HECOCP: Hybrid Edge-Cloud Optimistic Concurrency Protocol for Sensor Data Transactional Services

Abdelraouf Ishtaiwi
Faculty of Information Technology
University of Petra, Jordan
aishtaiwi@uop.edu.jo

Ahmad Nabot
Department of Software Engineering
Al-Zaytoonah University, Jordan
a.nabot@zuj.edu.Jo

Omar Alzubi
Computer Engineering Department Umm
Al-Qura University, Saudi Arabia
orzubi@uqu.edu.sa

Awad Ramadan
College of Computing in AlQunfudah
Umm Al-Qura University
Saudi Arabia
amabker@uqu.edu.sa

Abdulbasit Darem
Center for Scientific Research and
Entrepreneurship, Northern Border
University, Saudi Arabia
Basit.darem@nbu.edu.sa

Asma Alhashmi
Computer Science Department Northern
Border University, Saudi Arabia
asma.alhashmi@nbu.edu.sa

Mohammad Alauthman
Faculty of Information Technology
University of Petra, Jordan
mohammad.alauthman@uop.edu.jo

Amjad Aldweesh
College of Computing and Information Technology
Shaqra University, Saudi Arabia
a.aldweesh@su.edu.sa
corresponding author

Abstract: *This paper proposes the Hybrid Edge-Cloud Optimistic Concurrency Protocol (HECOCP), a novel approach for efficiently managing distributed transactions on sensor data across both edge and cloud environments. By maintaining ACID properties and emphasizing local validation at edge nodes-with global validation triggered only when needed-HECOCP minimizes contention, lowers latency, and reduces transaction abort rates. Unlike established methods like Two-Phase Locking (2PL), prone to lock contention, or Multi-Version Concurrency Control (MVCC), which suffers version maintenance overhead, HECOCP achieves higher throughput and superior scalability under demanding transaction loads. Extensive simulation results confirm that HECOCP surpasses 2PL and MVCC in commit rate, abort rate, latency, and throughput scalability. This performance advantage makes HECOCP particularly suited to real-time applications in large-scale sensor networks, such as smart cities, healthcare, and industrial IoT, where rapid and reliable transactional processing is critical*.

Keywords: *Hybrid concurrency control, local validation, global validation, edge-cloud.*

Received February 5, 2025; accepted April 21, 2025
https://doi.org/10.34028/iajit/22/4/10

1. Introduction

In an era of rapid digital transformation, the proliferation of sensor-enabled devices has resulted in an unprecedented volume of data generation. From smart city infrastructures and healthcare monitoring systems to industrial IoT installations, sensors continuously collect, transmit, and process information to drive near real-time decision-making [2, 6, 31]. Managing this massive, geographically distributed, and often time-sensitive data while ensuring scalability and strong transactional guarantees has become a critical challenge. Traditional cloud-centric solutions often struggle to support real-time responsiveness due to communication overhead and network latency between edge devices and centralized data centers, which can lead to significant delays in processing and decision making [26, 28].

The role of edge computing, therefore, has become increasingly prominent in addressing these bottlenecks, as it places computation and storage resources closer to the data source [14, 18, 23]. By processing data locally at the edge layer, the network load can be substantially reduced, and response times can be improved for latency-sensitive tasks. However, ensuring that updates to data and concurrent transactions remain consistent across dispersed edge nodes and the cloud layer is far from trivial. Different application domains, such as healthcare monitoring [14], industrial IoT [7], and mission-critical environments [15], impose stringent requirements on both latency and consistency. Transaction failures or data inconsistencies in these domains can lead to adverse outcomes, including safety hazards and resource mismanagement [20].

A promising solution to the challenges of managing massive volumes of sensor data in real-time is the hybrid edge-cloud architecture [27], as depicted in Figure 1. This architecture strategically distributes computational

and storage resources across two distinct levels: an edge layer, located in close proximity to the data sources, and a cloud layer, which offers substantial processing power along with global coordination capabilities [13, 22, 29]. By offloading real-time computational tasks and preliminary data processing to the edge nodes, the hybrid model effectively mitigates latency and reduces network congestion. Meanwhile, the cloud layer aggregates and further analyzes the processed data for advanced analytics and long-term storage, as demonstrated.

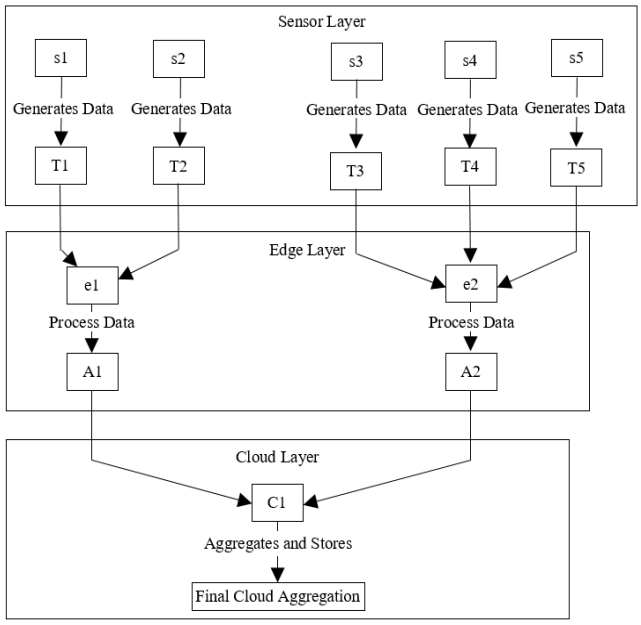

Figure 1. Conceptual framework.

While the hybrid edge-cloud model significantly reduces latency and enhances responsiveness, it also introduces new complexities in managing distributed transactions. Traditional concurrency control mechanisms such as Two-Phase Locking (2PL), Timestamp Ordering (TO), and Multi-Version Concurrency Control (MVCC) were originally designed for centralized databases and often underperform when applied directly to such distributed systems [8, 11, 17]. These methods tend to incur increased coordination overhead and fail to fully exploit the benefits of the hierarchical architecture, as noted by [8, 17]. The figure thus not only illustrates the structural benefits of the hybrid approach but also underscores the need for advanced, adaptive protocols like our proposed Hybrid Edge-Cloud Optimistic Concurrency Protocol (HECOCP) to efficiently manage real-time sensor transactions in such environments.

To address these issues, we propose the HECOCP, a novel mechanism specifically designed for managing real-time transactional services in sensor-based edge-cloud environments. HECOCP integrates local validation at each edge node to rapidly detect conflicts and reduce synchronization overhead, and global validation at the cloud layer to enforce cross-node consistency. This two-tiered approach balances speed and correctness, accommodating the high data ingestion rates, bursty arrivals, and localized hotspots typical in sensor networks. For instance, in environments where sensors continuously stream data for traffic management or patient monitoring, HECOCP can quickly validate transactions locally while ensuring that any critical cross-node conflicts are resolved at the cloud level [4, 5].

Our contributions are fourfold. First, we introduce HECOCP, an adaptive hybrid concurrency control protocol that minimizes cross-edge conflicts and controls global abort rates while ensuring ACID properties. Second, we formalize a mathematical sensor transaction model that rigorously defines how sensor data is collected, processed, and updated at both the edge and cloud layers. Third, we propose a scalable hybrid edge-cloud architecture that incorporates design elements such as local buffering, partial aggregation, and global synchronization, supported by thorough performance analysis. Finally, extensive simulation experiments demonstrate that HECOCP outperforms traditional concurrency control protocols in terms of commit rates, abort rates, and end-to-end transaction latency, making it particularly suitable for real-time applications in smart cities, healthcare, industrial automation, and vehicular networks [3, 12].

