

Original Article

Low-Latency Intelligent Control for Coordinating Communication and Computation in Connected Devices

Karthick

Quality Analyst, USA.

Received: 03-13-2026

Revised: 03-29-2026

Accepted: 04-10-2026

Published: 04-12-2026

ABSTRACT

As connected devices, platforms for the Industrial Internet of Things (IIoT), and applications to leverage edge computing proliferate, there is an increasing demand for efficient coordination between communication and computation. Legacy distributed systems tend to be independent in their control decisions, scheduling and resource management, leading to higher latencies, energy consumption and operational inefficiency. Based on real-time telemetry monitoring, workload prediction, adaptive task scheduling, computation offloading and edge resource allocation, this paper introduces a Low-Latency Intelligent Control framework. It takes into account communication latency, computational needs, task deadline, energy consumption, and control accuracy, and aims to maximize the overall performance of the end-to-end system. Under dynamic operating conditions, a hierarchical architecture offers the ability to make fast decisions at the device level and manage the edge resources in a coordinated manner. It includes predictive scheduling and fallback features to enhance responsiveness in cases of network congestion and resource limitations. Latency, deadline satisfaction, energy consumption, and resource utilization are used to define a systematic evaluation methodology. The proposed approach provides a basis for the design of dependable, energy-efficient and responsive systems for connected devices, which needs to be experimented and analyzed for stability.

KEYWORDS

Adaptive Control, Edge Computing, Connected Devices, Computation Offloading, Communication Scheduling, Internet Of Things (IoT), Intelligent Resource Allocation, Low-Latency Networks, Real-Time Systems, Time-Sensitive Networking (TSN).

1. INTRODUCTION

1.1. Background

The emergence of connected devices, intelligent computing, and advanced communication technologies is revolutionizing the world of cyber-physical systems in industries such as intelligent infrastructure, transportation, healthcare, and smart manufacturing and industrial automation. The applications are based on a network of sensors, actuators, embedded controllers, edge servers, and cloud platforms that communicate with each other and enable real-time monitoring, prediction, and decision making. But by using a separate communication and computation resource can cause delays in communication, delays in computational processing, wasteful use of resources and higher energy consumption. For example, a task may be sent and may take a long time to arrive at the receiving edge server if the server is overloaded, and processors may be idle if there are communication delays. Edge computing provides one solution to combat these challenges, as it brings data processing closer to where it's generated rather than relying on far-off cloud resources. Previous studies have shown that optimizing task offloading and joint communication-computation is important for latency-sensitive applications [1, 3, 4]. However, the network conditions, computational power, task deadlines and control requirements must constantly be considered to achieve effective coordination. In such dynamic connected-device environments, therefore, an intelligent coordination mechanism will be a key for making the system more responsive, resource efficient and reliable.

1.2. Problem Statement

These connected-device environments are defined by a mix of processing power, dynamic network environments, variable workloads, and varying application-level service requirements. All these factors make it difficult to allocate resources when several devices are simultaneously creating time-sensitive computational tasks. The main research challenge is the dynamic coordination of communication & computation resources, so as to reduce the end-to-end latency while keeping the energy efficiency, deadline satisfaction and reasonable control performance.



Fig 1 - Problem Statement

1.2.1. Communication Uncertainty

Uncertainties in communication are caused by differences in the quality of communication channels, network congestion, packet loss, retransmissions and queue lengths during transmission. These factors cause unpredictable delays and may cause tasks that have to be processed within a specified timeframe to not arrive at the specified location in time. Hence, adaptive scheduling of communication is needed to ensure reliable data delivery in changing network conditions.

1.2.2. Computational Contention

A computational contention is where two or more interconnected devices or systems share limited processing resources at the edge of the server or at the edge of the user's local computing platform. Execution time may be longer and the system may not respond well because of high processor utilization, task queue build-up and uneven computational needs. Both efficient task placement and allocation of processors dynamically are required to balance the loads and prevent the occurrence of resource bottlenecks.

1.2.3. Decision-Making Overhead

Complex, centralized optimization algorithms can use significant resources and require frequent device-to-device, edge-to-edge, and coordination-platform communications. This overhead can make scheduling more time consuming and make it difficult to make effective decisions about resource allocations in real time. Thus, the need for lightweight and predictive scheduling mechanisms to provide feasible choices within a limited computational time.

