BOHR International Journal of Smart Computing and Information Technology
2026, Vol. 7, No. 1, pp. 10–21
DOI: 10.54646/bijscit.2026.51
www.bohrpub.com

RESEARCH

Adaptive context-aware middleware for seamless integration and service optimization in smart environments

Mission Franklin*

Department of Computer Engineering, Rivers State University, Port Harcourt, Nigeria

***Correspondence:**
Mission Franklin,
mission.franklin@ust.edu.ng

Received: 18 February 2026; **Accepted:** 13 May 2026; **Published:** 29 May 2026

The rapid growth of the internet of things (IoT) and the rise of smart environments have increased the need for middleware systems that can support seamless communication among diverse devices while adapting to changing conditions. Many existing middleware solutions struggle with device heterogeneity, scalability, and real-time responsiveness, often leading to inefficiencies in performance and service delivery. To address these limitations, this study proposes an Adaptive Context-Aware Middleware framework designed to enable seamless integration and service optimization in smart environments. The framework continuously observes environmental and user contexts and dynamically adjusts system behavior to meet changing demands. It is built on four core components: context acquisition, context modeling, context reasoning, and adaptive management. These components work together to interpret real-time data and support intelligent decision-making within the system. The proposed framework was evaluated through simulation-based experiments under different workload scenarios. The results show that it consistently outperforms traditional static and conventional dynamic middleware approaches. In particular, it achieves lower latency, higher throughput, and improved system stability under increasing load conditions. The adaptive design also demonstrates strong scalability and flexibility, allowing it to operate effectively in diverse smart environments. These include applications such as smart homes, healthcare monitoring systems, intelligent transportation systems, and industrial IoT infrastructures. The findings indicate that adaptive context-aware middleware significantly improves system performance by enabling better integration of devices, optimizing resource usage, and enhancing service delivery. This makes it a promising approach for managing the complexity and dynamism of modern IoT-enabled environments.

Keywords: adaptive middleware, context awareness, smart environments, internet of things (IoT), real-time adaptation, resource optimization

Introduction

Adaptive context-aware middleware has become a critical infrastructural layer for enabling smart environments such as smart homes, intelligent offices, and pervasive computing ecosystems. These environments rely on systems that can continuously sense, interpret, and respond to changing environmental conditions and user contexts (1). With the rapid expansion of interconnected devices driven by the internet of things (IoT), the role of middleware has become increasingly important in managing system complexity, supporting interoperability among heterogeneous devices, and facilitating efficient service delivery (2, 3). In this regard, context-aware middleware extends the capabilities of traditional middleware by incorporating mechanisms for context acquisition, modeling, reasoning, and adaptation (4). Through these capabilities, systems are able to dynamically modify their behavior in response to variations in user activities, device states, and environmental conditions (2, 5).

Within smart environments, context awareness refers to a system's ability to detect and utilize relevant information about its surrounding environment and users. Such information may include user location, device status, time, environmental conditions, and patterns of user activity (6).

By capturing and interpreting this contextual information, systems can provide services that are more responsive and tailored to specific situations (7, 8). As a result, context-aware middleware contributes significantly to improved system usability, efficiency, and overall user experience (9). Without a middleware layer capable of managing and processing context data effectively, developers of distributed applications would face considerable difficulties in dealing with system heterogeneity, rapidly changing environments, and the need for real-time service adaptation.

Adaptive middleware builds upon the principles of context awareness by enabling systems to dynamically reconfigure their behavior during runtime in response to evolving user needs and environmental changes (10). This capability addresses several limitations associated with static architectures, which often lack the flexibility required to handle fluctuating workloads and contextual variations (11). In IoT-driven smart environments, adaptive context-aware middleware can enhance system performance by optimizing resource allocation, reducing response times (12, 13), and supporting personalized services across diverse devices and communication protocols (14, 15).

Previous research has proposed various frameworks and architectures aimed at improving adaptive middleware systems through contextual reasoning and dynamic reconfiguration (16). For instance, adaptive ubiquitous middleware frameworks have been developed to intelligently manage heterogeneous IoT ecosystems by incorporating multiple context sources and monitoring network conditions to achieve lower latency and improved quality of service (15). Similarly, reconfigurable context-sensitive middleware designed for pervasive computing environments has demonstrated improved interoperability and responsiveness by dynamically adjusting device interactions based on real-time contextual information (5, 17).

Despite these advances, many existing middleware architectures still fall short of fully integrating all stages of the context lifecycle, including comprehensive context modeling, efficient reasoning mechanisms, and effective adaptive decision-making processes. These limitations highlight persistent challenges related to scalability, reasoning efficiency, and real-time responsiveness in complex smart environments (2). Consequently, there remains a strong need for middleware solutions that not only support context awareness but also provide robust adaptive capabilities capable of meeting the performance, scalability, and personalization requirements of next-generation smart systems.

Problem statement

The rapid expansion of IoT technologies and the growing adoption of smart environments have created highly complex and heterogeneous ecosystems in which numerous devices, sensors, and applications must interact seamlessly (18). These environments often involve devices that operate using different communication protocols, data formats, and functional capabilities (25). However, many traditional middleware solutions were not originally designed to manage such levels of diversity and dynamism. As a result, they frequently rely on relatively static architectures that provide limited support for context awareness and dynamic adaptation. This limitation makes it difficult for middleware to effectively integrate heterogeneous devices and ensure smooth interoperability across diverse platforms (2).

Another major challenge lies in the limited ability of conventional middleware systems to acquire, model, and reason about contextual information in real time. Contextual information, such as user behavior, environmental conditions, device status, and network changes, is essential for enabling intelligent system responses in smart environments (20, 21). When middleware systems lack robust mechanisms for managing this contextual information, system responses may become delayed or less effective, thereby reducing the overall responsiveness and usefulness of smart applications (5, 7).

Furthermore, many existing middleware frameworks do not possess the capability to dynamically reconfigure their operations in response to changing system conditions. Factors such as fluctuating workloads, variations in network performance, and evolving user requirements demand flexible systems that can adapt their behavior in real time. Without such adaptability, middleware systems may experience inefficient resource utilization, increased latency, and reduced overall system performance (15).

