Transactional Services for Concurrent Mobile Agents over Edge/Cloud Computing-Assisted Social Internet of Things

AHMAD AL-QEREM, Department of Computer Science, Zarqa University, Zarqa, Jordan
ALI MOHD ALI, Communications and Computer Engineering Department, Faculty of Engineering, Al-Ahliyya Amman University, Amma, Jordan
SHADI NASHWAN, College of Computer and Information Sciences, Jouf University, Aljouf, Saudi Arabia
MOHAMMAD ALAUTHMAN, Department of Information Security, University of Petra, Amman, Jordan
ALA HAMARSHEH, Computer Systems Engineering, Faculty of Engineering, Arab American University, Jenin, Palestine
AHMAD NABOT and ISSAM JIBREEN, Department of Software Engineering, Zarqa University, Zarqa, Jordan

The Web of Things (WoT) is a concept that aims to create a network of intelligent devices capable of remote monitoring, service provisioning, and control. Virtual and Physical Internet of Things (IoT) gateways facilitate communication, processing, and storage among social nodes that form the social Web of Things (SWoT). Peripheral IoT services commonly use device data. However, due to the limited bandwidth and processing power of edge devices in the IoT, they must dynamically alter the quality of service provided to their connected clients to meet each user's needs while also meeting the service quality requirements of other devices that may access the same data. Consequently, deciding which transactions get access to which Internet of Things data is a scheduling problem.

Edge-cloud computing requires transaction management because several Internet of Things transactions may access shared data simultaneously. However, cloud transaction management methods cannot be employed in edge-cloud computing settings. Transaction management models must be consistent and consider ACIDity of transactions, especially consistency. This study compares three implementation strategies, Edge Host Strategy (EHS), Cloud Host Strategy (CHS), and Hybrid BHS (BHS), which execute all IoT transactions on the Edge host, the cloud, and both hosts, respectively. These transactions affect the Edge hosts as well. An IoTT framework is provided, viewing an Internet of Things transaction as a collection of fundamental

Authors' addresses: A. al-Qerem, Department of Computer Science, Zarqa University, Damascus Street, Zarqa, 13110, Jordan; e-mail: ahmad_qerm@zu.edu.jo; A. M. Ali, Communications and Computer Engineering Department, Faculty of Engineering, Al-Ahliyya Amman University, Amman 19328, Jordan; e-mail: a.mahmoud@ammanu.edu.jo; S. Nashwan, College of Computer and Information Sciences, Jouf University, Aljouf, Saudi Arabia and Cybersecurity Department, Faculty of Information Technology, Middle East University (MEU), Queen Alia Airport Street, Amman 11831 Jordan, e-mails: shadi_nashwan@ju.edu.sa, snashwan@meu.edu.jo; M. Alauthman, Department of Information Security, University of Petra, Airport Road. P.O Box 961343, Amman, Jordan; e-mail: mohammad.alauthman@uop.edu.jo; A. Hamarsheh, Computer Systems Engineering, Faculty of Engineering, Arab American University, Jenin P.O Box 240, Palestine; e-mail: ala.hamarsheh@aaup.edu; A. Nabot and I. Jibreen, Department of Software Engineering, Zarqa University, Damascus Street, Zarqa, 13110, Jordan; e-mails: {anabot@zu.edu.jo, ijebreen}@zu.edu.jo.

1936-1955/2023/09-ART36 $15.00
https://doi.org/10.1145/3603714

and additional subtransactions that loosen atomicity. Execution strategy controls essential and additional subtransactions.

The integration of edge and cloud computing demonstrates that the execution approach significantly affects system performance. EHS and CHS can waste wireless bandwidth, while BHS outperforms CHS and EHS in many scenarios. These solutions enable edge transactions to complete without restarting due to outdated IoT data or other edge or cloud transactions. The properties of these approaches have been detailed, showing that they often outperform concurrent protocols and can improve edge-cloud computing.

CCS Concepts: • **Information systems → Data management systems**;

Additional Key Words and Phrases: Concurrency control, transactional services, mobile agents, edge-cloud computing, execution framework

ACM Reference format:
Ahmad Al-Qerem, Ali Mohd Ali, Shadi Nashwan, Mohammad Alauthman, Ala Hamarsheh, Ahmad Nabot, and Issam Jibreen. 2023. Transactional Services for Concurrent Mobile Agents over Edge/Cloud Computing-Assisted Social Internet of Things. *ACM J. Data Inform. Quality* 15, 3, Article 36 (September 2023), 20 pages.
https://doi.org/10.1145/3603714

1 INTRODUCTION

As the number of devices connected to the **Internet of Things (IoT)** grows, the need for computing at the network's edge will increase. Analyzing data closer to its source has many benefits, including lessening the load on the internet, boosting privacy and security, and decreasing the overall cost of data management.

When it comes to gathering massive amounts of data and performing analysis on that mountain of information to extract insights that can be transmitted back to the devices closer to the edge, the cloud will continue to play a crucial role. By integrating edge and cloud computing, you can better manage and analyze your data, which can significantly increase the value of your Internet of Things systems. Even if decentralized edge networks offer certain advantages in terms of speed and scalability, the cloud often outperforms them in terms of cost [1].

The cloud will be responsible for delivering strategic and high-level insights to improve the operation of these edge devices, while edge devices may be responsible for undertaking real-time functional analysis [2].

The frontend of an IoT data management system is live and real-time, interacting with the connected IoT items and sensors, while the backend handles the large-scale storage and in-depth analysis of IoT data [3]. This categorization originates from the IoT data lifecycle that has been addressed in the previous sections. Transmission of query requests and results to and from the many sensors and intelligent objects is a crucial part of the frontend of the data management system. The back end is quite storage-intensive due to the necessity of not only performing analysis and additional thorough searches, but also storing massive amounts of data created for later processing.

The storage components may be located in the system's backend, but they are considered "online" because of their regular interaction with the system's frontend and their need for frequent updates. Autonomous edges in the lifecycle might be viewed as placing more value on communication than storage because of the speed with which they react to specific inquiries. Existing DBMS are generally storage-centric, while the data management architecture envisioned in Figure 1 is more concerned with data access and manipulation. Most information in traditional database management systems comes from static and constrained sources. After proper normalization and scalarization, this data is relationally stored. Queries are used to obtain customized "summary" views of the system or to modify individual entries inside the database. Whenever new information or data

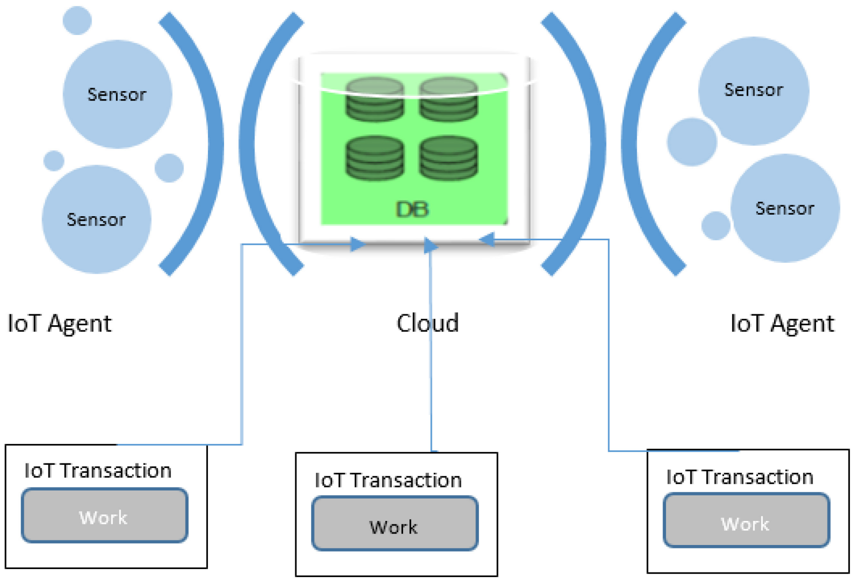


Fig. 1. Data-management-architecture-for IoT transactional services.

needs to be added to the database, insertion queries are used to do so. Typically, query operations are executed locally, with only the costs of processing and interim storage being incurred.