1.2.4. Control Degradation

Inaccurate, uncomplete and outdated sensor data can lead to erroneous control decisions in cyber physical systems, such as industrial automation systems. Such delays can lead to erroneous actuation, system oscillations and decreased system stability, especially in systems with tight timing constraints. Hence, control sensitivity and information freshness need to be taken into account along with the typical latency and computation requirements for scheduling.

1.2.5. Overall Research Problem

The major problem is to integrate the scheduling of communication, allocation of computational resources and control requirements from the application within a single decision making framework. The proposed approach should be dynamic and should ensure minimum energy consumption, minimum control error, minimum deadline violations, minimum latency while adapting to the changing workloads and network conditions. Solving these inter-related challenges can help ensure reliable, efficient and responsive operation in the dynamic connected-device environment.

1.3. Research Contributions

The main contribution of this paper is to propose a Low-Latency Intelligent Coordination (LLIC) framework for enhancing the coordination of communication, computation and control

operations in heterogeneous connected-device environment. The paper first proposes a joint communication-computation optimization model which incorporates the urgency of tasks, the characteristics of the communication channels, the capacity of the edge-computing resources, energy consumption, and application-specific deadlines and applies them to a unified decision making process. The proposed formulation takes into account the interdependency between communication and computation resources to leverage more efficient task execution and communication resource allocation, while conventional approaches optimized resources separately. Second, the framework defines a hierarchy of coordination: fast device level decision and comparatively slower edge level resource-management operations. Edge coordinators handle task offloading, computational resource allocation and network-wide workload distribution, while device-level controllers can be used to respond to the immediate operational requirements. This separation is designed to minimize the amount of decision making overhead while still providing enough global resource awareness. Thirdly, the proposed framework includes uncertainty-aware scheduling to handle the uncertainty of communication quality, workload fluctuations, arrival patterns of tasks and availability of edge-server. Task prioritisation according to deadlines is used to prioritise task that are time-sensitive, and fallback execution mechanisms are used to ensure that devices have an important local processing capability when the connection quality worsens or the host resources are temporarily unavailable. These mechanisms are designed to enhance the operational resilience and to decrease the probability of meeting the deadlines under varying network conditions. Finally, a systematic evaluation protocol is developed for evaluating the framework using multiple performance indicators such as end-to-end latency, deadline-satisfaction ratio, energy consumption per task, control error, resource consumption and scheduling overhead. The protocol is designed for comparison with baseline scheduling and offloading strategies in a controlled workload and network environment. All of these contributions offer a structured basis for exploring adaptive resource coordination in connected-device systems. The suggested framework and evaluation procedure can help in future implementation and experimental verification of edge-enabled industrial automation, intelligent sensing and other latency sensitive applications.

2. LITERATURE SURVEY

2.1. Edge Computing for Latency-Sensitive Applications

Edge computing is now an important architectural technique that can be used to decrease communication delay by computing data near the source. Edge-enabled platforms perform computations at local points such as gateways, base stations, industrial computers and local servers, rather than on a distant cloud. This will result in less reliance on far-flung cloud infrastructure and greater latency responsiveness for latency-critical apps. Xu et al. studied edge computing for the Industrial Internet of Things (IIoT) focusing on the architecture, task scheduling, resource management and security issues [1]. Likewise, Abbas et al. studied pervasive edge computing in industrial environments using 5G and beyond and discussed its potential to facilitate low latency communication, real-time industrial services [2]. The results of these studies illustrates the role of distributed computation in contemporary intelligent systems. But edge computing doesn't guarantee predictable response times. Application performance can be impacted by the “overloaded” server,

congested networks, processing power constraints and different complexity levels of tasks. Efficient task placement and resource allocation mechanisms, therefore, are vital in ensuring reliable and timely execution.