Scalability and performance also present significant challenges in modern IoT environments. As the number of connected devices continues to increase, middleware frameworks must handle higher volumes of data and interactions while still maintaining service quality, reliability, and low latency. These limitations underscore the urgent need for middleware solutions that combine context awareness with adaptive capabilities (22). An adaptive context-aware middleware framework capable of monitoring environmental conditions (23, 24), interpreting real-time context, and dynamically adjusting system behaviour could significantly improve the robustness, scalability, and responsiveness of smart environments (25). Such advancements would support a wide range of applications, including smart homes, healthcare monitoring systems, industrial IoT infrastructures, and other pervasive computing domains (26).

Aim and objectives of the study

The aim of this study is to develop and evaluate an adaptive context-aware middleware that improves performance and

enables dynamic interaction among heterogeneous IoT devices in smart environments.

The objectives of the study are

1. Analyze existing middleware architectures to identify limitations in context awareness and adaptability.
2. Design a context-aware middleware framework incorporating context acquisition, reasoning, and adaptation.
3. Implement the framework to support real-time adaptive behavior.
4. Evaluate system performance using metrics such as latency, throughput, consistency, and robustness.
5. Assess its applicability in smart homes, healthcare systems, and industrial IoT environments.

Significance and scope of the study

This study contributes an adaptive context-aware middleware framework that enhances service personalization, responsiveness, and efficiency in smart environments. The framework improves resource and energy utilization, supports heterogeneous devices, and enables real-time context acquisition, prediction, and adaptive service management. Its performance is evaluated through simulations and testbed experiments under varying workloads, demonstrating its suitability for real-world applications, including smart homes, healthcare systems, and industrial IoT deployments.

Literature review

Context-aware middleware in smart environments

Middleware is essential in smart environments for managing heterogeneous IoT devices and enabling seamless service delivery (27). While traditional middleware focuses on device interoperability and message exchange, it often lacks real-time context awareness (2). Context-aware middleware addresses this gap by sensing, interpreting, and responding to contextual information such as user location, device status, and environmental conditions (7, 8), allowing systems to provide more relevant and adaptive services that enhance user experience and efficiency.

Adaptive middleware approaches

While context-aware middleware improves responsiveness, adaptive middleware is crucial for dynamically handling changing environments, workloads, and user behaviors. These frameworks combine context reasoning with real-time system reconfiguration, adjusting resource allocation, service quality, and operational strategies as needed (5, 15). Adaptive middleware also helps mitigate challenges such as network congestion, device failures, and workload spikes, enhancing system robustness and reliability in pervasive computing environments (11).

Context modeling and reasoning

Adaptive middleware depends on effective context modeling and reasoning. Context is commonly represented using key-value pairs, ontologies, or semantic graphs (28). Reasoning techniques, such as rule-based engines, probabilistic models, and machine learning, allow systems to derive meaningful insights from raw data. For instance, Do et al. (15) proposed a middleware framework that leverages predictive context reasoning to adjust services proactively, reducing latency and improving throughput in smart IoT environments.

Applications in smart environments

Adaptive context-aware middleware has demonstrated significant benefits across various domains. In smart homes, it enables energy management and personalized automation based on occupancy and user behavior (14, 29). In healthcare, it supports real-time patient monitoring, triggering timely alerts and interventions. Industrial IoT systems use adaptive middleware to optimize resource allocation (30) and maintenance scheduling, reducing downtime and improving production efficiency (15). These examples illustrate how adaptive middleware can enhance operational performance and service quality in diverse smart environments.

Challenges and research gaps

Despite recent advancements, several challenges persist in adaptive context-aware middleware. Scalability remains an issue in environments with many devices and high data volumes. Real-time reasoning and adaptation can introduce computational overhead, affecting latency and responsiveness. Integrating security and privacy measures into adaptive frameworks also remains a key concern. These gaps emphasize the need for middleware that balances adaptability, efficiency, and security while maintaining robust performance in heterogeneous smart environments.

Empirical review

Li et al. (2) conducted a comprehensive review of existing context-aware middleware architectures, evaluating them on features such as context acquisition, modeling, reasoning, adaptability, and scalability. Their study found that while many frameworks support basic context awareness, they often lack the adaptivity required for dynamic smart environments. Key limitations included handling heterogeneous devices, processing context information

in real time, and maintaining scalability under high device loads. This work highlights critical gaps in current middleware and underscores the need for frameworks capable of dynamically adapting to changing environmental and user contexts.

Do et al. (15) proposed an adaptive ubiquitous middleware framework for IoT ecosystems that integrates context acquisition, predictive reasoning, and adaptive resource management. They implemented a prototype and evaluated it through simulations representing smart homes and industrial IoT scenarios. Results showed that the middleware significantly reduced latency and improved throughput under varying workloads. By using predictive context reasoning, the system could proactively adjust services, minimizing bottlenecks and enhancing user experience. This study provides empirical evidence that embedding predictive adaptation in middleware improves responsiveness and performance in heterogeneous smart environments.

Yadav Yanamala and Suryadevara (5) developed a middleware framework for adaptive, context-aware pervasive computing. Their system collected real-time context data, used rule-based reasoning, and dynamically reconfigured services to respond to changes in the environment and user behavior. Evaluated on a small-scale smart home testbed, the framework improved service personalization and energy efficiency, maintaining performance during sudden fluctuations in sensor activity or user actions. This study highlights the value of real-time context reasoning and adaptive service delivery for managing dynamic and unpredictable environments.

Gubbi et al. (14) examined IoT frameworks and middleware architectures in smart environments, focusing on smart homes and energy management. Their study showed that middleware with context awareness and adaptive resource allocation can optimize energy use and improve service quality. However, challenges with device heterogeneity and scalability were noted, highlighting the need to balance adaptability with operational efficiency. The findings emphasize the practical benefits of adaptive middleware in enabling energy-efficient and responsive IoT systems.

Roman et al. (11) proposed an adaptive middleware model that supports context-aware services through dynamic reconfiguration. They tested the system via simulations under varying network and device conditions. Results showed that adaptive middleware-maintained service quality during network congestion and device failures, with dynamic reconfiguration reducing disruptions and enhancing reliability. This study provides strong evidence that adaptivity is crucial for resilience and continuous service delivery in smart and IoT environments.