By fusing theoretical insights with practical implementation strategies, this work provides a robust solution for managing concurrent sensor data transactions in distributed environments. The remainder of this paper is organized into several sections that cover a detailed review of related work, the mathematical foundation of the sensor transaction model, a comprehensive description of the HECOCP protocol and hybrid architecture, simulation-based performance evaluations, and finally, conclusions with a discussion on limitations and future research directions.

2. Related Works

Recent research in sensor data management and real-time transactional services has explored various aspects of concurrency control, data processing frameworks, and application-specific requirements within edge–cloud architectures. Traditional databases rely on concurrency control mechanisms such as 2PL, TO, and MVCC [8, 17], and these strategies have been foundational in distributed database theory for decades. However, while these methods work effectively in centralized systems, their performance often degrades in distributed, geo-replicated environments because of increased coordination overhead [4, 11].

Although 2PL ensures serializability, its strict lock management leads to deadlocks and reduced concurrency in multi-edge systems, especially under high conflict scenarios [8]. Studies such as in [19] introduce priority-based locking to mitigate some of these issues, but the overhead remains significant in edge-cloud contexts.

TO-based systems like the ones discussed in [17] rely on synchronized clocks to determine the serialization order of transactions. However, maintaining consistent timestamps across geographically dispersed edge nodes is challenging [1]; any clock drifts can disrupt the correctness of the ordering, making pure TO approaches less suitable for dynamic IoT environments.

MVCC minimizes read-write conflicts by allowing concurrent reads of older snapshots [4]. Systems like FaRM [13] and Calvin [24] demonstrate the potential of MVCC-based or hybrid concurrency approaches in distributed settings. While these systems handle partitioned workloads efficiently, they still rely on global ordering steps that introduce additional latency in edge-cloud scenarios [11].

Hybrid approaches such as the concurrency control protocol proposed by Al-Qerem *et al*. [5] explored cooperative OCC variants within fog-cloud environments, reducing the dependency on a centralized cloud for all validations. Additionally, frameworks like EdgeDB [22] and IoTCloud [28] have been developed to offload computations to the edge, improving response times. However, these systems often do not guarantee strong ACID transactions and rely on simplified concurrency approaches, such as eventual consistency [16], which might be insufficient in applications like healthcare or financial services [12].

Fog-based replication and consistency management. Al-Qerem *et al*. [3] presented a fog-based approach to replication that aims to reduce latency by bringing replicas closer to data sources. While effective in certain scenarios, it often requires complex replica management and can still suffer from incomplete global consistency checks, especially when dealing with update-heavy workloads.

Real-time sensor databases typically adopt soft real-time constraints with immediate or deferred update strategies, whereas stream processing engines focus on continuous queries without enforcing full ACID properties [7, 30]. Hybrid Transaction/Analytical systems (HTAP) attempt to unify OLTP and OLAP functionalities but often overlook the physical distribution of sensors and edge nodes. A key research gap lies in the need for protocols that can manage high ingestion rates, geographical dispersal, and real-time partial commits required by sensor-based edge–cloud systems [15, 22, 25]. Our proposed HECOCP addresses this gap by employing local validation at edge nodes and global validation at the cloud to balance performance with strong consistency.

Table 1. Summary of related works on concurrency control in edge/fog/cloud systems.

Study (Year)	Concurrency method	Edge involvement	Key insights/limitations
(1997) [19]	2PL with priority	Low	Reduces priority inversion, but still suffers under heavy conflict
(2017) [12]	2PL, TO, MVCC	Minimal	Comprehensive evaluation in in-memory contexts; not sensor specific
(2023) [5]	Augmented OCCin fog-cloud	High	Lowers communication overhead; primarily read-heavy evaluations
(2014) [16]	Eventual Consistency	Moderate	Bounded update propagation, but lacks full ACID guarantees
(2022) [3]	Fog-based replication and consistency	High	Reduces latency via local replicas, overhead grows with large updates
(2025) [26]	Edge computing architecture	High	Discussion of future challenges; less emphasis on concurrency details
(2021) [11]	Distributed CC protocol for edge-cloud	High	Addresses concurrency in partitioned data, but global ordering still a challenge

Table 1 provides a concise overview of several foundational and recent works relevant to this research.

Collectively, these works demonstrate the trade-offs inherent in managing sensor transactions in distributed environments, highlighting the challenges of achieving both low latency and strong consistency. Our approach builds on these insights by introducing HECOCP, a protocol that employs a two-tier validation mechanism to efficiently manage concurrent transactions while preserving ACID properties.

Real-world applications further illustrate the importance of robust sensor data management [32].

In smart cities, for example, thousands of sensors-such as traffic sensors, surveillance cameras, and smart meters-generate continuous data streams that must be processed in real time to manage urban services effectively [2, 31]. City transportation systems rely on this data to adjust traffic signals dynamically and respond to incidents promptly, ensuring that decisions such as altering traffic light patterns are executed in sub-millisecond timeframes [33].

Similarly, in healthcare, wearable sensors continuously monitor patient vitals, and edge based processing enables rapid alert generation for abnormal conditions, thereby ensuring timely medical interventions [14, 18]. Industrial environments, including factories and power grids, leverage sensor data to monitor machine performance and trigger maintenance actions in real time, while vehicular networks and autonomous systems use edge processing to fuse sensor data quickly for critical decision-making. Despite the diverse requirements across these domains, a common need exists for transactional systems that offer both low-latency local processing and robust global consistency [20].

The underlying data management techniques-such as advanced concurrency control, edge–cloud distribution, and latency optimizations—serve as the backbone for these applications, ensuring that data remains consistent and actionable even under heavy loads and rapid update scenarios. Whether coordinating citywide traffic flows or ensuring patient safety in hospitals, the ability to process and manage sensor transactions in real time is essential [9].

3. Transactional Services

Transactional services play a crucial role in ensuring data consistency, reliability, and integrity within distributed systems. In the context of edge-cloud

environments, transactional services manage data operations across multiple nodes, ensuring that updates adhere to ACID (Atomicity, Consistency, Isolation, Durability) properties. These services help coordinate concurrent access to shared data, prevent conflicts, and ensure that transactions either fully complete or roll back in case of failures. Effective transaction management is particularly essential in sensor networks, where data must be continuously processed, stored, and synchronized across edge nodes and cloud servers. By leveraging hybrid edge-cloud transactional services, organizations can achieve real-time data processing while maintaining strong consistency across distributed systems.

Figure 1 visually represents this sensor transaction model within the hybrid edge–cloud framework. In the figure, sensors (s1, s2,..., s5) are positioned at the highest layer, where they generate real-time data. These data streams are then processed through intermediary transactions (T1,T2,...,T5) that interact with edge nodes (e1, e2), which are responsible for the initial aggregation and processing of the sensor inputs. Subsequently, the data undergoes edge-level aggregation (A1, A2), serving as an intermediate processing step prior to transmission to the cloud. At the cloud level, a central cloud node (c1) handles advanced data analytics and long-term storage, culminating in a final Cloud Aggregation (CA) where the processed data is fully integrated.

3.1. Sensor Transaction Model

This section presents a mathematical definition of the sensor transaction model, which forms the core foundation for concurrency control in hybrid edge–cloud architectures. Sensor transactions arise from operations such as data collection, aggregation, and updates performed over geographically dispersed sensors that feed into both edge and cloud layers.