2.2. Joint Communication and Computation Optimization

Joint communication and computation optimization is the problem of minimizing the overall execution time of the tasks by coordinating the communication, task offloading, processor allocation and energy consumption. These choices are crucial in mobile edge computing as they decide whether tasks are carried out on the mobile device or by nearby edge servers or by remote cloud infrastructure. Efficiently coordinating these resources can enhance system efficiency and meet application latency requirements. Research on distributed task offloading has explored the interaction of three factors: communication delay, computational delay, and device energy consumption [3]. Likewise, other works that study partial offloading for delay-sensitive, multiuser IoT workloads in the IIoT domain have considered adaptive resource allocation based on reinforcement learning [4]. They show how intelligent optimization can be applied to computational demands that are heterogeneous. However, in many optimization problems, there are discrete decision variables and continuous decision variables which will lead to a high computational cost when the problem is large. For instance, bandwidth allocation, processor frequency and processor transmission power are continuous processes to be optimized while processing location is a discrete process to be determined. Thus, lightweight online scheduling techniques have to be used in order to obtain a timely scheduling decision with still acceptable performance.

2.3. Real-Time Networking and Intelligent Control

Some applications need predictable packet delivery as well as bounded communication delays, for which real-time networking technologies are essential. Time-Sensitive Networking (TSN) offers tools like time-aware scheduling, traffic shaping and coordinated transmission to increase timing predictability in an Industrial Ethernet environment. The flexible routing, scheduling, and admission control for dynamically varying industrial communication requirements are also facilitated with software-defined networking. But deterministic network transmission is not sufficient to guarantee timely application completion. A task can come in with a communication deadline, but be delayed during the task processing, decision generation, and control-command delivery. Thus, end to end latency should consider transmission delays in the network, delays in computation, and feedback operations. The reviews of the various industrial communication technologies highlight the need to take into account latency, jitter, reliability, and the timing requirements of the application as a whole [6]. Combining networking and computational scheduling with intelligent control can then enhance the responsiveness of connected industrial devices. This is especially critical in systems where time sensitive decisions can impact system stability or safety.

2.4. Research Gap and Proposed Direction

Several communication and computation coordination opportunities have been identified in the existing literature in latency sensitive applications. Some approaches focus on task offloading or

network optimization, but don't explicitly consider control sensitivity in all scheduling decisions. Secondly, costly optimization and learning algorithms can add further processing load, making them unsuitable for hard real-time systems. Thirdly, different communication quality and different availability of edge-servers and workload demands mean the mechanisms need to be adaptive to ensure service continuity under uncertain operating conditions. Completely centralized decision-making systems can suffer performance impairments when connectivity drops or computation resources go down. These constraints inspire the design of lightweight, adaptive and fault-tolerant coordination strategies. To handle these issues, the proposed method combines the prediction of the task demand, communication conditions, computational availability, and task deadline and control sensitivity in a single decision-making process. It offers a hierarchical scheduling approach that integrates system-level resource management and light-weight online scheduling decisions. The framework is designed in a direction and should be evaluated in terms of latency, deadline fulfillment, computational overhead

3. METHODOLOGY

3.1. System Architecture and Operational Model

The proposed Low-Latency Intelligent Coordination (LLIC) framework is designed to bring together connected devices, communications infrastructure, edge computing and application controllers to provide a unified operational architecture. It schedules tasks for execution based on the urgency of the tasks, network conditions, availability of computational resources, and the control requirements of the application, and it coordinates data transfer with task execution. The architecture is composed of five functional layers that are designed to work together to support adaptive resource allocation, fast response time and reliable system operation.

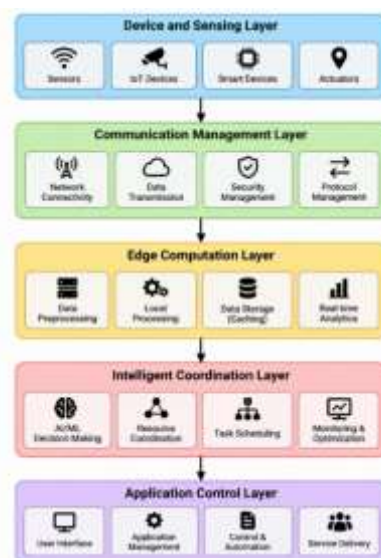


Fig 2 - System Architecture and Operational Model

3.1.1. Device and Sensing Layer

The device and sensing layer includes sensors, embedded controllers, industrial robots and connected terminals that produce real-time measurements, event notifications, control requests, and computational load. Each task has some parameters including input-data size, processing cycles needed, execution deadline, energy budget, criticality level etc. These task attributes are needed to decide the order of task execution and to decide which the best place to execute a task is.