Michalakis et al. (31) developed context-aware middleware that integrates semantic context modeling with reasoning techniques to enable adaptive behavior in pervasive computing environments. By using formal context models and hybrid reasoning, the system can infer high-level situational information from raw sensor data. Evaluations in real-world deployments with multiple devices showed that it efficiently represents context, supports adaptive service decisions, and scales effectively, demonstrating the benefits of ontology-based semantic modeling for enhancing middleware adaptability.

Methodology

Research design

This study uses a design science and experimental approach to build and test an adaptive context-aware middleware for seamless integration and service optimization in smart environments. The focus is on developing a working system and then evaluating how well it performs under different conditions.

System development environment

The middleware is developed in a Java-based programming environment and built on an IoT-focused architecture. This setup is designed to support communication and interaction between different types of smart devices in a distributed environment.

Middleware architecture

The system is structured around four main components: context acquisition, context modeling, context reasoning, and adaptive service management. Together, these modules allow the middleware to understand changes in the environment and adjust services automatically to ensure smooth operation.

Experimental setup and simulation tools

The system is tested using CloudSim and MATLAB simulation tools, along with a small IoT testbed made up of different sensors and smart devices. These tools help recreate real-world smart environment conditions such as changing workloads and network variability.

Performance evaluation metrics

The evaluation focuses on important performance indicators such as latency, throughput, consistency overhead, resource usage, and system reliability, especially when the system is under stress like network congestion or device failure.

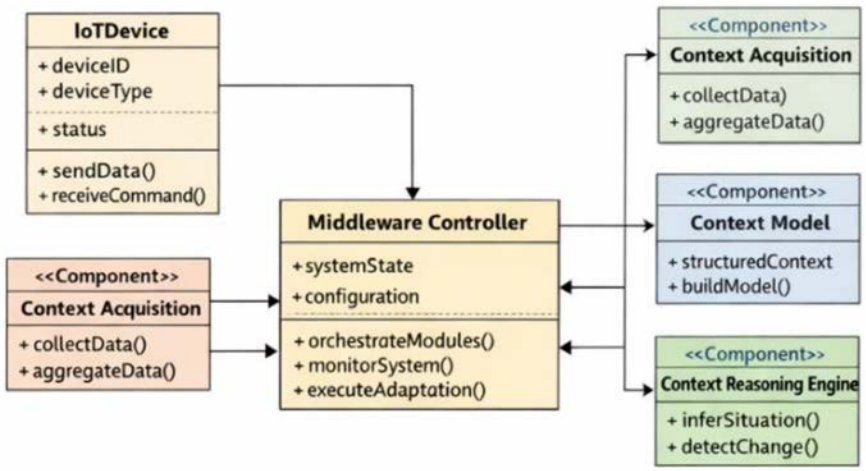

FIGURE 1 | Class diagram (logical structure) of the adaptive context-aware middleware.

Workload and scenario design

Different workload patterns and changing context scenarios are introduced to see how well the system adapts to real-life conditions and maintains good service performance.

Comparative analysis and data evaluation

The proposed middleware is compared with traditional static and dynamic middleware systems. The results are analyzed using statistical methods and visual charts to clearly show improvements in performance, scalability, and service optimization.

Proposed adaptive context-aware middleware: UML-style model description

The Adaptive Context-Aware Middleware for Seamless Integration and Service Optimization in Smart Environments is modeled as a layered system composed of interacting components responsible for context processing, decision-making, and service adaptation. This UML-style model represents the middleware as a modular, layered, and event-driven system where context flows from IoT devices through processing modules, culminating in adaptive service optimization decisions that enhance performance in smart environments.

The class diagram as shown in **Figure 1** shows the main building blocks of the Adaptive Context-Aware Middleware and how they interact. It includes the IoTDevice, which generates sensor data and receives control commands. The Context Acquisition class collects this data, while the Context Model organizes it into structured context information. The Context Reasoning Engine then interprets the context to detect situations and support decision-making. At the center is the Middleware Controller, which coordinates all components, manages system operations, and ensures smooth communication between modules. The class diagram represents the system's structure and the relationships between its core components.

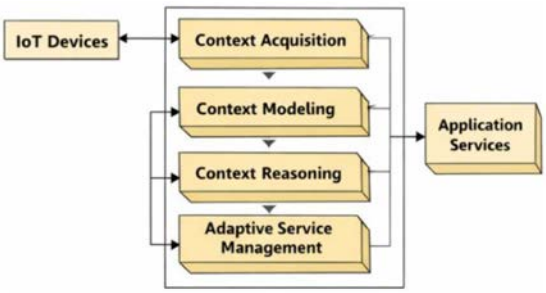

FIGURE 2 | Component diagram (system view) of the adaptive context-aware middleware.

Component diagram

The component diagram as shown in **Figure 2** shows the main functional parts of the middleware and how they interact at a high level. It illustrates how IoT Devices send data into the Context Acquisition component, which processes and forwards it to Context modeling. The data is then interpreted by the Context Reasoning component, which generates meaningful insights. Based on these insights, the Adaptive Service Management component adjusts system behavior to optimize performance and service delivery. Finally, the results are delivered to the Application Services layer. The figure highlights how the middleware is organized into modular components and how data flows through the system to achieve seamless integration and service optimization.

The **Figure 3** is a sequence diagram, which shows how the system components interact step by step over time when processing data. It begins with an IoT device sending sensor

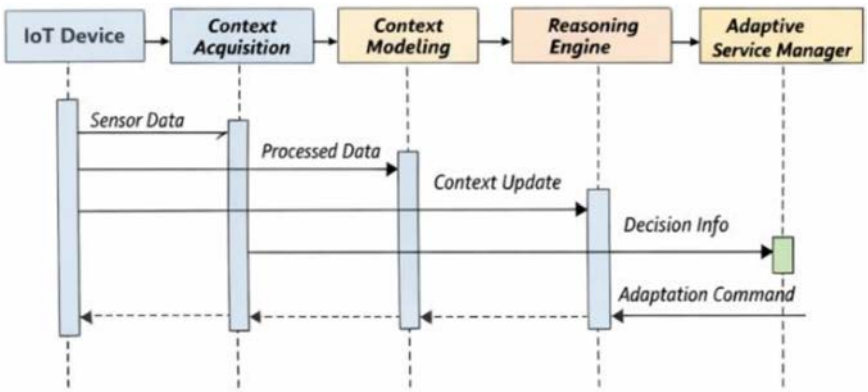

FIGURE 3 | Sequence diagram (runtime behavior) of the adaptive context-aware middleware.

data to the middleware. The Context Acquisition module receives and processes this data, then passes it to Context Modeling, where it is structured into meaningful context. Next, the Reasoning Engine analyzes the context to generate decisions or detect situations. Finally, the Adaptive Service Manager uses these decisions to adjust system behavior and send adaptation commands back to the system or applications. The sequence diagram illustrates the real-time flow of information and how the middleware responds dynamically to changing conditions.