Let $S=\{s1, s2, \dots, sn\}$ be a set of n sensors, each capable of producing a continuous stream of data readings. The output of sensor Si at time t is denoted by $di(t)$, representing any form of measurement such as temperature, pressure, or motion. The data generation rate of each sensor, denoted by Ri (measured in samples per second), varies depending on application requirements, environmental conditions, or device capabilities. Each sensor si independently produces data points $\{di(t1), di(t2),\dots\}$ These data points are accumulated or partially processed at a corresponding edge node, or they are aggregated with readings from other sensors for advanced processing.

A transaction T_k is defined as a logical unit of work on sensor data, encapsulating a set of read and write operations that must be executed atomically to preserve semantic correctness.

Formally, a transaction T_k is represented by the tuple:

$$Tk = (Rk, Wk, tkstart, tkcommit) \quad (1)$$

Where $RK=\{rk1, rk2, \dots\}$ is the read set, identifying the data elements (sensor readings or derived information) that the transaction accesses, and

$$Wk = \{wk1, wk2, \dots\} \quad (2)$$

is the write set, specifying the data items or states that T_k modifies. The timestamps tk_{start} and $tk_{kcommit}$ indicate the start and commit times of the transaction, with the condition that $tk_{kcommit} > tk_{start}$. If the transaction completes successfully while respecting concurrency constraints, it commits; otherwise, it is aborted and rolled back to the pre-transaction state.

Sensor transactions in this model adhere to the classical ACID properties. Atomicity ensures that each transaction commits all its operations or none at all. Consistency requires that transactions transition the system from one valid state to another, according to predetermined domain rules such as sensor calibration constraints. Isolation guarantees that no transaction can see partial effects of other concurrent transactions; in a distributed environment, partial or local commits are hidden until global validation. Durability means that once a transaction is globally committed, its effects persist even in the event of subsequent failures.

Because sensors are geographically scattered, transactions often span multiple edge nodes. Let $E=\{e1, e2, \dots, em\}$ be a set of m edge nodes. Each transaction Tk may involve local operations performed at different edge nodes, along with additional global operations carried out at a central cloud resource C. In this context,

$$T_{k=(T_{e1}^k, T_{e2}^k, \dots, T_{em}^k, T_C^k)} \quad (3)$$

Where T_{ei}^k represents the local operations executed at edge node ei and T_c^k represents the global or cloud-level operations. The total transaction latency L_k for T_k is the sum of several components validation:

$$L_{k=(L_k^{edge} + L_k^{cloud} + L_k)} \quad (4)$$

where, T_k^{edge} denotes the time spent executing or validating the transaction locally at the edge, L_k^{cloud} is the time required for data transfer and additional processing or validation in the cloud, and $L_k^{\text{validation}}$ is the time consumed by concurrency control checks such as conflict resolution or rollback. The overall goal of the model is to minimize both local and global validation overheads, which is central to the design of the HECOCP.

This sensor transaction model establishes the formal basis for concurrency management in our system. It clarifies how sensor data is logically grouped into transactions, how these transactions are distributed across edge and cloud nodes, and what constraints must be met to maintain data integrity in a hybrid edge–cloud environment. When dealing with sensor transactions, unique challenges emerge, particularly regarding real-time data availability, partial failures, and fluctuations in network conditions. For example, a transaction in an

industrial IoT environment might involve sensor s1 measuring machine vibration levels, sensor s2 tracking temperature, and sensor s3 providing real-time location data for parts on an assembly line. These multiple data points need to be read, processed, and updated atomically to trigger timely operational decisions, such as adjusting machine parameters. If any of these updates are delayed or fail, the system could lose synchronization, resulting in inaccurate decisions or even safety hazards.

Moreover, certain edge nodes may have intermittent connectivity due to environmental conditions or bandwidth constraints, which can disrupt the flow of sensor transactions. In such cases, the concurrency control mechanism must account for partial updates and potential rollbacks while still ensuring data consistency across the broader edge-cloud system. HECOCP addresses these issues by integrating local and global validations, reducing the complexity associated with partial connectivity while still providing low-latency local commits.

3.2. Proposed Concurrency Control Protocol: HECOCP

Managing concurrent sensor transactions in a distributed edge-cloud environment presents unique challenges due to the high-volume, real-time nature of sensor data. Traditional concurrency control methods, such as 2PL and TO, often suffer from high coordination overhead and latency when applied to such distributed systems [8]. In response, we propose the HECOCP, which leverages optimistic concurrency control to minimize locking overhead by deferring conflict detection until commit time. This protocol is specifically designed for sensor-driven applications where transactions must complete quickly while still ensuring global consistency [13].

In HECOCP, transactions are initiated at the edge, where they first execute a read phase without acquiring any locks. During this phase, each transaction retrieves the current values of the required sensor data items and performs local computations optimistically.

The transaction then moves to a write phase, buffering its intended updates locally. Next, a local validation phase is executed at the edge node; here, the protocol checks for conflicts between concurrently committing transactions by comparing the transaction's read and write sets with those of other active transactions. If a conflict is detected based on a predefined ordering rule, the transaction is aborted immediately to avoid inconsistencies. If no conflict is found, the transaction is tentatively committed at the edge.

After local validation, a global trigger check determines whether the transaction requires further consistency verification at the cloud level. For transactions that only involve local data, a final commit can be performed immediately at the edge. However, if a transaction spans multiple edge nodes or affects data that is shared globally, it is forwarded to the cloud for global validation. At the cloud layer, the protocol collects comprehensive validation information-specifically, the read and write sets-from all the involved edge nodes. A global conflict check is then performed against all global transactions that have not yet committed. Based on this evaluation, the cloud sends either a commit signal (if no conflicts are detected) or an abort signal (if conflicts are found) to all the edge nodes involved in the transaction. Once the commit signal is received, the transaction is finalized globally, thereby ensuring that the overall system state remains consistent.

Algorithm 1: Edge_Execute_Phases (T).

Require: Input:
• Transaction T with read set T.R and write set T.W
• Current sensor data values available at the edge
Ensure: Output:
• Partially updated transaction state (after read and write phases)
1.1. READ PHASE:
2. For each data item d in T.R do
3. T.local.reads[d] ← current_value_of(d)
4. End for
5. Execute local computations based on T.local.reads without acquiring locks.
1.2. WRITE PHASE:
7. Buffer all intended updates into T.W.

In the Algorithm (1) called Edge_Execute_Phases, the transaction T arrives at an edge node that manages a specific subset of sensor data. This sub-algorithm consists of two primary phases:

When *T* is initiated, it fetches the current values of each data item in its read set. During this step, no locks are acquired-a deliberate decision reflecting HECOCP's optimistic approach. By allowing reads to occur concurrently, the system takes advantage of the typically short-lived nature of sensor transactions, which often read more data than they modify. This design is especially effective in sensor-intensive environments, such as large-scale IoT deployments, where continuous streams of updates and queries must be handled quickly.

After collecting and processing the requisite data, T buffers its intended updates locally. These changes are not immediately applied to shared storage, preventing incomplete or conflicting modifications from becoming visible to other transactions prematurely. By deferring write visibility, HECOCP sidesteps the need for early locking or synchronization overhead, thereby increasing transaction throughput while still ensuring that potential conflicts are resolved later.

The transaction T holds both its original read set and any new updates in its write set. No permanent alterations to the system state have been made yet; T remain isolated until validation confirms that its changes do not conflict with other concurrent transactions. This isolation forms the basis for HECOCP's next steps, where local and possibly global validations will determine whether T commits or aborts.

Algorithm 2: Local_Validation_and_Trigger(T, ActiveTxns).