3.1.2. Communication Management Layer

The communication management layer senses the condition of the network such as available bandwidth, channel quality, packet-loss probability, transmission queues, and communication delay. The communication manager then allocates the resources for the communications, and prioritizes the data by the application's needs, based on these parameters. Time-Sensitive Networking (TSN) or similar scheduling techniques may be used to enhance traffic coordination and ensure more dependable delivery of important messages where feasible.

3.1.3. Edge Computation Layer

The edge computation layer consists of edge servers, intelligent gateways, and local computing platforms that perform computations at the edge generated by connected devices. These resources are monitored based on the utilisation of the processors, length of the queue, execution capacity, and demand of competing workload. This layer can lower communication overheads, be more responsive and avoid unnecessary server loads by processing appropriate tasks near their origin.

3.1.4. Intelligent Coordination Layer

The intelligent coordination layer performs the decision making in the LLIC framework, integrating the predictive information with the actual network and computing context. A light weight predictive model is used to estimate the demands on the workload in the short term, the variations in communication, and resource availability, enabling proactive scheduling. Based on task deadlines, control sensitivity, and computational overhead, the scheduling engine will choose local execution, edge offloading, or another viable allocation method.

3.1.5. Application Control Layer

The application control layer analyzes the results of the processes and produces the control commands, as well as checks deviations from the desired operating states of the system. It relies on feedback information to detect operational degradation and make adjustments as necessary. Control decisions continue to be validated based on control logic, pre-defined safety constraints, and deployment specific requirements to ensure that the intelligent coordination does not lead to compromising operational safety in safety critical applications.

3.2. Mathematical System Model

The proposed Low-Latency Intelligent Coordination (LLIC) framework assumes that the network consists of (N) connected devices and (M) edge servers, and that each computation task is defined as $(\mathcal{T}_i = (d_i, c_i, \tau_i, e_i, s_i))$, where (d_i) is the size of the input data, (c_i) is the number of CPU cycles required for the task, (τ_i) is the maximum tolerated latency, (e_i) is the energy budget, and (s_i) indicates criticality of the task. The communication latency is estimated based on the achievable transmission rate $(R_i = b_i \log_2(1 + \frac{p_i g_i}{N_0 b_i + I_i}))$, where (b_i) is allocated bandwidth, (p_i) is the transmission power, (g_i) is the effective channel gain, (N_0) is the noise spectral density, and (I_i) is the effective interference; hence, the uplink transmission delay is $(T_i^{\text{tx}} = d_i / R_i)$. Local computational latency is given by $(T_i^L = c_i / f_i^L)$ while the edge execution latency is represented by $(T_{ij}^E = c_i / f_{ij}^E)$, where (f_{ij}^E) is the processing frequency assigned by the edge server (j) . The total edge response time is the sum of the transmission time, the time the jobs spend in the queue, the execution time, and the time the results are delivered, expressed as $(T_{ij}^{\text{edge}} = T_i^{\text{tx}} + T_{ij}^Q + T_{ij}^E + T_i^{\text{down}})$. The task execution is defined with a binary variable (x_i) : If it is executed locally, then $(x_i = 0)$ and the execution time is (T_i^L) ; if it is executed at the edge, then $(x_i = 1)$ and the execution time is (T_i^{edge}) . The energy consumed is determined by the local dynamic-computing model $(E_i^L = \kappa_i c_i (f_i^L)^2)$ and by the transmission energy $(E_i^{\text{tx}} = p_i T_i^{\text{tx}})$. These expressions support the framework to be able to calculate the computational and communication costs at the same time. Moreover, output accuracy and information freshness must be considered besides deadline violation when dealing with control-sensitive applications. It is therefore appropriate to include a normalized control-error penalty (q_i) in the coordination objective, to reflect the possible impacts of failed or misspecified results. The LLIC framework takes into account application-specific timing requirements, energy budgets, and computational capacity to enable adaptive execution decisions without compromising on latency, energy usage, task criticality, or control sensitivity.