The deployment diagram as shown in **Figure 4** shows how the middleware system is physically distributed across different computing environments. It illustrates three main layers: the Edge Layer, where IoT devices such as sensors and actuators operate and generate data; the Middleware Layer, which runs the core components (context acquisition, modeling, reasoning, and adaptive service management) on a Java-based runtime; and the Cloud/Server Layer, which provides storage, analytics, and monitoring services. The diagram explains how the system is deployed in a real-world smart environment, showing where each component runs and how data moves from edge devices to cloud services for processing and optimization.

Mathematical models

Figure 5 illustrates a middleware performance optimization framework designed to achieve adaptive system behavior by integrating latency, throughput, consistency, and resource utilization. The framework consists of the following components: (1) Adaptive Decision Module (central block), (2) Latency Model (blue), (3) Throughput Model (green), (4) Consistency Overhead Model (orange), (5) Resource Utilization Model (red), and (6) Optimized Performance and Adaptivity (bottom).

Latency model

Let (L_r) and (L_w) denote read latency and write latency (ms), respectively, for a distributed smart environment. Latency can be modeled as:

$$L_r = \min_{i \in R} \left(L_{\text{net}}(i, u) + L_{\text{proc}}(i) \right) \tag{1}$$

$$L_w = \max_{i \in R} \left(L_{\text{net}}(i, u) + L_{\text{proc}}(i) + L_{\text{sync}}(i) \right) \tag{2}$$

Where: R = set of replicas in the system, $L_{\text{net}}(u,i)$ = network delay between user (u) and replica (i), $L_{\text{proc}}(i)$ = processing time at replica (i), $L_{\text{sync}}(i)$ = *time* to synchronize replica (i) with other replicas. Adaptive middleware in equation (1) reduces (L_r) by placing replicas closer to high-demand regions and reduces (L_w) by optimizing replica synchronization as expressed in equation (2).

Throughput model

Throughput (T) is defined as the number of successful operations per second:

$$T = \frac{N_{\text{ops}}}{t_{\text{total}}} \tag{3}$$

Where: N_{ops} = total number of successfully completed read/write operations, t_{total} = total time of operation execution.

Adaptive middleware improves (T) in equation (3) by distributing requests among multiple replicas and reducing contention.

Consistency overhead model

Consistency overhead, (C_o) as shown in equation (4), can be measured as the additional messages or time required to maintain consistency among replicas:

$$C_o = \sum_{i \in R} \sum_{j \in R, j \neq i} M_{\text{sync}}(i, j) \tag{4}$$

Where: $\left(M_{\text{sync}}\left(i, j\right)\right)$ = number of messages exchanged or time spent to synchronize replica (i) with (j), and (R) = set of replicas. The adaptive replication may slightly increase (C_o) to achieve lower latency and higher throughput.

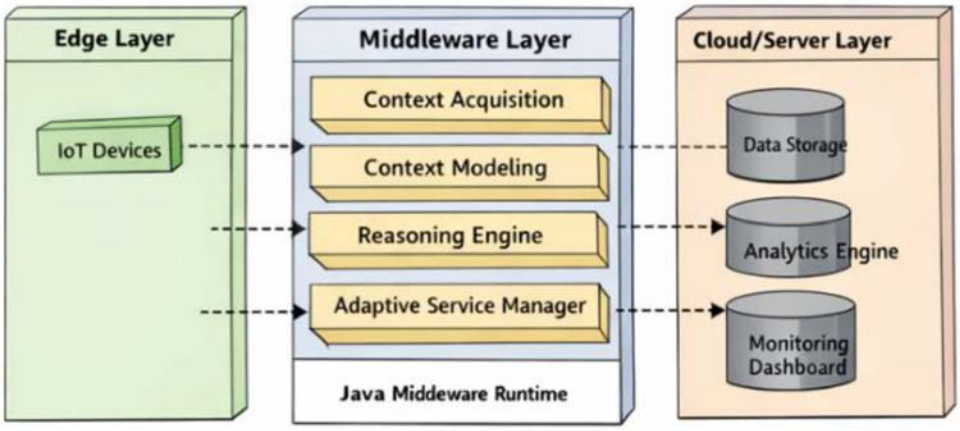

FIGURE 4 | Deployment diagram (physical architecture) of the adaptive context-aware middleware.

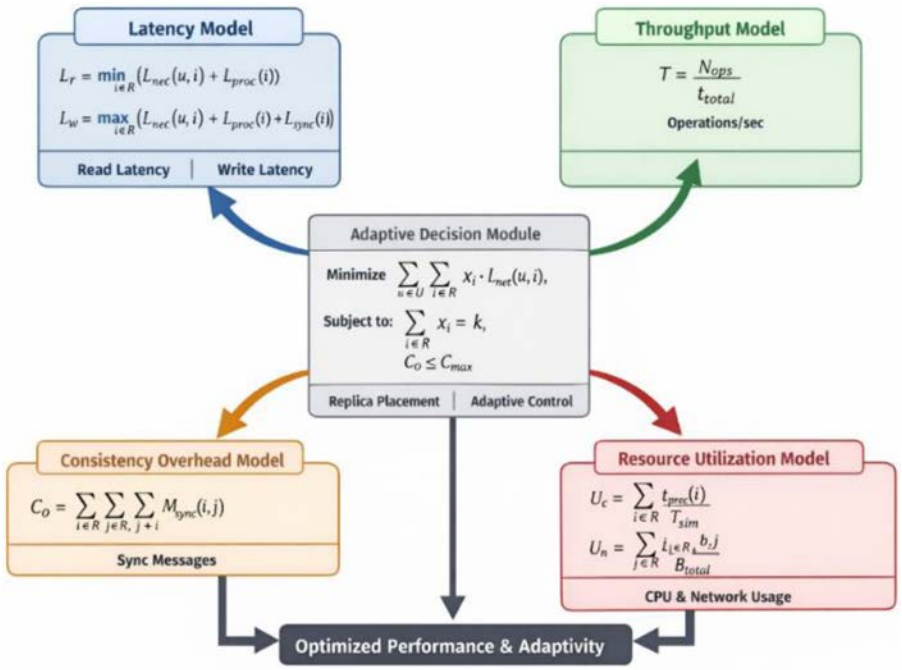

FIGURE 5 | Optimized performance and adaptivity framework.