Data integrity is preserved while using a transaction management strategy that ensures the ACID characteristics are met. If the database is split up among multiple servers, we will still run the queries and manage the dispersed transactions. A key tenet of database design is the "transparency principle," which requires all data in the database to be accessible and visible, and this forms the basis for the execution of distributed queries. A two-phase commit process is required in order to ensure that the ACID characteristics are met. Under this tenet, dispersed queries must be carried out adhering to the notion of transparency. Countless data sources, which are only growing in quantity, completely alter the picture in IoT systems. Sensors, RFIDs, embedded systems, and mobile devices are all examples of such data collectors. Data stored in IoT data stores is constantly streaming from a wide variety of "Things," and the needs of the queries are both more frequent and more varied than those submitted to traditional DBMSs. In order to assure scalability and provide more immediate processing functionality, it may be necessary to report and aggregate data in a hierarchical fashion. Unstructured and flexible forms, which can accept a wide range of data types and complex queries, can replace the inflexible relational database design and relational normalization technique. Optimizers still base their decisions on well-defined and rigid schemas, even if distributed DBMSs optimize queries according to communication considerations. However, this might not hold true in the Internet of Things, where the constant influx of new data sources and the importance of contextual, real-time information make for a challenging environment for query optimizers. Distributed database management systems make it challenging, if not impossible, to guarantee the transparency criteria that are imposed on IoT **data management systems (DBMSs)**. Furthermore, it is possible that with the IoT, transparency is not essential because unique applications and services can necessitate knowledge of both location and context. There are also obstacles for more globalized areas, even if it is possible to control the process of processing transactions while still keeping ACID properties in contained IoT spaces (subsystems). In spite of this, IoT data management systems have not yet addressed the new challenge presented by IoT data sources [4, 5]. The difficulty here is in figuring out how to incorporate the information that is produced by IoT data sources into the established data repository.

The most significant findings and conclusions of this paper are summarized as follows:

- We have proposed a framework for the execution of IoTT processing and are researching the impact of mobility on the repercussions of the different execution strategies for IoTT processing.
- We've built a model to represent the transactional system, and it includes support for updates and the Internet of Things. The model provides a solid foundation for studying how different concurrency control techniques fare in real-world scenarios.

This paper's subsequent sections will be provided as follows. In the first section, we survey prior research on transactional services enabled by edge-cloud computing infrastructure. The transaction processing environment of the Internet of Things is explored in Section 3. Section 4 explains the infrastructure required to support transactional services at the edge of cloud computing. In Section 5, we make use of the fact that the vast majority of IoTT models share a relaxation of the ACID properties. This allows us to experiment with various execution strategies to model the impact of mobility and disconnections on the processing of IoT transactions, and then debate the results. This section provides a high-level overview of our work and a description of the scientific contributions made.

2 RELATED WORKS

In recent years, the **Internet of Things (IoT)** has permeated virtually every aspect of modern life, from cities and homes to colleges and businesses to farms and hospitals [6–8]. Numerous capabilities, including data production and consumption and the use of internet services, enhance daily life and activities all around the world through the IoT environment [9]. Numerous applications are executed in the IoT ecosystem, and it is through these programs that the infrastructure and smart services are delivered [10]. As the needs of the users increase, the distribution of novel apps for monitoring, managing, and automating human activities will increase [11, 12]. In addition, IoT applications leverage cloud computing to provide necessary composite services by composing existing atomic services to back up service-based apps [9, 10]. Users in a wide range of fields utilize apps that leverage smart devices in their daily lives, and these apps are increasingly being adapted for usage in IoT contexts. There are many advantages to using IoT applications, such as improved decision-making, management, and monitoring of cloud-based resources in the environment [13].

Although different application domains are motivated by different factors, they all share the common goal of providing smart services to enhance the quality of human life [14, 15].

It's important to remember that mobile computing has its own set of constraints, such as sporadic connectivity, short battery life, low bandwidth communication, and limited storage space, while making a purchase. Reducing the likelihood of network outages is crucial for mobile computing. Transactions done on the central and peripheral hosts must be consistent, and this can only be achieved by operations on shared data. If we want to optimize concurrency and minimize communication costs, we need to make sure that the transaction execution on the stationary or edge hosts blocks as little as feasible. Authenticating IoT Hardware The transactions must be designed to be locally autonomous on the edge host so that they can be completed and committed even if the network connection is temporarily lost. Mobile computing [16] uses semantic-based transaction processing models [17] that rely on commutative activities to boost concurrency. These strategies necessitate either extensive database caching or the maintenance of many copies of certain data objects. Semantic-based transaction processing in mobile databases is made easier by the ability to partition data objects into smaller parts [18]. The method breaks down the data into manageable pieces. The inability to share information between silos means that data fragments must have their own independent caching systems and frequent upgrades. In response to the MU's request, we send across a chunk of the data object. After the transaction is finalized, the peripheral hosts will send the information back to the hub. A reorderable item is one that can have its constituent components assembled in any order. The fragmentation of the system is particularly useful for certain kinds of data objects, such as sets, stacks, and queues. To facilitate the restoration procedure, transaction proxies are included in [19]. In addition to the principal transaction, each MU also sends a proxy transaction to the base station at the same time. If you make a change to the original transaction, the proxy will be updated immediately. From time to time, a proxy transaction will support the work of an edge host's computational nodes.

Definition: "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [20]. The falling cost and rising availability of storage and processing power have made feasible a new computing model in which virtualized resources can be rented on an as-needed basis and offered as general utilities. Companies like Amazon, Google, Facebook, and the like, have migrated to this approach of offering services via the internet because of the financial and technical benefits [21]. Applications at the network's edge today ingest data generated by things at a scale and variety never before seen in human history. However, there are certain types of applications that may not work well with cloud-based service models due to factors such as the necessity for extremely fast reaction times, the need to access sensitive local data, or the sheer volume of data they produce. Edge computing could be used instead in those applications.

As a new paradigm, edge computing (also called "fog computing") involves placing large amounts of computing and storage resources near the periphery of the Internet, in close proximity to endpoints such as mobile devices, sensors, actuators, and machines. Edge computing is performed at or near the points of origination of data [22], or where choices or interactions with the physical environment are being made or experienced with little latency. It can handle information coming from the cloud as well as information coming from endpoints and other parts of the network. In most contexts, the phrase "fog computing" is used to refer to the concept of "multi-tiered edge computing," which refers to the presence of multiple tiers of compute infrastructure between end devices and cloud services.

To put it simply, an edge device is any type of computing or networking resource located on the periphery of a network, between the end devices themselves and the central data storage facilities, such as the cloud. Computing at the edge involves two-way data flow, with the end device serving as both a consumer and a producer. The devices at the network's edge not only make requests for services and data from the Cloud, but also perform computer functions such as processing, storing, caching, and load balancing on data that is transmitted to the Cloud [23]. In other words, this doesn't rule out the possibility of end devices hosting computation on their own, either on their own or as part of a distributed edge computing platform.