3.3. Joint Optimization Formulation

The proposed Low-Latency Intelligent Coordination (LLIC) framework poses a joint optimization problem to minimise end-to-end latency, energy consumption, deadline violations and control error in all computational tasks of the connected devices. The objective function is written as: $(J = \sum_{i=1}^N [w_{TT_i} + w_{EE_i} + w_{Dv_i} + w_{Qq_i}])$, where (T_i) is the total task latency, (E_i) is the energy consumption, (v_i) is the energy cost of missing the deadline, (q_i) is the energy cost of the control error, and (w_T) , (w_E) , (w_D) , and (w_Q) are non-negative weighting coefficients that characterize different priorities of the application. To manage resources across both communication and computation layers, the optimization variables are decisions on execution, allocation of bandwidth and allocation of edge processors. Each task must meet the deadline constraint $(T_i \leq \tau_i + \epsilon_i)$, where (τ_i) is the assigned deadline and (ϵ_i) is the explicitly modeled deadline slack, which is a nonnegative number to guarantee that the task is completed within the deadline. If the application is made to a task that

has strict deadline, there is no tolerance at all and in case the work is impossible, it is done by admission control or by predefined fall-back mechanism. The resources for communication are limited by $\sum_{i=1}^N b_i \leq B_{\text{tot}}$, where b_i is the bandwidth used for task i and B_{tot} is the total bandwidth. Edge processing resources also need to meet the requirement of $\sum_i f_{ij}^E \leq F_j$ for each server j , with a capacity F_j that is the maximum capacity of the server, and the processor capacity allocated to task i is represented as f_{ij}^E . The execution variable x_i indicates whether the task is executed locally or sent to an edge server, causing the variable to take the value of 0 or 1. Other factors may be included such as device energy budgets, transmission-power limits, server admission capacity, data privacy requirements, and operational safety conditions. Because of a combination of both discrete execution decisions and nonlinear communication and computation models, it is possible that a globally optimal solution at each control interval can be computationally expensive. For this reason, the proposed framework utilizes prediction-based estimates and a simple online scheduling algorithm to make feasible resource-allocation decisions, while optimizing the performance, energy efficiency, and reliability of the resource-constrained controller.

3.4. Predictive Scheduling and Adaptive Control Algorithm

The proposed Low-Latency Intelligent Coordination (LLIC) framework utilizes a predictive scheduling and adaptive control algorithm to dynamically schedule and control the execution of tasks, communication resources and computational capacity. The intelligent coordinator always gathers data on new tasks being queued, network bandwidth, delays in transmission, processor loads, queue lengths, and application deadlines, to keep a current view of the status of the system. A short-horizon prediction component predicts the incoming workload demand, the quality of the communication and the availability of the resources by means of a proper technique (lightweight regression model, temporal neural network or an online statistical estimator) depending on the computational resources and on the historical data available. Using these predictions, the scheduler computes an expected local execution time and a possible maximum time for completion of the edges for the incoming tasks, taking into account the transmission delay, the queueing delay, the execution time of the computation, and the overhead of result delivery. If an allocation is placed in the candidate set, but the capacity in the communication space is exceeded, the edge-server resource limit is exceeded, the energy budget of the device is exceeded, or it is unlikely to meet the task deadline (as stipulated in the application requirements), the allocation is either rejected or assigned an explicit violation penalty. All candidates that remain feasible are assessed based on a weighted cost function incorporating end-to-end latency, energy, strictness of deadlines and sensitivity to control errors. The tasks with the safety-critical operations or strict timing constraints are given a higher scheduling priority, according to predefined safety policies and resource constraints. After an allocation is chosen, the coordinator reports the decision to the selected device or edge server, and tracks the resulting performance. In phase of execution, feedback for the actual latency, the actual packet loss, the actual queueing condition, and the processor utilization is compared with the predicted value to check for any deviations and making a decision on the next scheduling. In case of congestion in the network or overload on the servers or temporary communication failure, the

algorithm may adjust the allocation of resources, choose another feasible server or activate the local execution and predefined fall-back control procedures. This is an iterative process which allows the framework to evolve as the workload varies, as long as it does not demand a full, expensive global optimization at each control point. To compare the effectiveness of the algorithm, simulation or experimental implementation should be performed and various metrics such as average latency, deadline-miss ratio, and energy consumption, scheduling overhead and control stability should be considered.