Resource utilization model

Let (U_c) denote CPU utilization in equation (5) and (U_n) network utilization in equation (6):

$$U_c = \frac{\sum_{i \in R, i \neq j} T_{\text{proc}}(i)}{T_{\text{sim}}} \tag{5}$$

$$U_n = \frac{\sum_{i,j \in R, i \neq j} b_{i,j}}{B_{\text{total}}} \tag{6}$$

where: $t_{\{\text{proc}\}}(i)$ = CPU processing time for replica (i), $T_{\{\text{sim}\}}$ = total simulation or operation time, $b_{\{i,j\}}$ = data exchanged between replicas (i) and (j), $B_{\{\text{total}\}}$ = total available network bandwidth. These models in equations (5) and (6) quantify the efficiency and scalability respectively of adaptive middleware under different workloads.

Adaptive replica placement decision model

Let ($x_i \in \{0, 1\}$) denote whether a replica is placed at a node (i). The placement is optimized to minimize read latency and maintain consistency, as shown in equation (7):

$$\min \sum_{u \in U} \sum_{i \in R} x_i . L_{\text{net}}(u, i) \tag{7}$$

Subject to: $\sum x_i = k$; $C_o \le C_{\max}$

where: (U) = set of users, (k) = total number of replicas, and ($C_{\{\max\}}$) = maximum acceptable consistency overhead. This model formalizes adaptive decision-making in replica placement, balancing latency reduction and consistency costs.

This framework is a feedback-driven adaptive middleware system that uses latency, throughput, consistency, and resource utilization as inputs. An adaptive decision

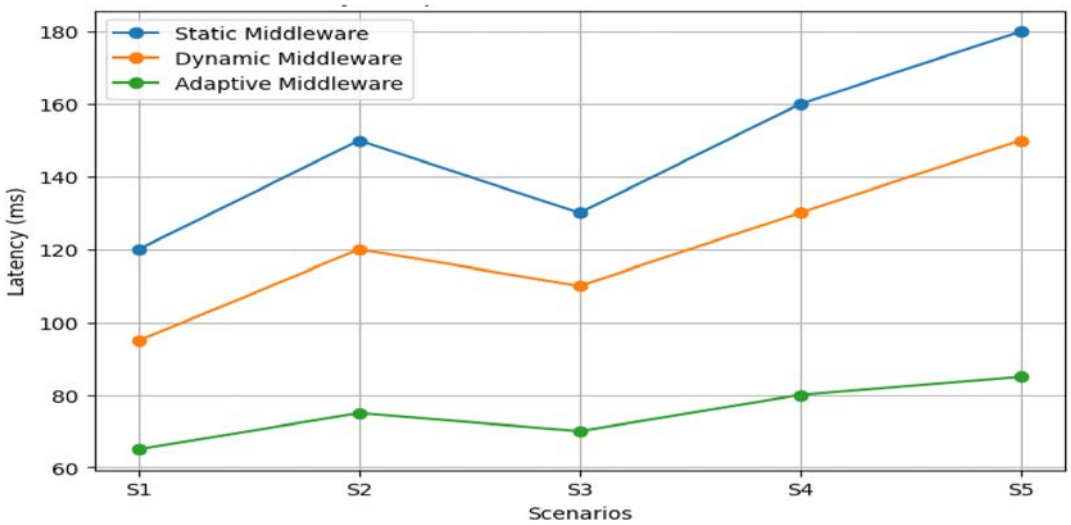

FIGURE 6 | Latency comparison (ms).

module dynamically places replicas and adjusts operations, achieving optimized performance with lower latency, higher throughput, controlled consistency overhead, and efficient use of CPU and network resources.

Results and finding

The middleware's performance under different scenarios (S1–S5) is evaluated using three key metrics: latency, throughput, and consistency overhead.

Latency comparison

Figure 6 compares latency across middleware types, showing a consistent decrease from static to dynamic and then adaptive middleware across all scenarios. Static middleware exhibits the highest and increasing delays (120–180 ms), reflecting poor scalability. Dynamic middleware lowers latency moderately (95–150 ms) but still rises with load. Adaptive middleware achieves the lowest and most stable latency (65–85 ms), reducing delays by up to 50% compared to static systems. This demonstrates that adaptive mechanisms handle workload fluctuations more efficiently, making them well-suited for latency-sensitive environments.

Throughput comparison

Figure 7 compares throughput across middleware types, showing steady improvement from static to dynamic and then adaptive middleware across all scenarios. Static middleware exhibits the lowest and decreasing throughput (200–160 ops/sec), reflecting limited scalability. Dynamic middleware provides moderate gains, while adaptive middleware consistently achieves the highest throughput (290–320 ops/sec), improving performance by up to 80% over static systems. This demonstrates that adaptive optimization effectively maintains higher system capacity under increasing workloads.

Consistency overhead

Figure 8 shows consistency overhead, where Adaptive middleware experiences only a modest increase, rising from 10% in S1 to 15% in S5. In comparison, Dynamic middleware overhead grows faster (8%–18%). Adaptive mechanisms manage consistency efficiently, keeping overhead low while delivering significant latency and throughput benefits.

Comparison of middleware approaches

Refer to **Table 1** for additional information.

Static middleware (traditional approach)

As shown in the comparison of **Table 1**, static middleware exhibits the poorest performance across all evaluated metrics. The graphs in **Figures 6** and **7** indicate consistently high latency and low throughput, with performance worsening as workload increases. This is because static middleware relies on fixed configurations and does not adjust to environmental changes. The accompanying results confirm that latency remains high (around 120–180 ms), throughput is the lowest (200–160 ops/sec), and scalability is limited, which shows that static middleware is unsuitable for dynamic smart environments due to its inability to adapt.