The literature outlines several distinct models for **Internet of Things Transactions (IoTT)**, including the centralized model, decentralized model, hybrid model, blockchain model, peer-to-peer model, and edge computing model. In addition to the existing columns for model name, architecture, communication protocol, this table has added two more columns for application and advantages. The *centralized* model is characterized by a centralized architecture and supports communication protocols such as HTTP, MQTT, and CoAP. This model is commonly used in smart homes and industrial automation due to its ease of deployment and cost-effectiveness. The *decentralized* model is characterized by a decentralized architecture and supports communication

Table 1. Taxonomy for Internet of Things Transactions (IoTT) Models

Model Name	Architecture	Supported Devices	IoTT Transactions Model	Advantages	Applications
Centralized	Centralized server	Low-end devices	Publish/Subscribe	– Easy to manage and control Centralized processing and storage	Smart home automation, Industrial monitoring, Healthcare
Distributed	Peer-to-peer network	Low-end to high-end devices	Peer-to-peer	– Low latency Distributed processing and storage	Smart grid, Smart traffic control, Military applications
Hybrid	Combination of centralized and distributed	Low-end to high-end devices	Publish/Subscribe and Peer-to-peer	– Scalable – Fault tolerant – Efficient resource utilization	Smart city, Agriculture, Logistics
Fog Computing	Cloud and edge computing	Low-end to high-end devices	Hybrid (Publish/Subscribe and Peer-to-peer)	– Low latency – Energy efficient – Decentralized processing and storage	Autonomous vehicles, Smart healthcare, Industrial Internet of Things
Blockchain-based	Distributed blockchain network	High-end devices	Distributed and secure transactions	– Transparency and trust – Immutable and tamper-proof – Decentralized processing and storage	Supply chain management, Financial services, Asset tracking

protocols such as HTTP, MQTT, and CoAP. This model is commonly used in healthcare and energy management applications due to its low latency and high scalability. The *hybrid* model is a combination of centralized and decentralized architectures, and supports communication protocols such as HTTP, MQTT, and CoAP. This model is commonly used in agriculture and smart cities due to its flexibility in deployment and ability to offer benefits of both centralized and decentralized models. The blockchain model is characterized by a decentralized architecture with blockchain technology and supports communication protocols such as MQTT and CoAP. This model is commonly used in supply chain management and financial transactions due to its high level of security and transparency. The peer-to-peer model is characterized by a decentralized peer-to-peer architecture and supports communication protocols such as HTTP, MQTT, and CoAP. This model is commonly used in personal devices and wearables due to its privacy features and reduced latency.

The edge computing model is characterized by a decentralized architecture with edge computing and supports communication protocols such as HTTP, MQTT, and CoAP. This model is commonly used in real-time monitoring and video surveillance applications due to its ability to reduce network traffic and latency. Table 1 provides a comprehensive overview of several distinct models for IoTT transactions. By highlighting the different architectures, communication protocols, applications, and advantages of each model, this table can help organizations make informed decisions when selecting a model for their IoTT transactions.

3 TRANSACTIONAL CONTEXT IN IoT

Connectivity, networking, and communication protocols used by these web-enabled devices are all determined by the specific IoT applications being used.

One of the main goals of IoT applications is to improve QoS metrics. Intelligent services in IoT applications should fulfill the requirements of the user. The aforementioned quality-of-service parameters (such as security, cost, service time, energy usage, reliability, and availability) should be addressed by these offerings.

The expanding edge computing capabilities made available by the Internet of Things (IoT) and the impending broad deployment of 5G cellular technology necessitates increased dispersal of platform components to achieve higher degrees of scalability.

With 5G networks and the proliferation of IoT devices, hybrid edge-cloud solutions provide increased benefits for data collection and processing.

Edge devices (those placed near the point of data collection or access) and the cloud work together in this architecture to handle data (which can scale to meet fluctuating demand). Although edge-cloud architectures have several advantages, data management presents a number of challenges. The fact that they are used on such a vast scale and comprise several hardware and software components presents the majority of the problems.

When replicating stateful (edge or otherwise) components for scalability, synchronization is expensive and complicates fault tolerance; when replicating large amounts of data between the edge and the cloud, concerns about network latency and storage capacity arise; in the context of the Internet of Things, in particular, the heterogeneity of edge devices presents a very wide collection of data models, necessitating data processing frameworks to handle each.

Transaction processing in cloud edge computing assisted **Social Web of things (SWoT)** has the potential to revolutionize various industries by enabling seamless communication and coordination among multiple devices and systems. One practical application of transaction processing in SWoT is smart home automation, where several devices such as thermostats, lighting systems, and security systems need to interact with each other and share data. With transaction processing, multiple devices can coordinate and manage transactions to ensure that data is consistent and accurate across all devices. This ensures a seamless user experience, where the user can easily control and monitor their smart home devices.

Another practical application of transaction processing in SWoT is industrial automation, where multiple sensors and machines need to interact with each other and share data. In an industrial automation environment, using transaction processing, multiple sensors and machines can coordinate and manage transactions to ensure that data is consistent and accurate across all devices. This is critical for safety and efficiency, as it enables real-time monitoring of production processes, reducing downtime and increasing productivity.

In the healthcare industry, transaction processing in SWoT can be used for healthcare monitoring. In a healthcare monitoring system, there may be multiple sensors and devices that collect data from patients and send it to a central database. With transaction processing, multiple sensors can coordinate and manage transactions to ensure that data is consistent and accurate across all devices and databases, which is critical for patient safety and effective treatment. This enables healthcare providers to have access to real-time data, allowing them to provide timely and effective care to patients.

Transportation management is another industry where transaction processing in SWoT can be used. In a transportation management system, multiple devices such as GPS trackers, cameras, and sensors can be used to collect real-time data on vehicle location, speed, and fuel consumption. Using transaction processing, multiple devices can simultaneously access and manipulate shared data such as route optimization and fuel efficiency. This ensures that transportation companies can optimize their operations and reduce costs while providing timely and efficient services to their customers.

However, there are also limitations and challenges to using transaction processing in cloud edge computing assisted SWoT. The use of cloud edge computing resources may introduce additional latency and network congestion, which can impact the performance of transaction processing. The complexity of distributed transaction management techniques may make them difficult to implement and maintain in a SWoT environment. The lack of standardization and interoperability

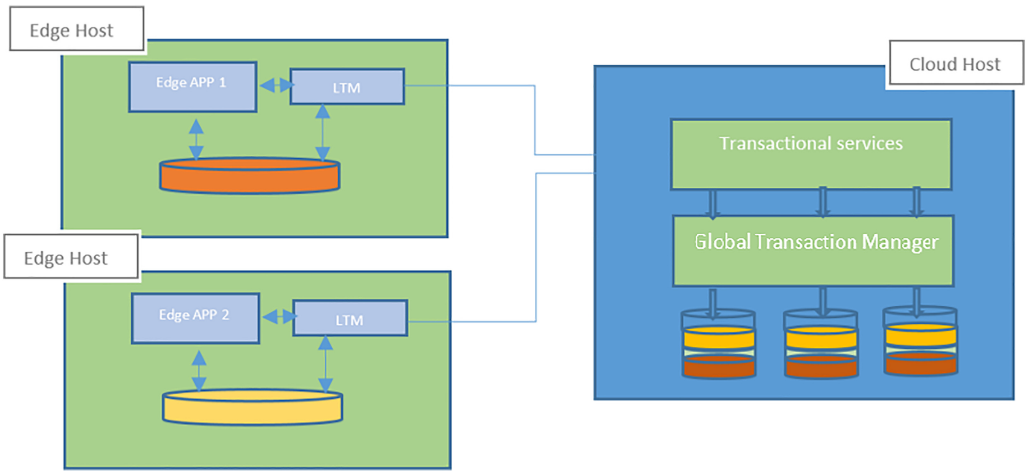


Fig. 2. Edge –Cloud computing architecture.

among IoT devices and platforms may also make it difficult to implement transaction processing across heterogeneous systems in a SWoT environment. It is essential to consider these limitations and challenges when implementing transaction processing in SWoT to ensure successful adoption and usage of these systems.

The Social Web of Things (SWoT) is a new paradigm that combines the Internet of Things (IoT) and social media technologies to enable new social interactions and services. **Concurrent transactional services (CTS)** are a type of transaction processing system that enables concurrent access to shared data without conflicts or inconsistencies. In this context, analyzing social behavior can be beneficial in advancing the use of CTS in SWoT.

Social behavior can be defined as the actions and interactions of individuals within a social group. In SWoT, social behavior can provide insights into user needs, preferences, and expectations, which can be used to design and implement CTS that are more effective and user-friendly. For example, social behavior can be analyzed to identify common patterns of interactions and collaborations between users, which can be used to develop personalized and context-aware CTS.

Moreover, social behavior can also help to address some of the challenges associated with CTS in SWoT, such as security and privacy issues. By understanding user behavior and preferences, CTS can be designed to provide better security and privacy protections that are aligned with user expectations. For example, social behavior analysis can be used to identify the most sensitive data and transactions, which can be protected with stronger security measures.