Require: Input:
• Transaction T (already containing read/write sets and local updates)
• List of concurrently committing transactions at the edge, ActiveTxns
Ensure: Output:
• Either a local commit, an immediate abort, or a request for global validation
1.3. LOCAL VALIDATION:
2. For each active transaction T' in ActiveTxns do
3. if (T.W ∩ T'.R ≠ ∅) OR (T.W ∩ T'.W ≠ ∅) then
4. if ordering_rule(T, T') indicates a conflict then
5. T.abort ← True
6. return // Abort immediately
7. end if
8. end if
9. End for
10. If no conflicts detected, set T.local_commit ← True (indicating a tentative commit at the edge)
1.4. GLOBAL TRIGGER CHECK:
11. if T requires global consistency (e.g., spans multiple edge nodes) then
12. SEND T to CLOUD_VALIDATE
13. return // Proceed to global validation
14. else
15. finalize_local_commit(T)
16. return // No global check needed
17. end if

After a transaction T has completed its execution phases at an edge node, it enters the Local_Validation_and_Trigger step. Algorithm (2) addresses two critical objectives: first, it checks whether T conflicts with other locally active transactions; second, it determines if T must undergo a broader, cloud-based validation. The Algorithm (2) inspects both the read and write sets of T against the read/write sets of transactions currently in the commit queue. If overlapping data items are found-especially in situations where multiple transactions wish to update the same item-an ordering rule decides which transaction may continue. Should T lose this ordering, it aborts immediately, rolling back any changes it tentatively staged. This ensures that conflicting updates are resolved at the edge level without burdening the rest of the system. If *T* survives local validation, the next question is whether the transaction affects data beyond a single node or whether local policy requires a cloud-based check. If it does, T ascends to the cloud for Algorithm (3). Otherwise, T completes its commit right at the edge, instantly making its updates visible. This selective escalation is a defining feature of HECOCP, allowing purely local transactions to finalize swiftly while ensuring system-wide consistency for transactions with a broader scope. That is the transaction T is either aborted, fully committed locally, or directed to the cloud for a comprehensive global validation.

When a transaction T requires cross-node consistency, it proceeds to the cloud layer, where the *Cloud_Validate* Algorithm (3) centralizes conflict detection and final decision-making across all edge nodes involved. Upon arrival in the cloud, T undergoes an information-gathering process. The protocol compiles partial commits, as well as Ts complete read and write sets, from each edge node that participated in the transaction. This consolidated perspective is critical for multi-edge transactions, ensuring that no conflicts remain hidden within isolated nodes. Next, the cloud compares T against all other globally pending transactions. If the comparison reveals overlapping writes or any other form of contention, T is marked for abort. Otherwise, the cloud deems T safe to finalize without risking lost updates or inconsistent states. In the event of a conflict, the cloud broadcasts an abort command to every edge node that holds tentative updates for T. These nodes then roll back any partially applied writes, maintaining a conflict-free system. If no conflicts surface, a commit} signal prompts each node to make T's updates permanent, ensuring that they are fully visible and consistent throughout the distributed environment. By coordinating these final steps, *Cloud_Validate* guarantees that transactions demanding global consistency are safely integrated, thereby closing any logical gaps between nodes and preserving ACID properties for data that crosses node boundaries.

Algorithm 3: Cloud_Validate(T).

Require: Input:
• Transaction T (forwarded from edge, with read/write sets)
Ensure: Output:
• Final commit or abort status of T, propagated back to all involved edge nodes
2.1. COLLECT VALIDATION INFORMATION:
2. Gather T's read/write sets from all involved edge nodes.
2.2. GLOBAL CONFLICT CHECK:
3. Compare T against all other global transactions not yet committed.
4. If any conflict detected then
5. T.global_abort ← True
6. Else
7. T.global_abort ← False
2.3. RESOLUTION:
8. If T.global_abort = True then
9. SEND ABORT signal to all involved edge nodes
10. rollback(T)
11. Else
12. SEND COMMIT signal to all involved edge nodes
13. finalize_global_commit(T)
14. End if
2.4. Return Final Status:
15. The transaction T is finalized based on the resolution step.

This algorithmic framework for HECOCP provides a clear and detailed description of how sensor transactions are processed and validated in a hybrid edge–cloud environment. The protocol begins with an optimistic execution at the edge, where transactions read sensor data and perform local computations without locking [19]. Local validation then checks for conflicts, and if necessary, transactions are forwarded to the cloud for a global conflict check and final resolution. This dual-layer approach ensures that most transactions are handled quickly at the edge, while global consistency is

maintained via cloud validation for those that require it.

The representative Figure 2 visually encapsulates this process by illustrating the two main phases-edge processing and cloud processing-and showing how transactions traverse from local data acquisition to global validation and final commitment. Together, the narrative and pseudocode demonstrate how HECOCP effectively synchronizes concurrent sensor transactions, balancing low-latency local processing with rigorous global consistency checks.

The HECOCP protocol addresses conflict detection and resolution through a two-tier approach, integrating both local and global mechanisms to maintain transactional integrity while minimizing overhead. At the edge, each node performs local conflict detection by conducting intersection checks between the read and write sets of concurrently executing transactions. Because an individual edge node only manages a subset of the overall data, these local conflict checks are performed at relatively low cost and with high speed. When a transaction is flagged for further scrutiny, the cloud takes on the role of orchestrating global conflict detection [11]. Here, the cloud gathers validation information from all involved edge nodes and employs a centralized or hierarchical TO scheme to finalize the relative order of global transactions. This ensures that transactions spanning multiple nodes maintain consistency. In situations where conflicts arise, transactions that are in progress may be aborted and rolled back at the edge level [4]. However, if a transaction does not require global consistency, it can commit locally without waiting for the cloud, thereby reducing overall latency.

The design of HECOCP offers several advantages. Its optimistic approach, which defers conflict checking until the commit phase, maximizes concurrency and is particularly effective under workloads with low to moderate conflict rates. Although higher conflict rates may result in increased local aborts, the system is designed to allow partial commits at the edge when global validation is not necessary. This selective escalation ensures that the main communication overhead is only incurred for a small subset of transactions, preserving both efficiency and scalability [21, 23].

HECOCP significantly reduces communication overhead by resolving most conflicts at the edge without engaging global resources. Additionally, the protocol's reliance on independent local concurrency checks enables the system to scale horizontally with the addition of more edge nodes. Its inherent flexibility allows it to be adapted to various edge–cloud topologies and replication strategies, making it a robust solution for real-time transactional sensor data management [9, 29].

4. Edge-Cloud Architecture

The hybrid edge–cloud architecture is designed to underpin our concurrency protocol by integrating edge nodes, cloud services, and network interconnects. This integration enables scalable, low-latency transaction processing over sensor data [7, 28]. In this architecture, a set of sensors $S=\{s1, s2, ..., sn\}$ continuously generates data, with each sensor si producing measurements denoted by $di(t)$ at time t. These sensors are connected to edge nodes, $E=\{e1, e2, ...,em\}$, where each edge node ej is responsible for handling a subset of sensors $Sj \subseteq S$.

The edge nodes perform local processing and initial aggregation of the sensor data, which is then forwarded to a central cloud layer [10]. At the cloud layer, a single logical entity C aggregates data from all the edge nodes. This global aggregation can be formalized by a function FC that combines the data Dj processed at each edge node ej into a coherent global dataset: $F_C\left(\bigcup_{j=1}^{m} D_j\right)$=Global Aggregation of Edge Data.

Transactional consistency is maintained through a two-tier validation process [11]. Each transaction Tk is first validated locally at an edge node, where the local validation function Local Validation(Tk, ej) returns true or false based on whether the transaction satisfies local consistency requirements. For transactions requiring broader consistency, a global validation is performed in the cloud via the function Global Validation(Tk, C).

