, these two areas are often treated separately: on the one hand, studies on BI architecture and on the other hand, studies on the performance of network protocols. Taken together, this paper argues that new BI architectures should be designed as network-aware, that is, aware of the impact of network choices on the entire data value chain.

However, the work developed has several limitations. The analysis is mainly theoretical and based on existing literature. No practical experimentation was conducted to empirically measure the performance of the protocols in different BI scenarios. Also, security aspects in complex environments such as multi-cloud or hybrid architecture were not extensively addressed. These gaps present an opportunity for future research that can test and validate the hypothesis raised here through experimentation and simulations.

For future work, the development of an experimental framework where network protocols are tested on real BI scenarios, such as CRM and ERP integration, IoT sensor management, or real-time updating of financial dashboards is recommended. Another research direction is the integration of edge computing and 5G/6G networking concepts, which can further reduce latency and increase integration efficiency. Also, security and privacy should be an integral part of any future study, as business data often contains highly sensitive information. In conclusion, this study highlights that network techniques are a critical component for the future of BI. Organizations that aim to build robust, scalable, and real-time analytics-capable systems must look beyond traditional approaches and adopt a strategic combination of modern protocols. Only in this way can BI evolve from a retrospective reporting tool to a predictive and proactive tool that supports real-time decision-making.

References

1. Bizer, C., Boncz, P., Brodie, M. L., & Erling, O. (2012). The meaningful use of big data: Four perspectives – four challenges. *SIGMOD Record, 40*(4), 56–60. https://doi.org/10.1145/2094114.2094129
2. Chappell, D. (2018). *Introducing gRPC: Modern open-source RPC framework*. O'Reilly Media.
3. Debnath, S., & Katal, A. (2020). Real-time data integration approaches for business intelligence. *International Journal of Advanced Computer Science and Applications, 11*(6), 145–154. https://doi.org/10.14569/IJACSA.2020.0110619
4. Fielding, R., & Reschke, J. (2014). *Hypertext transfer protocol (HTTP/1.1): Semantics and content* (RFC 7231). IETF. https://doi.org/10.17487/RFC7231
5. Huang, H., & Wu, J. (2021). Real-time big data processing with change data capture: A systematic review. *Journal of Big Data, 8*(1), 1–20. https://doi.org/10.1186/s40537-021-00477-5
6. Idzikowski, F., & Rathgeb, E. P. (2019). Performance evaluation of HTTP/3 over QUIC. *IEEE Communications Magazine, 57*(6), 24–30. https://doi.org/10.1109/MCOM.2019.1800895
7. Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB Workshop* (pp. 1–7). ACM.
8. Loukides, M., & Laurie, B. (2017). *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems*. O'Reilly Media.

9. Luckenbach, T., Gober, P., Arbanowski, S., Kayssi, A., & Schmitt, J. (2004). TinyREST: A protocol for integrating sensor networks into the Internet. In *Proceedings of the International Conference on Real-Time Systems* (pp. 153–162). IEEE. https://doi.org/10.1109/RTS.2004.1317088
10. Microsoft. (2022). Change data capture in SQL Server. https://learn.microsoft.com/sql/relational-databases/track-changes/about-change-data-capture-sql-server
11. OASIS. (2019). *MQTT version 5.0*. https://docs.oasis-open.org/mqtt/mqtt/v5.0
12. Peinl, R., & Holzschuher, F. (2015). Choosing between WebSockets and REST for real-time big data applications. *Procedia Computer Science, 53*, 130–139. https://doi.org/10.1016/j.procs.2015.07.286
13. Schroder, A., & Rahman, M. (2019). ETL vs ELT: Data integration paradigms for modern BI. *ACM SIGMOD Record, 48*(2), 29–36. https://doi.org/10.1145/3378906.3378910
14. Thalheim, B., & Tropmann-Frick, M. (2020). Data freshness and latency in BI systems. *Journal of Database Management, 31*(2), 67–85. https://doi.org/10.4018/JDM.2020040104