4 EDGE –CLOUD COMPUTING SYSTEM

The fundamental idea behind an edge cloud computing system is to use the computing approach as close to the data source as is feasible. In Figure 2 we see the edge-cloud computing architecture used to back up transactional services. Edge computing systems use the **local transaction manager (LTM)** to process data, thereby bringing the cloud server to the hardware. The edge computing architecture is divided into the physical layer, the network layer, and the application layer. The physical layer organizes the underlying hardware, such as sensors, robotics, actuators, and so on. In the second tier, you'll find all the edge servers that do the heavy lifting of processing terminal devices to feed into the third layer. An edge server, in contrast to a cloud server, offers a more constrained computational service in terms of both capacity and resources. IoT data processing on cloud servers is where it all begins. The potential benefits of looking at edge computing for transactional services far outweigh the costs of additional cloud server resources. Existing studies on Internet of Things applications [2] mostly agree on this architecture, in which a global database is dispersed among the cloud nodes.

As a collection of data points, the term **"global database" (GDB)** is used to describe a certain type of information repository. In a GDB, data is split between hosts in the cloud and those in the periphery. Where the data items stored on cloud hosts constitute the **fixed database (FDB)** and the data items stored on edge hosts constitute the mobile database of the global database. More so, the MDB's data items stored on a single edge host are referred to as a mobile part of the MDB. A piece of information stored in the FDB is called a "fixed data item," while the same piece of information stored in the edge is called a "mobile data item." There is only ever one authoritative version of any given data item. The primary copy of an item of data can only be modified by its owner. The MDB data item is held by a node on the edge, while the FDB data item is owned by a node in the cloud. We also presumed that each edge host has access to local memory and processing power, and can independently assess and share its mobility status. While the cloud host performs a full replication of the mobile data items, the edge host performs a caching replication of the fixed data items. In addition, information stored on other mobile devices is off-limits in an IoT transaction. In other words, it can only access mobile and fixed data items within its immediate vicinity. However, the system must respond to analytical questions based on edge data. The ability to easily perform ad hoc searches for exploratory data analysis is only one example. For one thing, it's not always easy to anticipate what kind of information will need to be retrieved. Therefore, the system needs to be adaptable enough to deal with different workloads by finding a happy medium between network throughput and processing speed at the edge. In an environment with a large number of IoT devices, network and CPU resources on the edge are sometimes constrained due to cost and energy considerations, making it difficult to send all collected data to the cloud. Furthermore, it causes lags in the most critical information at the moment. Finally, the heterogeneity of the devices and the collected data makes it challenging for the cloud to have a consistent and holistic view of data, adding to the complexity of analytical workloads.

The idea behind the edge computing paradigm is to use edge devices for their processing and storage capabilities while relying on cloud services for the more demanding calculations that edge devices can't handle. It may become necessary to upload massive amounts of data, created by many different kinds of edge devices, to the cloud. Given this, we propose a global transaction manager capable of handling IoT transactional services as part of a distributed data management architecture for an edge-to-cloud computing environment, as shown in Figure 1. By coordinating the work of a local and global transaction manager, data transfer from the edge to the cloud may go smoothly. To achieve this, it considers both the data that is now being requested by the cloud and the data that is already present in the cache. After that, it makes up for missed information by balancing network latency and edge node load.

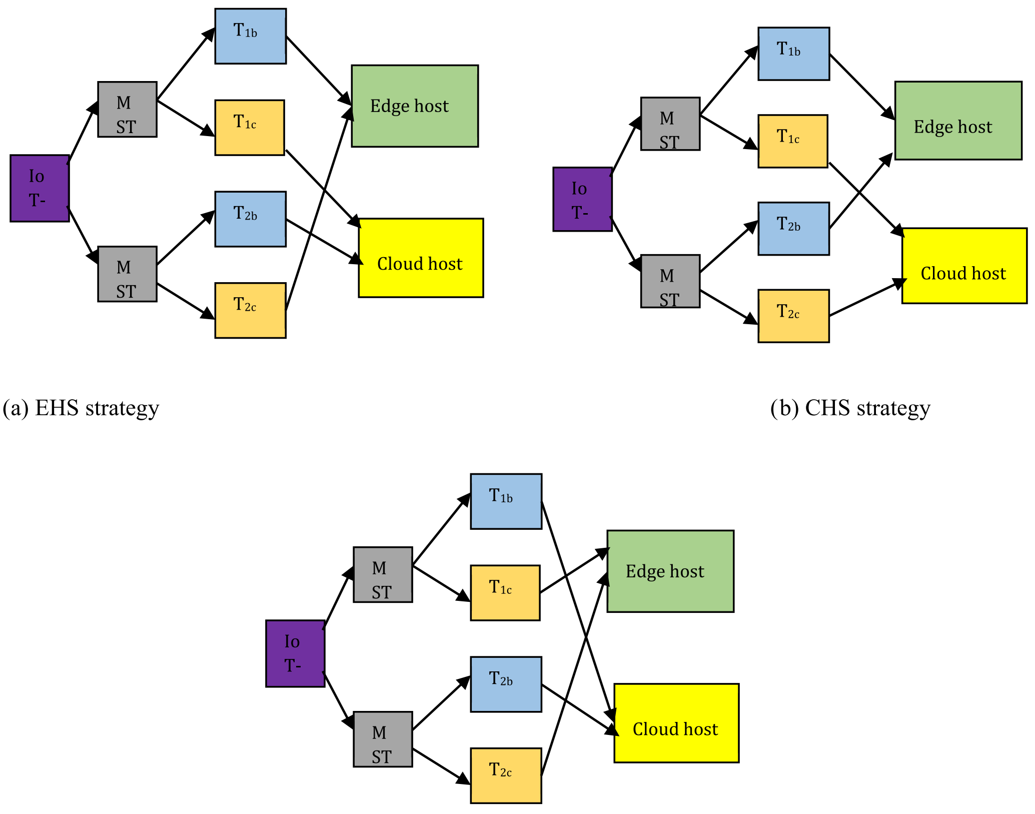
Finally, the transaction manager guarantees consistent reading and writing between the edge and the cloud.

5 IoTT PROCESSING FRAMEWORK

Several distinct models for **Internet of Things Transactions (IoTT)** [6] have been presented. These models expect an IoTT to be broken down into a series of smaller transactions, allowing the atomicity to be loosened. All of these models have the common ground of being an extension of sophisticated transaction models. To that end, rather than analyzing a particular IoTT model, the focus of our study here is on assessing how well various execution strategies operate under varying network connectivity constraints. Our working hypothesis is that an IoTT is a collection of smaller transactions executed on IoT data. In order for a subtransaction's modification at the execution location to take effect, it must be mirrored on the other side of the network. (That is, changes made on the edge host should be mirrored on the cloud host and vice versa if execution occurs at the edge host.) Each IoTT is broken down into Tb, the most fundamental part of the transaction, and Tc, the most fundamental part of the transaction's complement. There are two types of commit points because of this: local commit and global commit. When all of the basic subtransactions have completed processing, the status of each is recorded in a local commit because their complementary subtransactions might not be sent until the network is restored. Any changes made to data during a subtransaction will be reflected in the master copy after a global commit has been performed. Only when the network is up and running can a global commit be performed. A IoTT is considered committed if and only if its foundational and ancillary subtransactions are also committed. There are two phases to processing a mobile subtransaction, with the first phase consisting of a straightforward subtransaction. This is the time when the bulk of the transactional work is done. A supplementary subtransaction is carried out once the data items on the fixed component and the mobile part are linked. This second-stage mobile subtransaction processing is a derivation of the original basic subtransaction. Since a basic subtransaction always executes in the opposite position as its complimentary subtransaction, the latter often updates the same data items as the former. Each component of an Internet of Things transaction is typically scheduled as either a data request or a transaction request. As part of a data request transaction, the most fundamental part is handled by an IoTT host using data that has been previously stored in a fixed component's cache. A cloud server handles the complementing subtransaction that goes with it. On the other hand, the cloud host processes the transaction request's fundamental subtransaction by replicating the edge host's data items. The edge host handles the complimentary subtransaction that corresponds to it. Any changes made to the data will be reflected in the original. If there is inadequate connectivity between the IoTT host and the edge host's local base station, the basic subtransaction will be executed using cached data (i.e., poor bandwidth or disconnected). If some piece of information is missing, the transaction is canceled; otherwise, a companion subtransaction is generated and added to the queue. As soon as the IoTT host is back online, the base station initiates transmission of the queued complementary subtransactions to the cloud host. If a subtransaction that is meant to work with it fails, the mobile subtransaction is canceled; otherwise, it should be waiting for global commitment. If the connection to the network is strong, an IoTT will begin. As can be seen in Figure 3(a-c), the execution strategy determines the position of its fundamental subtransaction.