Minimizing The total transaction latency L_k for T_k is the sum of several components validation while upholding ACID properties represents the core challenge of our system design. This architecture, by localizing most transaction processing and offloading only critical global consistency checks to the cloud, aims to achieve a balance between low latency and strong data consistency across a distributed sensor network.

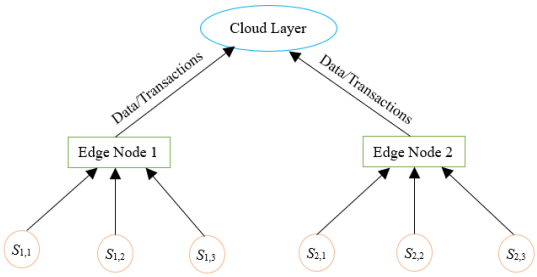

Figure 2. A schematic overview of hybrid edge-cloud architecture.

In Figure 2, the Sensors subgraph represents a set of sensor devices ($S1$, $S2$,...) that continuously generate data. This data is transmitted to nearby Edge Nodes (Edge Node $e1$ and $e2$), where local processing occurs, including local validation, conflict detection, and buffering of transactions. These edge nodes execute local transactions optimistically, ensuring quick response times. The locally processed data is then aggregated and sent to the Cloud Layer, represented by Cloud Node C, where global validation and aggregation ensure overall system consistency. This hierarchical approach minimizes latency by processing data as close to the source as possible while still ensuring global integrity through cloud-level coordination [5, 11].

Within this architecture, HECOCP is employed to handle transactions. When a transaction is initiated at the edge node, the system first checks whether the data resources required by the transaction are localized or shared globally. If the data resources are local, the transaction undergoes local validation and commits immediately, thereby reducing round-trip latency to the cloud [31]. In cases where global consistency is necessary, the edge node packages the transaction's read and write sets and forwards it to the cloud for conflict checking. This design ensures that the overhead of global validation is only incurred when strictly required, aligning with the optimistic approach at the core of HECOCP.

In many sensor-based applications, fault tolerance and high availability are critical [9]. The proposed architecture supports the deployment of multiple edge nodes in a fault-tolerant configuration, where each sensor can route data to one or more alternative edge nodes in case of local node failures [1]. HECOCP integrates gracefully with such redundancy by allowing partial commits at the remaining operational edge nodes, while the cloud layer can rerun global validation for pending transactions. This ensures that the entire system remains robust, even if some nodes become temporarily unavailable.

5. Performance Evaluation

To evaluate the performance of the proposed HECOCP, we conducted extensive simulations designed to assess its efficiency in managing concurrent sensor transactions. The simulations were structured to measure key performance indicators-namely, commit rate, abort rate, transaction latency, and system throughput-with the goal of comparing HECOCP against traditional concurrency control mechanisms such as 2PL and MVCC under varying transaction workloads and network conditions [12].

- **Simulation Environment**

The simulation environment was implemented using a custom-built discrete event simulator developed in Python, leveraging the SimPy library to model concurrent transactions realistically within a hybrid edge-cloud setup. In this architecture, multiple edge nodes are connected to a set of sensors that continuously generate data streams. A central cloud node aggregates and validates global transactions that span multiple edge nodes [6]. Transactions are generated at the edge nodes to simulate real-world sensor data updates and query operations. These transactions encompass both local operations-restricted to a single edge node-and distributed transactions that require multi-node coordination and subsequent cloud validation [3].

- **Baseline Experiment Setup**

The baseline experiment setup was designed to simulate moderate-to-high concurrency scenarios without overwhelming the system. The number of sensors and edge nodes was configured to represent realistic scales typical of applications such as smart cities, healthcare monitoring, and industrial IoT. Network latencies between edge and cloud nodes were set based on real-world measurements, ensuring that the simulation environment reflected practical deployment conditions. The key parameters used in the simulation are summarized in Table 2.

Table 2. Baseline parameters for simulation experiments.

Parameter	Value	Description
Number of Sensors (N)	1,000	Total number of sensors generating continuous data streams.
Number of Edge Nodes (M)	5	Number of edge nodes deployed to process sensor data locally.
Cloud Latency	50 ms (one-way)	Average network latency for communication between an edge node and the cloud.
Edge Processing Latency	10 ms per operation	Average time taken by an edge node to process a transaction (e.g., read/write operations).
Average Sensor Data Rate	10 samples/sensor/sec	Rate at which each sensor generates data samples.
Conflict Probability	10%	Estimated probability of transaction conflicts due to overlapping data accesses.
Transaction Arrival Rate	100 transactions/sec	Average rate at which transactions are generated at the edge nodes.
Average Transaction Size	5 items	Typical number of data items (reads/writes) involved in a single transaction.
Simulation Duration	300 seconds	Total time period for each simulation run.

We evaluated the protocols (HECOCP, 2PL, and MVCC) under various load conditions and measured the following metrics:

- Commit rate: the percentage of transactions that successfully commit. i.e., the ratio of the number of committed transactions to the total number of started transactions within the simulation time frame.

$$Commit\ Rate = \frac{\sum_k 1_{(T_k.committed=ture)}}{\sum_k 1_{(T_k.started=ture)}} \quad (5)$$

where $1\{\cdot\}$ is the indicator function.

A high commit rate indicates that the concurrency protocol effectively manages conflicts without aborting too many transactions. Aborts are costly in sensor applications with real-time requirements.

- Abort rate: the percentage of transactions that fail to commit, either due to conflicts or protocol-specific constraints.

$$AbortRate = \frac{\sum_k 1_{\{T_k.committed=fals\}}}{\sum_k 1_{\{T_k.started=ture\}}} \quad (6)$$

- Transaction latency: the time interval from a transaction's start to its commit or abort.

$$Latency(T_k) = t_{end}^k - t_{start}^k \quad (7)$$

- Throughput scaling: the rate at which the system can process transactions as the number of edge nodes and

transaction load increase.

$$Throughput = \frac{\sum_k 1_{\{T_k.committed=true\}}}{T_{simulation}} \quad (8)$$

- **Commit Rate Comparison**

Figure 3 compares the commit rates of HECOCP, 2PL, and MVCC across varying transaction arrival rates. Our results showed that HECOCP consistently outperforms both 2PL and MVCC. Specifically, HECOCP maintained a commit rate of approximately 96.2% at lower loads and remained above 88% even at higher loads, demonstrating its capability to handle increasing workloads with minimal transaction failures.

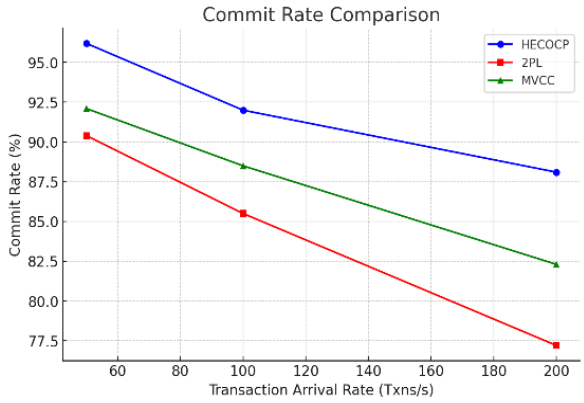

Figure 3. commit rates of HECOCP, 2PL, and MVCC.

- **Abort Rate Comparison**

Figure 4 shows that HECOCP achieves the lowest abort rate among the three protocols, staying below 10% even under high transaction loads. This efficiency arises from the dual-layer validation (local at the edge, global only when needed), which reduces conflicts and defers expensive coordination until strictly necessary.