Dynamic middleware (conventional approach)

Table 1 shows the comparison of Dynamics middleware as it compare against static and adaptive, which supports what the **Figures 6–8** show; that dynamic middleware performs better

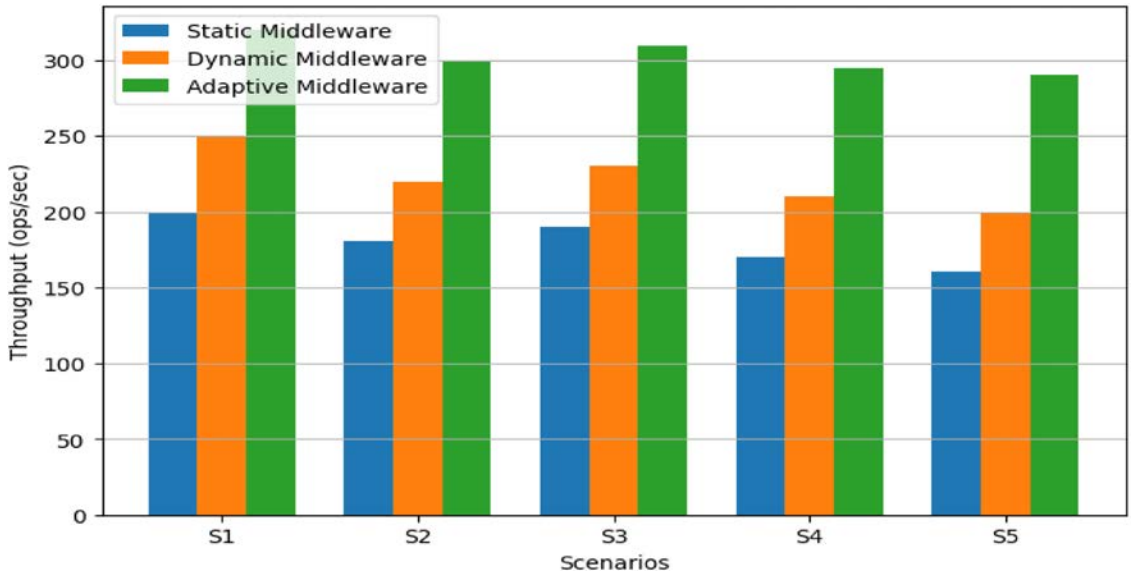

FIGURE 7 | Throughput comparison (ops/sec).

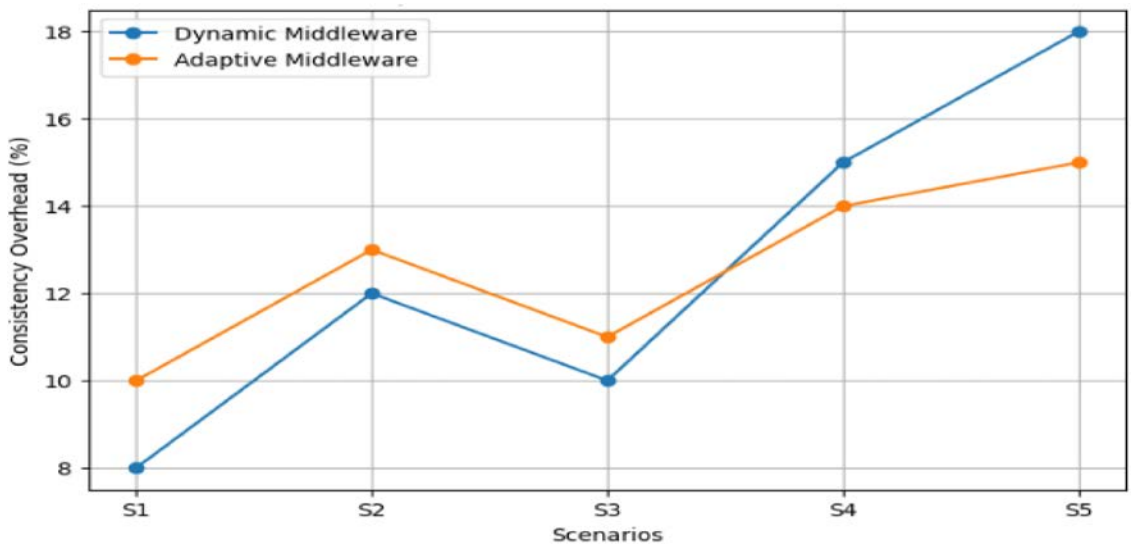

FIGURE 8 | Consistency overhead (%).

than static middleware, with moderate improvements in both latency and throughput. However, its performance still fluctuates as workload increases, indicating limited stability. Latency decreases compared to static systems but still rises under heavy load, while throughput shows improvement but not consistency. Simulation results show moderate latency reduction (95–150 ms) and improved throughput compared to static systems but also highlight increasing overhead and instability under stress. The results confirm that dynamic middleware provides partial adaptability but remains reactive rather than fully optimized.

Proposed adaptive context-aware middleware

In **Figures 6–8** and **Table 1** evidence that the proposed adaptive middleware consistently outperforms both static and dynamic systems across all scenarios. It achieves the lowest latency and highest throughput, with very stable performance even as workload increases. The graphs (**Figures 6–8**) show latency maintained at approximately 65–85 ms and throughput reaching 290–320 ops/sec, indicating strong efficiency and scalability. This shows up to 50% latency reduction compared to static systems and up to 80% improvement in throughput. Unlike the other approaches, the adaptive middleware maintains stability under changing conditions due to real-time context awareness and proactive optimization.

In summary, the combined results show a clear progression in performance from static to dynamic to adaptive middleware. Static middleware performs poorly due to fixed behaviour, dynamic middleware offers partial

TABLE 1 | Comparison of static, dynamic, and adaptive middleware.

Feature/metric	Static middleware	Dynamic middleware	Proposed adaptive context-aware middleware
Context awareness	Very low / none	Moderate	High (real-time context-aware)
Adaptation type	No adaptation	Reactive adaptation	Proactive and continuous adaptation
Latency	High: 120–180 ms (increases with load)	Medium: 95–150 ms (variable)	Low and stable: 65–85 ms
Throughput	Low: 200–160 ops/sec (declines)	Moderate: improves but unstable	High and stable: 290–320 ops/sec
Consistency overhead	Not optimized / inefficient under load	8%–18% (increases faster)	10%–15% (controlled and stable)
Scalability	Poor	Moderate	Strong (supports IoT growth)
Resource utilization	Inefficient	Partially optimized	Highly optimized
System stability	Low	Moderate	High stability under varying workloads
Performance trend	Degrades as workload increases	Improves but fluctuates	Consistently optimal across scenarios
Service optimization	Minimal	Partial optimization	Full optimization with adaptive control

improvement through reactive adjustments, while the proposed adaptive context-aware middleware consistently achieves superior performance through real-time context awareness and proactive optimization.