If a data request is chosen, the cloud server will be queried for the transaction-specific fixed data items (the size of downloaded data is determined by the amount of valid data in the cache of the IoTT host). As a result, processing times for IoT transactions are about the same as they are during periods of low network availability. In contrast, a transaction request will be used if and only if it is selected. The IoTT host, who owns the mobile component, will provide the necessary mobile data items to the cloud host so that the basic subtransaction can be executed. Similarly, if the network connectivity between the IoTT host and its local base station is enhanced over a particular level, the additional subtransaction will be queued and conducted later. If network connectivity is stable once the primary subtransaction is complete, the secondary subtransaction can be issued to update mobile data items on the IoTT host. Pseudo-code for the processing of an Internet of Things transaction is shown in Algorithm 1.

Several factors should be taken into account before deciding whether to use the cloud or the edge. In the same breath, it is not necessary for all IoT device data to be sent to the cloud for processing, which would significantly reduce available network capacity. A portion of this information need not be kept indefinitely. Therefore, as the number of IoT devices grows and people



(a) EHS strategy

(b) CHS strategy

(c) BHS strategy

Fig. 3. Execution framework for the three strategies.

have greater access to resources, edge computing will emerge as a viable alternative to cloud computing.

5.1 Cloud Host Execution Strategy (CHS)

With this method, data from the mobile device that is necessary to complete an Internet of Things transaction is uploaded to a central server in the cloud. The cloud host works with the edge host to coordinate the transaction's execution and then sends the completed results back to the mobile device. Therefore, an IoTT can be executed in a client-server architecture in the same way as a regular distributed transaction would. The only real difference is that, due to the nature of wireless communication, updates may take longer to spread and findings may take longer to return to an edge host. This plan of action is simple to implement. Its primary benefits are (1) strict data consistency, which is maintained because a transaction's entirety is managed and executed on the cloud host, allowing for the use of traditional transaction schemes; (2) reduced battery consumption, which is achieved by shifting the operations to the fixed network, computation tasks on an edge host being avoided; and (3) increased concurrency at the cloud host, which is achieved by avoiding long delays resulting from poor communication. Disconnecting a mobile device from its cloud host prevents it from doing independent tasks since critical information is not available. This tactic is appropriate when all of the available database space is being utilized by the dedicated servers. Simply put, an edge host acts as a client or other distant device. This strategy is universally acknowledged as a cornerstone of the processing paradigm.

ALGORITHM 1: Processing Algorithm for IoTT

```
IoTT Processing(Tb ,Tc)
Begin
For each IoT subtransa
Build ($T_{basic}$ and $T_{complemnetary}$ )
Check availability of (R-Set, W-Set )
If the subtrans T is scheduled as D-req transaction then
    Begin
For all Di belong to (R-Set, W-Set ) of $T_{basic}$,
Edge_Host Invoke a Lock_Req(for all Di belong to T_b)
IOT-TM (IoT agent) Handel all read values.
If Lock_Req(for all Di belong to T_b) =True
Send (Return(R-set(Tbasic)) to Edge Host
Edge Host_Execute( Tb)
CloudHost_Excute($T_{complementary}$ )
    End
Else if the subtrans T is scheduled as T-req trans then
    Begin
For all Di belong to (R-Set, W-Set ) of $T_{basic}$,
Edge_Host Invoke a Excute_Req(for all Oi belong to T_b)
IOT-TM (IoT agent) Handel all the required read values.
If Lock_Req(for all Di belong to T_b) =True
Send (Return(R-set(Tbasic)) to Cloud Host
Edge Host_Execute( Tc)
CloudHost_Excute($T_b$ )
    End
If all succesof (IoT Subtransaction)=True then
State( IoT Trans ) == commit
End
```

5.2 Edge Host Execution Strategy (EHS)

Instead of moving data to a cloud host, a mobile unit can just have a copy of the data kept there. This allows for continuing computing even if the connection drops. As an alternative to creating a complete copy on a local disk, caching copies is a typical method used to facilitate self-sufficient activities, boost availability [24], and reduce the need for network connection. This policy makes it possible to perform transactions without relying on persistent data services, as all data necessary to complete an IoTT is stored locally. Consequently, the mobile unit can do the autonomous tasks. Since the network link between a mobile unit and a cloud host is asymmetric, this policy utilizes less battery power than the CHS method when transmitting data. The transmitter on a cloud host is usually more powerful, and the latter has access to an infinite supply of power. Because of this, it is more likely that data transferred from a cloud host to a mobile unit will take less time than data transferred in the other direction. As far as we know, there are two ways in which information can be transmitted. To begin, let's pretend we're operating under a set of predetermined guidelines. In this method, necessary information is delivered to a mobile device from cloud hosts just before a break in the connection is made. In this configuration, a disconnection protocol must anticipate which data will soon be needed. Second is the improvised mode. When operating in this mode, data is downloaded based on the requirements of active transactions and the availability of the network. For example, the mobile device will get data items X and Y if the transaction requires them and the bandwidth is sufficient at this time. When there are two types of disconnection to

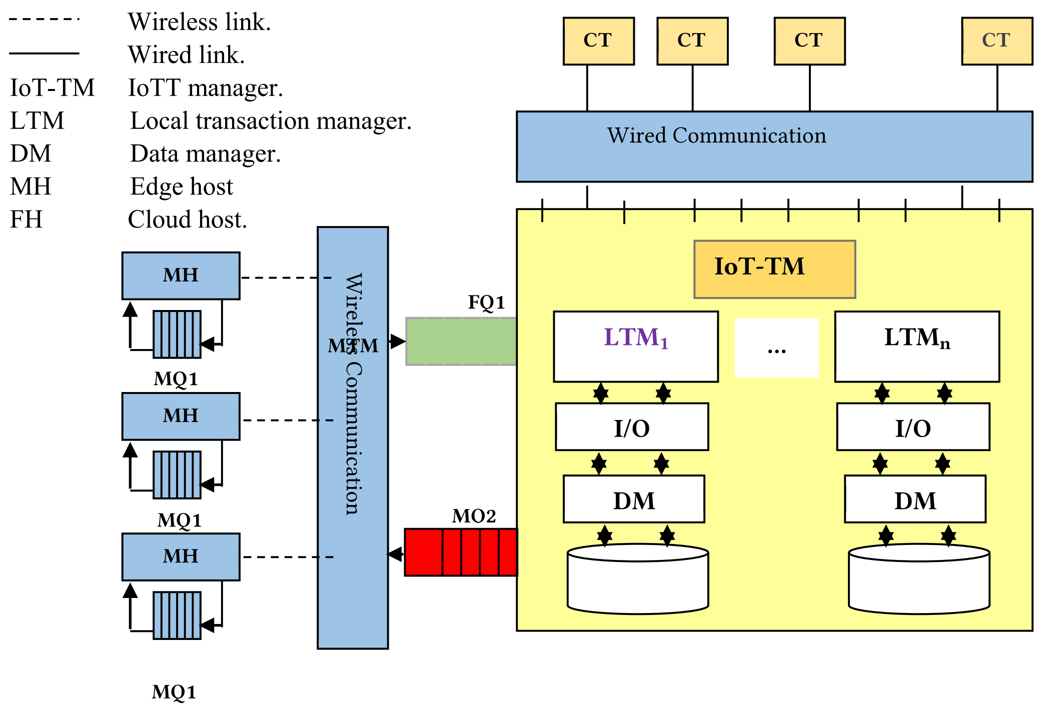


Fig. 4. The simulation model overview.

consider, the first mode is more suited for predictable disconnection, while the second option is better suited for unpredictable disconnection.