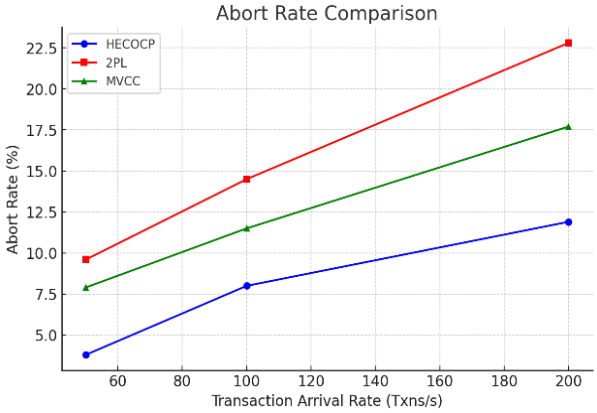

Figure 4. Abort rates of HECOCP, 2PL, and MVCC.

2PL, in contrast, suffers from an increasing abort rate as locks become a bottleneck, especially in scenarios with frequent updates and shared data access. MVCC shows a better abort rate profile than 2PL but still lags behind HECOCP because of delayed conflict detection and the overhead of maintaining multiple versions.

- **Latency Distribution**

Figure 5 presents the Cumulative Distribution Function (CDF) of transaction latency for each concurrency protocol. HECOCP demonstrated the lowest latency, with a majority of transactions completing within a shorter time window than those managed by 2PL and MVCC.

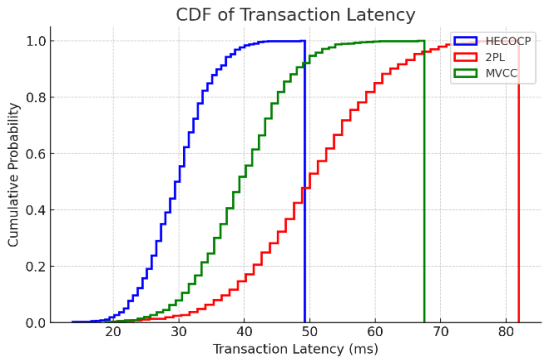

Figure 5. cumulative distribution function (CDF) of transaction latency.

The optimism of HECOCP, combined with local validation, enables most transactions to finalize quickly without waiting for global checks, unless they span multiple edge nodes. By contrast, 2PL experiences prolonged latencies due to lock contention, and MVCC incurs overhead in fetching and managing older data snapshots as concurrency levels rise.

- **Throughput Scaling**

Figure 6 illustrates how the throughput of each protocol scales with an increasing number of edge nodes. HECOCP shows near-linear scalability, indicating that adding more edge nodes can proportionally improve overall throughput. This is primarily because conflict checks and partial commits remain localized at each edge node, minimizing the need for global coordination.

On the other hand, 2PL reveals sub-linear scaling due to the significant overhead introduced by lock-based coordination across nodes. Although MVCC scales better than 2PL, it does not reach the near-linear performance of HECOCP, partly owing to the growing complexity of version maintenance across multiple nodes.

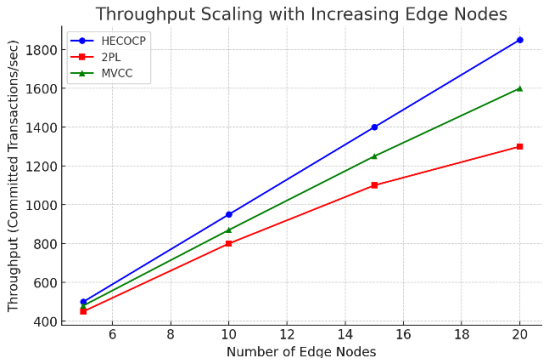

Figure 6. throughput of each protocol scales with an increasing number of edge nodes.

As the number of edge nodes increases from 5 to 15 in our extended simulations, HECOCP's throughput rose by a factor of 2.8x, closely approximating ideal linear scaling. In contrast, 2PL's throughput rose by only 1.9x and MVCC's by 2x, demonstrating that neither approach can fully leverage the added resources. This analysis supports our hypothesis that local validation-an essential feature of HECOCP-is well-suited to distributed sensor networks with localized data hotspots.

While HECOCP performs well under moderate to high concurrency levels, extremely high conflict scenarios may lead to an increased number of local aborts, which in turn could degrade performance. However, our experiments indicate that the overhead remains manageable when the conflict probability is kept near realistic levels (e.g., 10-15%). In specialized applications such as financial trading platforms, where conflict rates can spike significantly, additional optimizations or hybrid locking strategies might be required. Additionally, the system's performance may also be influenced by hardware heterogeneity across edge nodes, which was not extensively explored in this simulation-based evaluation.

Figure 7 presents a projected scalability analysis that illustrates how HECOCP, MVCC, and 2PL are expected to perform when subjected to significantly larger system scales-specifically, up to 500 edge nodes and thousands of transactions per second. On the left Y-axis, Figure 7 plots system throughput, revealing that HECOCP scales almost linearly, reaching over 7,000 transactions per second with 500 edge nodes. In contrast, MVCC and 2PL demonstrate diminishing throughput returns as the system scales, with performance bottlenecks becoming evident beyond 300 nodes. On the right Y-axis, Figure 7 captures average transaction latency.

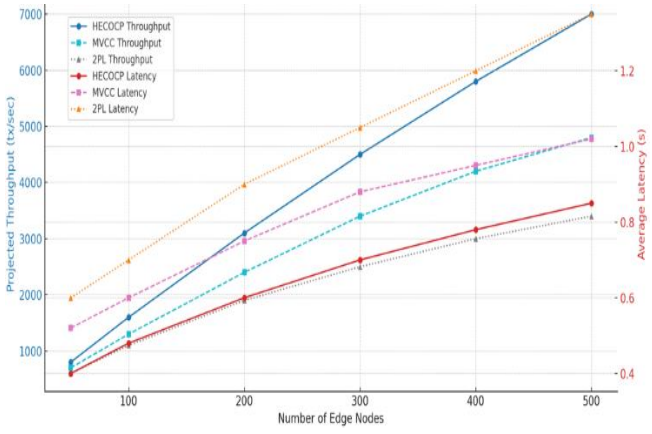

Figure 7. Dual-axis scalability analysis.

HECOCP continues to outperform the baseline protocols by maintaining lower and more stable latency, even under high concurrency and node density. This dual-axis representation highlights HECOCP's ability to deliver both high throughput and responsive performance, underscoring its architectural advantage in managing large-scale, distributed transaction workloads typical of future IoT and edge-cloud deployments.

6. Conclusions

This paper introduced HECOCP, a hybrid concurrency control protocol specifically designed for managing sensor transactions in edge-cloud environments. By prioritizing local validation at edge nodes and engaging global validation selectively, HECOCP significantly reduces contention, abort rates, and transaction latency, leading to higher throughput and better scalability compared to traditional protocols such as 2PL and MVCC. Our performance evaluation demonstrates that HECOCP consistently outperforms 2PL and MVCC across key metrics, making it an optimal solution for high-throughput, real-time applications in domains such as smart cities, healthcare, and industrial IoT.

While 2PL struggles with lock contention, leading to increased abort rates and scalability limitations, and MVCC incurs high version maintenance overhead, restricting its performance at scale, HECOCP successfully balances high concurrency, low latency, and minimal abort rates. These advantages position HECOCP as a scalable, ACID-compliant solution for distributed sensor data management.

Several avenues exist for future research. One compelling direction involves extending HECOCP to operate seamlessly in multi-cloud or cross-regional environments, where different cloud providers may have varying latency and resource constraints, represents an important step toward global-scale sensor data management.