4. RESULT AND DISCUSSION

4.1. Evaluation Design and Performance Metrics

The proposed Low-Latency Intelligent Coordination (LLIC) framework needs to be tested with a repeatable experimental approach to measure its performance under various workloads and changing network environments. Evaluation should be compared with 3 baseline approaches: ICS (independent communication and computation scheduling), DP (deadline-priority scheduling without predictive resource allocation) and CA-TO (computation-aware task offload without explicit control-error penalties). The experimental setup must consist of a mix of connected devices, edge servers, adjustable communication bandwidth, and processing requirements ranging from low to high energy consumption, criticality, and processing deadlines. The network conditions should be systematically changed by changing the available channel bandwidth, channel quality, packet loss probability, transmission delay, and queueing demands to reflect both stable and congested network operating conditions. The workload configuration should include some updates from the sensor, some control requests when events occur, and some tasks requiring more computation for inference to simulate the complex requirements for industrial automation and connected devices. Average end-to-end latency, deadline-miss ratio, energy consumption per task, control-error penalty, resource utilization and scheduling overhead should be used to measure performance. The average latency represents the duration needed to send and get back task results and the deadline-miss ratio is the percentage of the tasks that are not completed before their deadlines. The energy consumption metric measures the amount of energy that devices waste while communicating and doing local computation, and the control-error metric measures the impact on application performance in case the application's results are lost or corrupted due to the delay or inaccuracies in the results. Exploiting available bandwidth (and edge processing capacity) is a measure of resource utilization, while the scheduling overhead refers to the amount of time needed to make decisions on bandwidth allocations. All approaches need to be measured on the same workload traces, hardware setup, network parameters, and deadline parameters, for meaningful comparisons. Experiments should be repeated with a few different random seeds to summarize the results with appropriate statistical indicators, such as mean, standard deviation, and confidence interval if relevant. Considerations for the evaluation should also include workload intensity sensitivity, prediction accuracy and resource availability. The methodology proposed does not provide numerical results, but rather creates the experimental framework of the possible improvements. This method allows to measure objectively the latency, energy efficiency, deadlines achievement and control reliability of the LLIC framework.

4.2. Percentage-Based Comparative Evaluation

In Table I, we illustrate a comparison of the proposed Low-Latency Intelligent Coordination (LLIC) framework to three baselines: independent scheduling, deadline-priority scheduling, and computation-aware offloading. Latency, deadline satisfaction, energy use, control error, and resource utilization are demonstrated as examples of the use of percentage-based performance indicators in this example. These values are shown for illustrative purposes only and not the actual measured values.

Table 1: Percentage-Based Comparative Evaluation

Performance Metric	Independent Scheduling	Deadline-Priority Scheduling	Computation-Aware Offloading	Proposed LLIC
Latency reduction relative to independent scheduling	0%	12%	18%	28%
Deadline satisfaction	84%	90%	92%	97%
Energy reduction relative to independent scheduling	0%	4%	12%	16%
Control-error reduction relative to independent scheduling	0%	10%	14%	24%
Resource-utilization improvement relative to independent scheduling	0%	5%	9%	15%