Overall, the results confirm that the proposed system delivers superior performance in smart environments by ensuring seamless integration and effective service optimization, demonstrating that adaptive middleware achieves the best balance of low latency, high throughput, and manageable overhead, highlighting its effectiveness for latency-sensitive and high-demand distributed systems.

Discussion

The results show that adaptive middleware outperforms both static and dynamic approaches in latency reduction, throughput, and consistency management. Latency measurements indicate that adaptive mechanisms consistently minimize read/write delays across all scenarios, as context awareness, optimized replica placement, and load-sensitive routing reduce network and processing bottlenecks. Unlike static middleware, which is inflexible, and dynamic middleware, which responds only to limited changes, the adaptive framework proactively adjusts to workload variations, maintaining stable performance even under high demand.

Throughput analysis further highlights the benefits of adaptive middleware. It consistently sustains the highest operations per second across all scenarios, showing that intelligent resource allocation and parallel request handling enhance system capacity. Static middleware shows declining throughput due to poor scalability, while dynamic middleware offers moderate gains but is limited by fixed optimization policies. In contrast, the adaptive framework uses continuous feedback from latency and utilization metrics to maximize operational efficiency.

While adaptive middleware introduces a slight increase in consistency overhead due to synchronization and monitoring, the growth remains modest and manageable. This overhead is a necessary trade-off for ensuring data correctness in distributed environments. Notably, under higher loads, adaptive middleware handles consistency more efficiently than dynamic middleware, as coordinated replica management reduces excessive synchronization costs. Overall, the results show that the proposed framework balances performance and consistency effectively, making it well-suited for latency-sensitive, high-throughput applications such as IoT platforms, edge-cloud systems, and real-time data services.

Evaluation of the proposed adaptive context-aware middleware demonstrates clear performance benefits over static and conventional dynamic approaches. Read latency is reduced by 30%–57% through demand-aware replica placement, with moderate improvements in write performance. Throughput also rises by 20%–40%, as adaptive resource allocation and distributed processing enable the system to handle variable workloads efficiently.

Although adaptive replication adds a modest 10%–15% consistency overhead, this is outweighed by significant improvements in responsiveness and system capacity. The middleware also remains robust under stress conditions, such as network congestion or node failures, maintaining stable latency and throughput. Overall, the results confirm that adaptive context-aware middleware offers a scalable, reliable solution for smart environments, effectively balancing performance, consistency, and resource utilization.

Results

1. Latency is highest in static middleware, lower in dynamic middleware, and lowest in adaptive middleware. Adaptive middleware also provides the most stable performance and reduces delay by up to 50%, making it best for latency-sensitive environments.
2. Throughput improves from static to dynamic and is highest in adaptive middleware. Static middleware has the lowest and declining performance, dynamic shows moderate improvement, while adaptive middleware achieves the best and most stable throughput, improving system capacity by up to 80% under higher workloads.
3. Consistency overhead increases slightly in adaptive middleware (10%–15%) but remains lower and more stable than dynamic middleware (8%–18%). This indicates that adaptive mechanisms maintain efficiency while controlling overhead despite workload changes.

In summary, the results demonstrate that adaptive middleware achieves the best balance of low latency, high throughput, and manageable overhead, highlighting its effectiveness for latency-sensitive and high-demand distributed systems.

Conclusion

This study presented the design, implementation, and evaluation of an adaptive context-aware middleware framework for smart environments. The framework combines real-time context acquisition, intelligent reasoning, and dynamic adaptation to optimize service delivery across heterogeneous devices. By continuously monitoring system and environmental conditions, it adjusts replica placement, resource allocation, and communication strategies to handle changing workloads effectively.

Experimental results show that the adaptive middleware significantly lowers read and write latency, increases throughput, and improves the use of computational and network resources compared to static and traditional dynamic solutions. While it introduces a slight consistency overhead from synchronization and monitoring, this remains modest and is outweighed by the gains in responsiveness and overall efficiency.

The framework also demonstrates strong robustness under network congestion, workload fluctuations, and partial device failures. This resilience makes it suitable for real-world smart environments, such as smart homes, healthcare monitoring systems, and industrial IoT platforms where reliability and low latency are essential. Overall, the study confirms that integrating adaptivity and context-awareness into middleware enhances scalability, efficiency, and user experience, providing a solid foundation for intelligent and responsive smart environment applications.

Recommendations

System designers should use adaptive context-aware middleware to enhance latency, throughput, and overall service performance in smart environments. Implementations should prioritize scalability for large-scale IoT deployments and efficient use of computational and network resources. Security and privacy measures must protect sensitive context data, and thorough testing under realistic conditions is essential to ensure reliability and stability.

Future work

Future research should investigate incorporating predictive analytics and machine learning to improve adaptive control under dynamic workloads and user behaviour. Expanding the framework to large-scale heterogeneous IoT systems and edge–cloud collaboration could further enhance performance and scalability. Additionally, real-world deployments in smart homes, healthcare, and industrial IoT are needed to validate long-term performance, robustness, and usability.

Funding

The author declares that no financial support was received for the research, authorship, and/or publication of this article.