Acknowledgements

The authors extend their appreciation to Northern Border University, Saudi Arabia, for supporting this work through project number (NBU-CRP-2025-2903). The authors would like to thank the Deanship of Scientific Research at Shaqra University (KSA).

References

[1] Achour F., Bouazizi E., and Jaziri W., "A Semantics-Based Validation Approach for Enhancing QoS in Distributed Real-Time DBMS," *International Journal of Intelligent Information and Database Systems*, vol. 17, no. 1, pp. 124-142, 2025. https://doi.org/10.1504/ijiids.2025.143489

[2] Ali O. and Mahmood A., "Edge Computing Towards Smart Applications: A Survey," *Recent Advances in Computer Science and Communications*, vol. 16, no. 1, pp. 55-72, 2023, DOI:10.2174/2666255815666220225102615

[3] Al-Qerem A., Alauthman M., Almomani A., and et al., "IoT Transaction Processing Through Cooperative Concurrency Control on Fog-Cloud Computing Environment," *Soft Computing*, vol.

24, pp. 5695-5711, 2020. https://doi.org/10.1007/s00500-019-04220-y

[4] Al-Qerem A., Ali A., Nabot A., Jebreen I., Alauthman M., Almomani A., Chamola V., and Aldweesh A., "Balancing Consistency and Performance in Edge-Cloud Transaction Management," *Computers in Human Behavior*, vol. 167, pp. 108601, 2025. https://doi.org/10.1016/j.chb.2025.108601

[5] Al-Qerem A., Ali A., Nashwan S., Alauthman M., Hamarsheh A., Nabot A., and Jibreen I., "Transactional Services for Concurrent Mobile Agents over Edge/Cloud Computing-Assisted Social Internet of Things," *ACM Journal of Data and Information Quality*, vol. 15, no. 3, pp. 1-20, 2023. https://doi.org/10.1145/3603714

[6] Alsurdeh R., Calheiros R., Matawie K., and Javadi B., "Hybrid Workflow Scheduling on Edge Cloud Computing Systems," *IEEE Access*, vol. 9, pp. 134783-134799, 2021. DOI:10.1109/ACCESS.2021.3116716

[7] Al-Talafheh K., Aplop F., Al-Yousef A., Obiedat M., and Khazaaleh M., "Predictive Big Data Analytics Capability Model to Enhancing Healthcare Organization Performance," *International Journal of Advances in Soft Computing and its Applications*, vol. 16, no. 3, 2024. DOI: 10.15849/IJASCA.241130.09

[8] Barnaghi P., Tonjes R., Holler J., Hauswirth M., Sheth A., and Anantharam P., "Real Time IoT Stream Processing and Large-Scale Data Analytics for Smart City Applications," *in Proceedings of the European Conference on Networks and Communications*, Bologna, pp. 1-5, 2014.

[9] Bernstein P. and Goodman N., "Concurrency Control in Distributed Database Systems," *ACM Computing Surveys*, vol. 13, no. 2, pp. 185-221, 1981. https://doi.org/10.1145/356842.356846

[10] Boiko O., Komin A., Malekian R., and Davidsson P., "Edge-Cloud Architectures for Hybrid Energy Management Systems: A Comprehensive Review," *IEEE Sensors Journal*, vol. 24, no. 10, pp. 15748-15772, 2024. DOI:10.1109/JSEN.2024.3382390

[11] Celesti A., Fazio M., Galletta A., Carnevale L., Wan J., and Villari M., "An Approach for the Secure Management of Hybrid Cloud-Edge Environments," *Future Generation Computer Systems*, vol. 90, pp. 1-19, 2019. https://doi.org/10.1016/j.future.2018.06.043

[12] Chaudhry N. and Yousaf M., "Concurrency Control for Real-Time and Mobile Transactions: Historical View, Challenges, and Evolution of Practices," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 3, pp. e6549, 2020. https://onlinelibrary.wiley.com/doi/10.1002/cpe.6549

[13] Dragojevic A., Narayanan D., Hodson O., and Castro M., "FaRM: Fast Remote Memory," *in Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, Seattle, pp. 401-414, 2014. https://dl.acm.org/doi/10.5555/2616448.2616486

[14] Duan Q., Wang S., and Ansari N., "Convergence of Networking and Cloud/Edge Computing: Status, Challenges, and Opportunities," *IEEE Network*, vol. 34, no. 6, pp. 148-155, 2020. DOI:10.1109/MNET.011.2000089

[15] Ferreira L., Coelho F., and Pereira J., "Databases in Edge and Fog Environments: A Survey," *ACM Computing Surveys*, vol. 56, no. 11, pp. 1-40, 2024. https://doi.org/10.1145/3666001

[16] Gao X., He P., Zhou Y., and Qin X., "A Smart Healthcare System for Remote Areas Based on the Edge-Cloud Continuum," *Electronics*, vol. 13, no. 21, pp. 1-18, 2024. https://doi.org/10.3390/electronics13214152

[17] Gomes V., Kleppmann M., Mulligan D., and Beresford A., "Verifying Strong Eventual Consistency in Distributed Systems," *in Proceedings of the ACM on Programming Languages*, Barcelona, pp. 1-28, 2017. https://doi.org/10.1145/3133933

[18] Gray J., *Operating Systems Review: An Advanced Course*, Springer, 1978. https://doi.org/10.1007/3-540-08755-9_9

[19] Kim T. and Lim J., "An Edge Cloud-Based Body Data Sensing Architecture for Artificial Intelligence Computation," *International Journal of Distributed Sensor Networks*, vol. 15, no. 4, pp. 1-12, 2019. DOI:10.1177/1550147719839014

[20] Lam K., Lee V., Hung S., and Kao B., "Impact of Priority Assignment on Optimistic Concurrency Control in Distributed Real-Time Databases," *in Proceedings of 3rd International Workshop on Real-Time Computing Systems and Applications*, Seoul, pp. 128-135, 1996. DOI:10.1109/RTCSA.1996.554969

[21] Li Q., Guo M., Peng Z., Cui D., and He J., "Edge-Cloud Collaborative Computation Offloading for Mixed Traffic," *IEEE Systems Journal*, vol. 17, no. 3, pp. 5023-5034, 2023. DOI:10.1109/JSYST.2023.3277003

[22] Maheshwari S., Netalkar P., and Raychaudhuri D., "DISCO: Distributed Control Plane Architecture for Resource Sharing in Heterogeneous Mobile Edge Cloud Scenarios," *in Proceedings of the IEEE 40th International Conference on Distributed Computing Systems*, Singapore, pp. 519-529, 2020. DOI:10.1109/ICDCS47774.2020.00095

[23] Masadeh R., Sharieh A., Abu-Jazoh M., Alshqurat K., Masadeh S., and Alsharman N., "Independent Task Scheduling in Cloud Computing

Environment using Modified Orca Optimizer," *International Journal of Advances in Soft Computing and its Applications*, vol. 16, no. 2, 2024. DOI: 10.15849/IJASCA.240730.01

[24] Mughaid A., Obeidat I., Abualigah L., and et al., "Intelligent Cybersecurity Approach for Data Protection in Cloud Computing Based Internet of Things," *International Journal of Information Security*, vol. 23, pp. 2123-2137, 2024. DOI: 10.1007/s10207-024-00832-0

[25] Rahimi H., Picaud Y., Singh K., Madhusudan G., Costanzo S., and Boissier O., "Design and Simulation of a Hybrid Architecture for Edge Computing in 5G and Beyond," *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1213-1224, 2021. DOI:10.1109/TC.2021.3066579