5.3 Both Execution Strategy (BHS)

For IoTT s with numerous subtransactions whose execution lifetimes span periods of fluctuating resource availability, neither EHS nor CHS alone can provide improved performance in terms of system throughput and battery consumption. Due to the dynamic nature of the mobile environment, an adaptive strategy to control transaction execution is preferred. Given a transaction request, the appropriate execution strategy for each subtransaction of the same IoTT is decided between cloud host and edge host execution strategies, taking into account the present mobile computing environment. This combined strategy enables the pairing of a mobile subtransaction with a data request or a transaction request. There are two major benefits that we can reap from this approach: The ability to (1) temporarily commit an IoTT using locally cached data and (2) submit other transactions even when the device is disconnected from the network paves the way for autonomous activities. (2) The host used to handle an IoTT is decided on in the moment.

6 PERFORMANCE EVALUATION

With the help of simulation, the efficacy of various execution approaches is analyzed. The purpose of this simulation is to compare the efficiency of three distinct approaches to IoTT processing over a range of network conditions. We detail the system parameters, simulation mode1, and discussion of the outcome.

The Simulation system is abstracted as a queuing model. Its general structure is given in Figure 4. The model relies on three servers—MH, IoT-TM, and FH—to function. **IoTT Manager-Related Services Server (IOT-TM)**. In most cases, this service is prioritized for requests made through the Internet of Things. Depending on the chosen execution method, each IoTT will issue a different set of requests, each of which will represent a different subtransaction. IoTT host services are represented by the MH server, and cloud host transaction coordination services are represented by the FH server. When the primary subtransaction of an Internet of Things transaction completes execution, the secondary subtransaction, if scheduled as a Dreq-transaction, is placed in queue MQ1. A Dreq complimentary subtransaction enters queue FQ1 on a cloud host for processing if the network is connected, else it is placed in queue MQ1 until the network is connected again. If a Dreq complimentary subtransaction is successfully finish its execution, the mobile subtransaction is completed; otherwise, the subtransaction fails. When all sub-transactions have been committed successfully, the whole IoTT will be committed. This is the same procedure followed for a Treq-scheduled subtransaction, albeit with different notations in Figure 4: if the network is connected, a T-req complementary subtransaction is submitted to an edge host for processing; otherwise, it must wait in MQ2 for network re-connection. The BHS tactic is best understood as an amalgamation of the EHS and CHS. Data is downloaded, the basic subtransaction is processed, a supplementary subtransaction is submitted, and finally, the supplementary subtransaction is processed, all as part of the EHS method for handling Internet of Things transactions. So, in total, we have four elementary offerings. Similarly, the CHS model has four simple services as well. The data necessary for an Internet of Things transaction is sent from an edge host to a cloud host in the CHS model, in contrast to the EHS approach. Additionally, a cloud host is used to run the first three elementary services, which are the submission of supplementary subtransactions, processing of basic subtransactions, and the uploading of data. Whereas complimentary subtransaction processing is conducted by the edge host that issues the transaction. When an edge host drops out of the transaction chain, the BHS approach treats the transaction as a Dreq transaction and stops transmitting data. In the event of a successful network connection, the data will be transferred either via upload or download, depending on the schedule in effect. All three tactics are typically simulated independently of one another.

Requested transactions are always scheduled as Treq transactions if the policy is CHS. If a request is in the process of being processed when the network goes down, the request will abort. If the network is not available after the first phase completes, the complimentary subtransaction will have to wait in the queue. The EHS methodology, in contrast to the CHS method, does not routinely ignore a transaction request. Instead, cached data is used to complete the initial stage of transaction processing even if the network is down. If the network fails after this step, the request will be queued until service is restored. When the network is down, the BHS strategy operates similarly to an EHS method, and when it's up and running, it's a hybrid of the CHS and EHS. As an added complication, a supplementary subtransaction may or may not be prevented. Transaction processing speed is critically dependent on the availability of a reliable network connection. To represent this, the time required to complete a Dreq transaction can be broken down into four categories, depending on the availability of the network. One can think of a CC-type transaction as being identical to any other transaction in a fully connected network. When one of the other three transaction types occurs, it usually means that one of two communication processes has failed (denoted by the D). Second, a CD-type transaction shows that the network was online throughout the data transmission but offline for the submission of the complementing subtransaction. If the transaction is of the DC kind, then the network will be down for the data transfer but up during the submission of the complementing subtransaction. If a transaction is of the DD type, it means that the network was down both during the data transmission and the submission of the complimentary subtransaction. These four sorts of deals can each have varying results. The environment is a factor in the calculations that determine how long tasks will take to execute and how long messages will take to send. The two main categories for the system's services are those

Table 2. Parameters Settings

Communication Parameters	Default Values
Arrival of IoTT	5
Arrival of fixed transaction	5
Mobility timer	5,10,15, ..., 50
Disconnection Threshold	1000, 150, 50
UplinkBW	10–1500
DlinkBW	10–3000
Disconnection Period	20–30
Service time Uplink Channel	30
Service time Downlink Channel	10

focused on communication and those focused on processing. Intended for a service that emphasizes open lines of dialogue. The time it takes to provide a service depends on the size of the data being transferred and the available bandwidth, as well as a constant communication overhead. The length of time required to complete a processing-oriented service depends on a variety of parameters, including the following: the number of data items accessible; the average number of operations performed on these data items; and the amount of time spent in the cloud host.

Multiple databases hosted on both central and edge hosts make up the foundation of the cloud database system. The term "transaction-generators" refers to the individuals responsible for creating new transactions within the system. The workload of an IoT system is defined by its average, minimum, and maximum number of subtransactions in a given amount of time. There are a variety of factors that contribute to the total number of transactions, including the average, minimum, and maximum number of database operations in a subtransaction, as well as the likelihood of a write operation. The number of locations and local and IoT Transactions that make up the global burden are generated at random. Many different kinds of deals are made at each individual location. The regional infrastructure does not distinguish between the two varieties (local transaction and mobile subtransaction). When a transaction reaches the execution phase, it is scheduled by first obtaining a lock on the data it will use. If the lock is given, the operation is processed by the CPU and I/O queue, and a signal to abort or commit is sent back to the IoT-TM for a mobile subtransaction, ending the subtransaction. Any local transaction or mobile subtransaction can be cancelled by the local system. An IoTT is terminated and restarted at all sites if a mobile subtransaction is cancelled locally. Edge hosts and IoT-TMs can exchange information in both directions, with uplink channels used for sending information and downlink channels used for receiving it. Uplink and downlink communication overhead is 30 and 10, respectively. Twenty to thirty seconds of inaccessibility to the mobile device is experienced during the disconnection. Several parameters are tweaked between simulation runs to help compare and contrast the various execution strategies.

The simulation of the proposed system for transaction processing in cloud edge computing assisted social web of things (IoT) relies heavily on the parameter settings presented in Table 2, Table 3, and Table 4. The communication parameters listed in Table 2 are particularly critical for ensuring the efficient and reliable transmission of IoT and fixed transactions. The arrival of IoT and fixed transactions, mobility timer, and disconnection threshold all affect the communication between IoT devices and edge hosts. The uplink and downlink bandwidths are also critical parameters that affect the rate of data transmission between IoT devices and the edge hosts. Additionally, the disconnection period and service time parameters determine the maximum amount of time that IoT devices can remain disconnected from the edge hosts without losing connectivity.