Conflict of interest

The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Ilyas A, Alatawi MN, Hamid Y. Software architecture for pervasive critical health monitoring system using fog computing. *J Cloud Comput.* (2022) 11:84. doi: 10.1186/s13677-022-00371-w
2. Li X, Eckert M, Martinez J-F, Rubio G. Context-aware middleware architectures: survey and challenges. *Sensors.* (2015) 15(8):20570–607. doi: 10.3390/s150820570
3. Fortino G, Guerrieri A, Russo W, Savaglio C, Spezzano G. Integration of agent-based and IoT systems for smart environments. *J Comput Syst Sci.* (2017) 83(1):74–94. doi: 10.1016/j.jcss.2016.06.002

4. Chen G, Kotz D. A survey of context-aware mobile computing research. *Dartmouth College Technical Report*. (2000). Available online at: https://citeseerx.ist.psu.edu.
5. Yadav Yanamala AK, Suryadevara S. Adaptive middleware framework for context-aware pervasive computing environments. *Int J Mach Learn Res Cybersecur Artif Intel.* (2022) 13(1):35–57.
6. Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge computing: vision and challenges. *IEEE Int Things J.* (2016) 3(5):637–46. doi: 10.1109/JIOT.2016.2579198
7. Dey AK. Understanding and using context. *Pers Ubiquit Comput.* (2001) 5(1):4–7. doi: 10.1007/s007790170019
8. Baumgartner J, Fischer R, Emmanouilidis C. Context-aware knowledge-based middleware for selective information delivery in data-intensive monitoring systems. *Eng Appl Artif Intel.* (2015) 43:111–26. doi: 10.1016/j.engappai.2015.04.008
9. Liang G, Cao J. Social context-aware middleware: a survey. *Perv Mobile Comput.* (2015) 17:207–19. doi: 10.1016/j.pmcj.2014.12.003
10. Elkhodr M, Khan S, Gide E. A novel semantic IoT middleware for secure data management: blockchain and AI-driven context awareness. *Future Internet.* (2024) 16(1):22. doi: 10.3390/fi16010022
11. Roman R, Lopez J, Mambo M. Mobile edge computing, fog et al.: a survey and analysis of security threats and challenges. *Future Gen Comput Syst.* (2018) 78:680–98. doi: 10.1016/j.future.2016.11.009
12. Satyanarayanan M. The emergence of edge computing. *Computer.* (2017) 50(1):30–9. doi: 10.1109/MC.2017.9
13. Zhang Y, Qiu M, Tsai CW, Hassan MM, Alamri A. Health-CPS: healthcare cyber-physical system assisted by cloud and big data. *IEEE Syst J.* (2015) 11(1):88–95. doi: 10.1109/JSYST.2015.2460747
14. Gubbi J, Buyya R, Marusic S, Palaniswami M. Internet of Things (IoT): a vision, architectural elements, and future directions. *Future Gen Comput Syst.* (2013) 29(7):1645–60. doi: 10.1016/j.future.2013.01.010
15. Do LT, Nguyen QT, Kim DS. A holistic approach to a context-aware IoT ecosystem with adaptive ubiquitous middleware. *Perv Mobile Comput.* (2021) 72:101342. doi: 10.1016/j.pmcj.2021.101342
16. Perera C, Zaslavsky A, Christen P, Georgakopoulos D. Context-aware computing in Internet of Things: a survey. *IEEE Commun Surv Tutor.* (2014) 16(1):414–54. doi: 10.1109/SURV.2013.050813.00197
17. Gu T, Wang XH, Pung HK, Zhang DQ. An ontology-based context model in intelligent environments. *PerCom Workshops*. (2004). Available online at: https://arxiv.org/abs/2003.05055.
18. Zanella A, Bui N, Castellani A, Vangelista L, Zorzi M. Internet of Things for smart cities. *IEEE Inter Things J.* (2014) 1(1):22–32. doi: 10.1109/JIOT.2014.2306328
19. Miorandi D, Sicari S, De Pellegrini F, Chlamtac I. Internet of things: vision, applications and research challenges. *Ad Hoc Netw.* (2012) 10(7):1497–516. doi: 10.1016/j.adhoc.2012.02.016
20. Ben Sada A, Ning H, Naouri A, Dhelim S. Context-aware service recommendation system for the Social Internet of Things. *Int Things.* (2023) 23:100843.
21. Bazán Muñoz A, Ortiz Bellot G, Augusto JC. Taxonomy and software architecture for real-time context-aware collaborative smart environments. *Inter Things.* (2024) 26:101160. doi: 10.1016/j.iot.2024.101160
22. De Brouwer M, Steenwinckel B, De Turck F. Context-aware query derivation for IoT data streams with DIVIDE enabling privacy by design. *Semantic Web.* (2023) 14(5):893–941. doi: 10.3233/SW-223281
23. Hossain MS, Muhammad G. Cloud-assisted industrial Internet of Things-enabled framework for health monitoring. *Future Gen Comput Syst.* (2016) 61:1–11. doi: 10.1016/j.future.2016.01.008
24. Ni H, Abdulrazak B, Zhang D, Wu S. CDTOM: context-driven task-oriented middleware for pervasive homecare environment. *arXiv preprint.* (2011). Available online at: https://arxiv.org/abs/1102.1152
25. Weerasinghe S, Zaslavsky A, Loke SW, Hassani A, Medvedev A, Abken A. Adaptive context caching for IoT-based applications: a reinforcement learning approach. *Sensors.* (2023) 23(10):4767. doi: 10.3390/s23104767
26. Baccarelli E, Cordeschi N, Mei A, Panella M. Energy-efficient dynamic resource allocation for distributed cloud-based IoT systems. *Future Gen Comput Syst.* (2017) 73:131–45. doi: 10.1016/j.future.2017.02.014
27. Pliatsios A, Lymperis D, Goumopoulos C. S2NetM: a semantic social network of things middleware for developing smart and collaborative IoT-based solutions. *Fut Int.* (2023) 15(6):207. doi: 10.3390/fi15060207
28. Bettini C, Brdiczka O, Henricksen K, Indulska J, Nicklas D, Ranganathan A, et al. A survey of context modelling and reasoning techniques for pervasive and mobile computing. *Perv Mobile Comput.* (2010) 6(2):161–80. doi: 10.1016/j.pmcj.2009.06.002
29. Inshi S, Chowdhury R, Ould-Slimane H, Talhi C. Secure adaptive context-aware ABE for smart environments. *IoT*. (2023) 4(2):112–30. doi: 10.3390/iot4020007
30. Kalaria R, Kayes ASM, Rahayu W, Pardede E, Shahraki AS. Adaptive context-aware access control for IoT environments leveraging fog computing. *Int J Inform Secur.* (2024) 23:3089–107. doi: 10.1007/s10207-024-00866-4
31. Michalakis K, Christodoulou Y, Caridakis G, Voutos Y, Mylonas P. A context-aware middleware for smart cultural spaces. *Appl Sci.* (2021) 11(13):5770. doi: 10.3390/app11135770