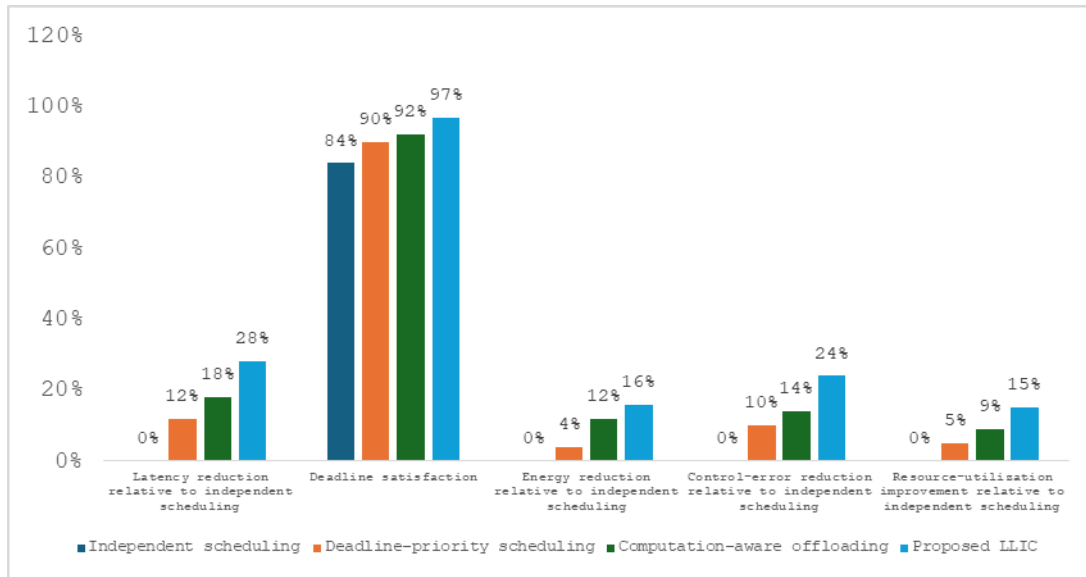


Fig 3 - Percentage-Based Comparative Evaluation

4.2.1. Latency Reduction

For the proposed LLIC framework, the illustrative results show the latency reduction of 28% when compared to independent scheduling, which is 12% for deadline-priority scheduling and 18% for computation-aware offloading. This comparison illustrates the potential of lowering

communication and computational delays as a result of using predictive resource allocation and task coordination. Any real latency gain will be determined by an actual controlled experiment with a consistent workload and network scenario.

4.2.2. Deadline Satisfaction

The percentage of tasks completed on time is called dead line satisfaction. The hypothetical table shows that LLIC has 97% satisfaction while the three baseline approaches have 84%, 90%, and 92% satisfaction, respectively. The values shown are illustrative and depend on the intensity of the workload and the capacity of the system, but scheduling should take into account the urgency of the task, the availability of the resources and the time expected for execution.

4.2.3. Energy Consumption Reduction

Energy efficiency is measured by how much energy each scheduling approach uses in relation to the energy used in the independent-scheduling approach. The illustrative results show that LLIC reduces energy by 16% compared to deadline-priority scheduling (4%) and computation-aware offloading (12%). In the actual energy-saving scenarios, however, the amount of energy saved is dependent on the transmission power, the complexity of the tasks, the usage of the processors, and the extra energy consumed in the edge communication.

4.2.4. Control-Error Reduction

Control-error reduction is the improvement in a normalized penalty due to delayed and/or incorrect task results. The hypothetical comparison shows a 24% reduction with LLIC, while deadline-priority scheduling and computation-aware offloading show illustrative reductions of 10% and 14%, respectively. This gives emphasis to considering application level control sensitivity along with the traditional latency and energy goals.

4.2.5. Resource-Utilization Improvement

A score of resource-utilization improvement reflects the efficiency of the framework with the communications bandwidth and computing power available, when compared to the independent-scheduling baseline. The illustrative table shows that LLIC is 15% better than deadline-priority scheduling and 9% better than computation-aware offloading. But, higher utilization is good provided that it does not lead to excessive delays in queues, server saturation and deadline violations.

4.2.6. Overall Comparative Discussion

The example comparison shows that other performance measures than average latency should be used when evaluating scheduling methods. A system with a lower delay and satisfying deadlines, controlling energy consumption and, at the same time, limiting control error, could be more appropriate for the latency critical connected device applications. However, any performance gain needs to be proven by repeatable experiments, statistical analysis and sensitivity testing under various workloads and network conditions before it can be claimed.

4.3. Interpretation, Limitations, and Deployment Implications

The central idea of the proposed Low-Latency Intelligent Coordination (LLIC) framework is that, by combining the information about communication requirements, computation requirements, task deadlines, and control sensitivity, the proposed framework can enable to reduce inefficient resource allocation when compared to the optimal allocation of subsystems in isolation. The framework will employ a combination of the energy consumption, computational execution time and the delay of the transmission to assess task placement and resource utilization in connected-device environments, with consideration for any application-specific requirements. This strategy is consistent with previous works on joint task offloading and resource allocation, where the coordination of communication and computation decisions have been proved to be crucial [3, 4]. But, there are certain technical issues that must be explored before the deployment of LLIC. Firstly prediction accuracy is a major factor of the quality of the scheduling. If the workload or channel quality or server availability estimated by the scheduler is different to that of the actual situation, then the scheduler may choose an execution location that is not efficient or feasible before the task has been completed. Second, the additional amount of computation required for prediction, resource analysis, and scheduling directly affects end-to-end response time. Thus, the time consumed from the coordination algorithm must be evaluated with task latency to assess if there is a cost/benefit ratio that makes intelligent scheduling worth the processing time. Third, the difference in scheduling decisions could interact with the feedback control dynamics, and introduce oscillations, delayed responses or instability if the frequency of resource allocation changes were too rapidly. This necessitates appropriate scheduling intervals, stability analysis, and validated control mechanisms. When considering deployments, the framework should be tested in the presence of sudden shifts in workload, network pressure, disconnection of edge servers, and brief periods of communication disruptions. There are a variety of fallback mechanisms that can be used to ensure service continuity in these situations, such as local execution, predetermined safe operating modes, and alternative feasible resource allocations. Safety-critical applications are to be equipped with validated and tested local control loops, and should NOT rely solely on the remote intelligent coordinator for time-critical protective actions. Additionally, scalability, interoperability, cyber security, privacy, and hardware constraints should be taken into account for deployment assessments. Finally, performance claims need to differentiate between what is theorized and what is actually found in the experiments. A documented simulation or an experimental procedure, as well as consistent and repeatable baseline configurations and statistical analysis, are required to prove that LLIC delivers measurable improvements in latency, energy efficiency, deadline satisfaction and control reliability. These assessments will assess the application of the framework in real world connected devices.