Table 3. Cloud Host Parameter

Cloud host Parameter	**Default Values**
Number of LTM	10
# of local trans	Up to 500
# of op in each UTrans	6–12
Prob(write op)	0.3–0.5
Lock time/op	10
CC protocol	2 Phase Locking

Table 4. Edge Host Parameters

Edge host Parameters	**Default Values**
# of Edge units	500
# of edge sub trans	1–10
# of op/sub-trans	6–12
Prob(write op)	0.3–0.5
Execution cost at the edge host	2–5 Sec/operation
# of visited *IOT-TM (*Mobility)	5–50

Table 3 presents the cloud host parameters, which are also essential for accurate simulation of the proposed system. The number of LTM, local transactions, and operations per unit transaction are critical parameters that affect the capacity of cloud hosts to process transactions. The probability of write operation and lock time per operation also affect the efficiency of the transaction processing system in the cloud. Moreover, the chosen **concurrency control (CC)** protocol can significantly affect the consistency and reliability of the transaction processing system.

Table 4 provides the edge host parameters, which are vital for accurately simulating the performance of edge hosts in the proposed system. The number of edge units, sub-transactions, and operations per sub-transaction affect the processing capacity of edge hosts. The probability of write operation is also a critical parameter that affects the efficiency of transaction processing at the edge. The execution cost at the edge host parameter is particularly crucial, as it determines the time and resources required to process transactions at the edge. Additionally, the number of visited IoT-TM during mobility is a crucial parameter that affects the mobility of IoT devices and their ability to remain connected to edge hosts during movement.

The experiments aim to compare the performance metrics of the various execution strategies while studying the effect of the mobility ratio under varying disconnection scenarios. The effect of fluctuating network circumstances on the response time of transaction processing using various execution strategies is displayed in Figures 5 and 6. One thousand Internet of Things transactions are created in each trial. Depending on the situation, the mobility timer can be anywhere from 5 seconds to 1 second.

In Figure 5, the outcomes are displayed when an edge host does not experience disconnection (BW = 1000). It is evident that the CHS is the most effective of the three techniques, while the EHS is the least effective. This is mainly due to the fact that an IoTT scheduled as a data request transaction requires more data transmission than a transaction request. In addition, the time required for a global commitment during a data request transaction is higher. Given that the BHS strategy is more similar to the CHS than the EHS, most mobile subtransactions are scheduled as transaction requests. It's possible that the BHS won't surpass the CHS method even if the network is completely interconnected. As a result, the BHS strategy takes into account more than just network connectivity when deciding where the mobile subtransaction is executed. Other factors, such as

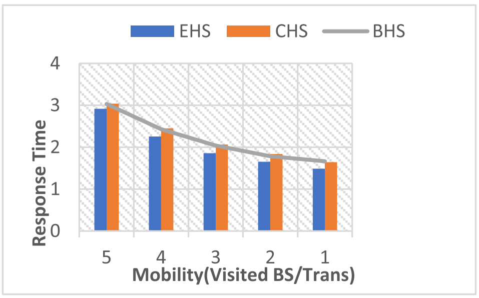


Fig. 5. Response time for IoTT.

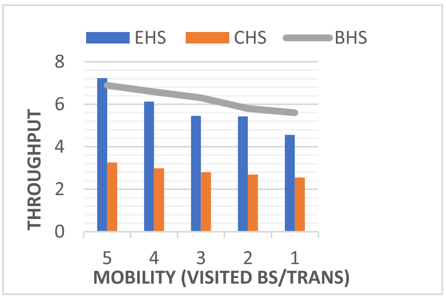


Fig. 6. Throughput for IoTT.

the amount of data being sent and the availability of funds, also play a role. It is possible for a transaction to be scheduled as a data request even if the network is fully connected if the cache status shows that cached data for the transaction is valid. Therefore, the IoT Transaction's subtransactions can be scheduled in a more ad hoc fashion than the CHS method allows, depending on the mobility vector associated with the transaction at the time. We can observe that lowering the disconnecting value slows reaction times across the board. However, as the likelihood of disconnection grows, CHS's performance drops below that of the other two techniques, while BHS continues to excel. The reason for this is because more mobile subtransactions are scheduled as data requests when there is a higher possibility of network disruption. But because the mobile subtransaction at connection time was scheduled as a transaction request, this environment provides more opportunities for the BHS strategy to use the mobility information, making it superior to CHS at disconnection time and EHS at connection time. The CHS tactic has a better chance of preventing the initial stage of executing a mobile subtransaction, the basic transaction. If the cash status of the necessary data items is valid, however, an EHS or BHS strategy can use cached data to conduct the basic transaction without the need to block a mobile subtransaction. Compared to the CHS method, the response time is drastically reduced.

The Internet of Things Transaction (IoTT) Processing Framework is a collection of smaller transactions executed on a mobile device, allowing the atomicity to be loosened. Rather than focusing on a particular IoTT model, the study assesses various execution strategies under varying network connectivity constraints. The framework proposes that an IoTT is a collection of smaller transactions executed on a mobile device, where each subtransaction is broken down into the most fundamental part (Tb) and the most fundamental part of its complement (Tc). Two types of commit points are created: local commit and global commit. Local commit is recorded when all the basic subtransactions have completed processing, while global commit is performed after changes made to data during a subtransaction are reflected in the master copy. The transaction is committed only when its foundational and ancillary subtransactions are also committed.

Processing a mobile subtransaction involves two phases, with the first phase consisting of a straightforward subtransaction where the bulk of the transactional work is done. The second-phase mobile subtransaction processing is a derivation of the original basic subtransaction that is carried out once the data items on the fixed component and the mobile part are linked. Each component of an IoTT is typically scheduled as either a data request or a transaction request, with the IoTT host and cloud server handling the fundamental and complementing subtransactions, respectively. If there is inadequate connectivity between the IoTT host and the edge host's local base station, the basic subtransaction will be executed using cached data. If some information is missing, the transaction is canceled; otherwise, a complementary subtransaction is generated and added to the queue.

The execution strategy determines the position of the fundamental subtransaction. The IoTT Processing Framework has limitations and challenges related to network connectivity and bandwidth issues that can affect the execution of the transactions. These challenges can be addressed by implementing appropriate strategies such as using a hybrid approach that combines both edge and cloud computing, as well as implementing efficient caching techniques. Additionally, reducing the number of transactions sent over the network by batching them together and optimizing the network communication can improve the performance of the IoTT Processing Framework. By overcoming these limitations and challenges, the IoTT Processing Framework can facilitate efficient transaction processing in the context of IoT applications.

A greater amount of time is needed for processing on an edge host when using the EHS method as opposed to the BHS strategy. The time spent on the edge host is decreased because more mobile subtransactions are scheduled as transaction requests with the BHS. When it comes to lowering edge host power consumption, BHS continues to perform better than EHS even as the frequency of network outages rises. However, Figure 6 displays experimental data for system throughputs in terms of the number of completed transactions within a time interval as the network disconnection probability increases. In the first test, results are presented under varying conditions of workload, including those in which the mobility value is set to 20 and the disconnection value is set to 150. There are two goals for this study. In the first place, you need to ensure that your simulation model is accurate. This verification can be accomplished by comparing the execution strategies' response time and throughput gains. Using the fact that throughput under a given execution strategy should be inversely proportional to response time under the same conditions, the second goal of this experiment is to demonstrate how the BHS provides superior performance over alternative strategies as workload increases at the median network connection.

The BHS approach outperforms the EHS and CHS options on a regular basis. Figure 6 shows that the CHS strategy has the maximum throughput in a connected state, the EHS approach has the lowest, and the BHS approach falls somewhere in the middle. This finding is in line with the one presented in Figure 6, which demonstrates that a method's response time can be slower the lower its throughput. However, all techniques degrade in Figure 6 when the probability of being disconnected is significant. Since the fundamental transaction under the CHS strategy is unable to go through during network disconnection, the BHS is capable of producing superior throughput. Even while the EHS can use cached data to complete transactions, it does not take into account the time delay caused by the network's connection.

Transaction processing has practical applications in various fields of the social Web of Things, and some examples and case studies illustrate this. In smart home automation, multiple devices can access and manipulate shared data, such as temperature settings and light controls, using transaction processing, providing a seamless user experience. In industrial IoT, machines and devices can access and manipulate shared data, such as production schedules and inventory levels, ensuring efficient production processes. Healthcare monitoring can use transaction processing to allow multiple devices to access and manipulate shared data such as patient health records and treatment plans, enabling healthcare providers to provide timely and effective care. In transportation management systems [25, 26], GPS trackers, cameras, and sensors can collect real-time data on vehicle location, speed, and fuel consumption, and using transaction processing, optimize operations, reduce costs and provide efficient services to customers.