[26] Ramyasree B. and Naveen P., "A Survey on Edge Computing Mechanisms to Improve Transactional Data in Manufacturing System," *Journal of Engineering Sciences*, vol. 14, no. 1, pp. 557-564, 2023. https://jespublication.com/upload/2023-V14I168.pdf

[27] Sutar S., Byranahallieraiah M., and Shivashankaraiah K., "Objective Approach for Allocation of Virtual Machine with improved Job Scheduling in Cloud Computing," *The International Arab Journal of Information Technology*, vol. 21, no. 1, pp. 46-56, 2024. DOI: 10.34028/iajit/21/1/4

[28] Thomson A., Diamond T., and Ren K., "Calvin: Fast distributed Transactions for Partitioned Database Systems," *in Proceedings of the ACM SIGMOD International Conference on Management of Data*, Arizona, pp. 1-12, 2012. https://doi.org/10.1145/2213836.2213838

[29] Tran-Dang H. and Kim D., "FRATO: Fog Resource Based Adaptive Task Offloading for Delay-Minimizing IoT Service Provisioning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 10, pp. 2491-2508, 2021. DOI:10.1109/TPDS.2021.3067654

[30] Veeramachaneni V., "Edge Computing: Architecture, Applications, and Future Challenges in a Decentralized Era," *Recent Trends in Computer Graphics and Multimedia Technology*, vol. 7, no. 1, pp. 8-23, 2025. https://doi.org/10.5281/zenodo.14166793

[31] Wang C., Bi Z., and Xu L., "IoT and Cloud Computing in Automation of Assembly Modeling Systems," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1426-1434, 2014. DOI:10.1109/TII.2014.2300346

[32] Wang T., Liang Y., Shen X., Zheng X., Mahmood A., and Sheng Q., "Edge Computing and Sensor-Cloud: Overview, Solutions, and Directions," *ACM Computing Survey*, vol. 55, no. 13, pp. 1-37, 2023. https://doi.org/10.1145/3582270

[33] Xiang X., Cao J., and Fan W., "Secure Authentication and Trust Management Scheme for Edge AI-Enabled Cyber-Physical Systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 3, pp. 3237-3249, 2025. DOI:10.1109/TITS.2025.3529691

Abdelraouf Ishtaiwi is a highly accomplished academic with over 22 years of experience in teaching and research in the field of Artificial Intelligence (AI). He earned his Master's degree in AI from Griffith University, Brisbane in 2001, followed by a Ph.D. in the same field in 2007. Dr. Ishtaiwi's academic career has been dedicated to advancing the field of AI through exceptional research and teaching skills. His expertise in the field has led to numerous top scientific contributions, including groundbreaking research on machine learning, local search algorithms, and optimization methods. Throughout his career, Dr. Ishtaiwi has published many research papers in highly regarded academic journals, demonstrating his significant impact on the field of AI. His research has been widely cited and has received recognition from the academic community for its innovative approach to AI. In addition to his research accomplishments, Dr. Ishtaiwi is an experienced teacher and mentor. He has taught a wide range of courses in AI, including advanced topics in machine learning and optimization. His dedication to teaching has earned him accolades from his students and colleagues alike.

Ahmad Nabot is an Assistant Professor at the Faculty of Science and Information Technology, Department of Software Engineering, Al-Zaytoonah University of Jordan, Amman, Jordan. He holds a PhD in Computer Science with a specialization in Software Engineering. Prior to joining Al-Zaytoonah University, he served as a faculty member at Zarqa University, where he contributed to academic programs in computer science and engaged in collaborative research initiatives. His research interests include software development, decision-making, machine learning, and software component reuse. He has published in these areas and actively works on integrating intelligent techniques into software engineering practices.

Omar Alzoubi is an Assistant Professor of Computer Engineering at the University of Umm Al-Qura in Saudi Arabia. With a focus on Computer Engineering, Dr. Alzoubi brings a wealth of expertise to his role. He is committed to advancing the field through both his research and teaching endeavors. As a respected academic, Dr. Alzoubi is dedicated to nurturing the next generation of computer engineers, guiding students to excel in their studies and research pursuits. His contributions to the university community are invaluable as he works towards furthering knowledge and innovation in the field of Computer Engineering.

Awad Ramadan is a dedicated academician with a strong commitment to excellence in education and administration. He holds the position of Lecturer in the Computer Science Department at the College of Computing in AlQunfudah, Umm Al-Qura University, Saudi Arabia, a role he has served in since 2007. Additionally, he plays a vital role as a member of the Academic Oversight Committee, ensuring quality standards in education since 2021. With a wealth of experience and a proactive approach, Awad Mohamed Ramadan continues to make significant contributions to the academic community at Umm Al-Qura University.

Abdulbasit Darem an Associate Professor of Cybersecurity at Northern Border University (NBU) in Saudi Arabia, is a prominent researcher with extensive contributions to the fields of cybersecurity. Holding a Ph.D. in Computer Science from Mysore University, India, Dr. Darem has led numerous research initiatives, including serving as the Principal Investigator (PI) for multiple funded projects, such as those addressing AI, cybersecurity threats, ransomware detection, and privacy in social networks. He has published over 70 articles in international refereed journals with high impact factors and presented several conference papers, tackling critical issues using AI (machine learning and Deep learning). As the Chair of the Cybersecurity Research Group (RG-NBU-2022-1724) and RDO-1385 project, Dr. Darem actively conducts innovative research and collaborates with international teams to address emerging challenges in cybersecurity. His work has earned him recognition and multiple awards for academic excellence like the Excellence Award in Scientific Research 2024 in Northern Border University, highlighting his dedication to advancing cybersecurity knowledge and solutions through impactful research.

Asma Alhashmi an Associate Professor at the Computer Science Department, Northern Border University (NBU), Saudi Arabia, is an accomplished researcher specializing in cybersecurity, software engineering, and web engineering. With a Ph.D. in Computer Science from Mysore University, India, she has contributed extensively to advancing cybersecurity and related fields through her research. Dr. Alhashmi has led multiple high-impact projects, including serving as Principal Investigator (PI) for research on ransomware early detection and phishing countermeasures, both funded by NBU. She has co-led projects on cybersecurity threats, privacy in social networks, and smart home security systems, showcasing her interdisciplinary approach to addressing pressing technological challenges. Dr. Alhashmi has published over 50 articles in internationally referred journals and conferences, covering areas such as deep learning-based malware detection, fraud detection in financial systems, and cybersecurity in cloud-assisted IoT environments. Her work has been widely cited, reflecting its impact on the research community. She also actively collaborates with international teams, contributing to projects on digital forgery detection and behavioral malware detection. In addition to her research contributions, Dr. Alhashmi serves as a reviewer for high-impact journals, including Springer's Cluster Computing and Elsevier's Computers and Security. Her research projects and publications underscore her commitment to advancing cybersecurity knowledge and developing practical solutions for real-world problems.

Mohammad Alauthman Received PhD degree from Northumbria University Newcastle, UK in 2016. He received a B.Sc. degree in Computer Science from Hashemite University, Jordan, in 2002, and received M.Sc. degrees in Computer Science from Amman Arab University, Jordan, in 2004. Currently, he is Assistant Professor and senior lecturer at Department of Information Security, Petra University, Jordan. His main research areas cybersecurity, Cyber Forensics, advanced machine learning and data science applications.

Amjad Aldweesh is a computer assistant professor interested in the Blockchain and Smart contracts technology as well as cyber security. Amjad has a Bachelor degree in computer science. He has a MSc degree in advanced computer science and security from the University of Manchester with distinction. Amjad is the second in the UK and the first in the middle-east to have a PhD in the Blockchain and Smart contracts technology from Newcastle University.