Despite the potential benefits, there are some limitations to transaction processing in cloud edge computing assisted social Web of Things [27]. Complex transaction management algorithms can increase processing time and system overhead, leading to decreased performance. Edge devices may have limited resources and processing power, which can affect system scalability and reliability. There may also be security and privacy concerns when sensitive data is transmitted and stored in a distributed manner across multiple devices and networks [28]. Therefore, it is essential to carefully consider these limitations and challenges to ensure successful implementation and adoption of such systems.

7 CONCLUSION

In this paper, we take a look at the challenges of modeling edge-cloud computing environments and processing IoT transactions. The investigation in this section is extensive, and the outcome is still preliminary. Simulation is used to test how well the proposed method of scheduling transactions performs. This study uses simulated experiments to examine how well three transaction execution algorithms fare when faced with the possibility of network disconnection. Several viewpoints are explored in these studies by modifying the model's characteristics like the mobility timer and the disconnection ratio. If there is little to no network disconnection, simulation results show that. The CHS is the fastest and most responsive of all the systems. This behavior is to be anticipated because the system imitates a traditional fixed network in many respects. The EHS is preferable to the HS when it comes to system throughput and response time, and this is especially true when network connectivity is poor. The BHS approach typically results in better performance in terms of response time, elapsed processing time of an edge host, and overall number of transactions. By considering an edge host and a cloud host to be two distinct data centers, the BHS is able to boost system performance and enable autonomous operations.

LIST OF ACRONYMS

Acronym	Definition
IoT	Internet of Things
WoT	Web of Things
SWoT	Social Web of Things
EHS	Edge Host Strategy
CHS	Cloud Host Strategy
BHS	Hybrid Strategy
ACID	Atomicity, Consistency, Isolation, Durability
IoTT	Internet of Things Transactions
HTTP	Hypertext Transfer Protocol
MQTT	Message Queuing Telemetry Transport
CoAP	Constrained Application Protocol
QoS	Quality of Service
CTS	Concurrent Transactional Services
IoT-TM	IoT Transaction Manager
LTM	Local Transaction Manager
DM	Data Manager

REFERENCES

[1] M. Abu-Elkheir, M. Hayajneh, and N. A. Ali. 2013. Data management for the Internet of Things: Design primitives and solution. *Sensors* 13, 11 (2013), 15582–15612.

[2] H. Ning, Y. Li, F. Shi, and L. T. Yang. 2020. Heterogeneous edge computing open platforms and tools for Internet of Things. *Future Generation Computer Systems* 106 (2020), 67–76.

[3] L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado, L. Villas, L. DaSilva, C. Lee, and O. Rana. 2018. The Internet of Things, fog and cloud continuum: Integration and challenges. *Internet of Things* 3 (2018), 134–155.

[4] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran. 2018. The role of edge computing in Internet of Things. *IEEE Communications Magazine* 56, 11 (2018), 110–115.

[5] G. Wang and F. Xu. 2020. Regional intelligent resource allocation in mobile edge computing based vehicular network. *IEEE Access* 8 (2020), 7173–7182.

[6] A. Al-Qerem, M. Alauthman, A. Almomani, and B. B. Gupta. 2020. IoT transaction processing through cooperative concurrency control on fog–cloud computing environment. *Soft Computing* 24 (2020), 5695–5711.

[7] C. Cicconetti, M. Conti, and A. Passarella. 2020. Architecture and performance evaluation of distributed computation offloading in edge computing. *Simulation Modelling Practice and Theory* 101 (2020), 102007.

[8] Q. Abbas, H. Shafiq, I. Ahmad, and S. Tharanidharan. 2016. Concurrency control in distributed database system. *IEEE* (2016).

[9] A. Al-Qerem, A. Hamarsheh, Y. A. Al-Lahham, and M. Eleyat. 2018. Scheduling of concurrent transactions in broadcasting environment. *KSII Transactions on Internet and Information Systems (TIIS)* 12, 4 (2018), 1655–1673.

[10] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy. 2017. Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study. *IEEE Access* 5 (2017), 9882–9910.

[11] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya. 2011. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 41, 1 (2011), 23–50.

[12] H. R. Chi, R. Silva, D. Santos, J. Quevedo, D. Corujo, O. Abboud, A. Radwan, A. Hecker, and R. L. Aguiar. 2023. Multi-criteria dynamic service migration for ultra-large-scale edge computing networks. *IEEE Transactions on Industrial Informatics* (2023).

[13] B. Liu, Y. Zhang, G. Zhang, and P. Zheng. 2019. Edge-cloud orchestration driven industrial smart product-service systems solution design based on CPS and IIoT. *Advanced Engineering Informatics* 42 (2019), 100984.

[14] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb. 2018. Survey on multi-access edge computing for Internet of Things realization. *IEEE Communications Surveys & Tutorials* 20, 4 (2018), 2961–2991.

[15] M. S. Aslanpour, S. S. Gill, and A. N. Toosi. 2020. Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research. *Internet of Things* 12 (2020), 100273.

[16] F. A. Salaht, F. Desprez, and A. Lebre. 2020. An overview of service placement problem in fog and edge computing. *ACM Computing Surveys (CSUR)* 53, 3 (2020), 1–35.

[17] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. Khan, and S. K. Das. 2022. Edge-computing-driven Internet of Things: A survey. *ACM Computing Surveys* 55, 8 (2022), 1–41.

[18] M. S. Mekala, A. Jolfaei, G. Srivastava, X. Zheng, A. Anvari-Moghaddam, and P. Viswanathan. 2020. Resource offload consolidation based on deep-reinforcement learning approach in cyber-physical systems. *IEEE Transactions on Emerging Topics in Computational Intelligence* 6, 2 (2020), 245–254.

[19] G. Nagarajan, S. Simpson, and T. Sasikala. 2022. *An Application-Oriented Study of Security Threats and Countermeasures in Edge Computing–Assisted Internet of Things.* Auerbach Publications, (2022).

[20] H. Yao, C. Bai, M. Xiong, D. Zeng, and Z. Fu. 2017. Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing. *Concurrency and Computation: Practice and Experience* 29, 16 (2017), e3975.

[21] A. Yousefpour, A. Patil, G. Ishigaki, I. Kim, X. Wang, H. C. Cankaya, Q. Zhang, W. Xie, and J. P. Jue. 2018. QoS-aware dynamic fog service provisioning. *arXiv preprint arXiv:1802.00800* (2018).

[22] K. Cao, S. Hu, Y. Shi, A. W. Colombo, S. Karnouskos, and X. Li. 2021. A survey on edge and edge-cloud computing assisted cyber-physical systems. *IEEE Transactions on Industrial Informatics* 17, 11 (2021), 7806–7819.

[23] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya. 2016. Fog computing: Principles, architectures, and applications. *Elsevier*, (2016).

[24] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. 2016. Edge computing: Vision and challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646.

[25] M. Satyanarayanan. 2017. The emergence of edge computing. *Computer* 50, 1 (2017), 30–39.

[26] A. Botta, W. de Donato, V. Persico, and A. Pescapé. 2016. Integration of cloud computing and Internet of Things: A survey. *Future Generation Computer Systems* 56 (2016), 684–700.

[27] Z. Ashraf, A. Sohail, S. Latif, A. Hameed, and M. Yousaf. 2023. Challenges and mitigation strategies for transition from IPv4 network to virtualized next-generation IPv6 network. *Int. Arab J. Inform. Technol.* 20, 1 (2023), 78–91.

[28] A. Mijuskovic, A. Chiumento, R. Bemthuis, A. Aldea, and P. Havinga. 2021. Resource management techniques for cloud/fog and edge computing: An evaluation framework and classification. *Sensors* 21, 5 (2021), 1832.

Received 15 December 2022; revised 13 May 2023; accepted 15 May 2023