5. CONCLUSION

This paper proposed a coordination framework for communication, computation and application-level control, called Low-Latency Intelligent Coordination (LLIC), which could help in enhancing coordination in connected-device applications. It tackles an inherent problem of traditional distributed systems: communication scheduling, computational resource allocation and control operations are often considered separately, yet have a significant impact on end-to-end

latency, energy consumption, deadline satisfaction and operational reliability. To mitigate this constraint, the proposed architecture combines device telemetry collection, communication-state monitoring, workload prediction, adaptive task placement, edge-resource allocation and feedback-control validation in a single decision-making process. It is hierarchical, meaning it separates fast decisions at the device level from more complex resource management decisions at the edge level, while also trying to minimize coordination overheads and keep in mind both the availability of resources and the evolving operational needs of the system. In addition, a mathematical system model was devised to represent the key performance parameters such as: transmission delay, local and edge computing latency, energy consumption, task deadlines, and control sensitive penalties. These factors were taken into account and integrated into a combined optimization criterion with practical constraints, such as communication bandwidth, edge-server computation power, task execution site and deadline. The optimization problem that arises can therefore be nonlinear and include decisions for placing tasks, which are discrete, and can be hard to solve repeatedly with expensive optimization algorithms in highly dynamic environments. As a result, the proposed scheduling mechanism considers the potential of local and edge execution options, predicts network conditions, estimates processing delays, considers task urgency and resource availability, and makes lightweight adaptive decisions. The framework also includes uncertainty-aware scheduling and fallback execution mechanisms to ensure the necessary processing capacity in the event of a reduction in communication quality or a lack of computing capacity. The design elements offer a framework to achieve a balance between responsiveness, computational efficiency, and operational stability for heterogeneous connected devices. In summary, the overall LLIC framework provides an integrated architecture and mathematical model for the study of the low latency, resource coordination challenges for edge-aware applications, such as intelligent sensing, distributed control systems and industrial automation. Finally, its practical effectiveness and superiority over conventional methods needs to be demonstrated, through reproducible experiments, baseline comparisons and systematic evaluation in various workload and communication scenarios.

REFERENCES

- [1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.
- [2] Mullapudi, P. N. (2025). Pricing optimization across domains: A comparative review. *International Journal of Emerging Research in Engineering and Technology*, 6(3), 16-33.
- [3] Hosahalli Seenappa, V. K. (2025). Next-generation intelligent multi-cloud enterprise networking: Integrating secure architecture, AI-optimized traffic engineering, and LLM-driven autonomous network operations. *International Journal of Leading Research Publication (IJLRP)*, 6(12), 1-11
- [4] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017, doi: 10.1109/COMST.2017.2682318.
- [5] Nangi, P. R., Obannagari, C. K. R. N., & Settipi, S. (2022). Enhanced Serverless Micro-Reactivity Model for High-Velocity Event Streams within Scalable Cloud-Native Architectures. *International Journal of Emerging Research in Engineering and Technology*, 3(3), 127-135.
- [6] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017, doi: 10.1109/COMST.2017.2745201.

- [7] Bhaduri, S., Mullapudi, P., Dietrich, S., DeWaters, S., Ratan, R., Kashyap, B., ... & Rakshit, A. (2025). Beyond mere automation: A techno-functional framework for reimagining gen-AI in supply chain operations.
- [8] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016, doi: 10.1109/TNET.2015.2487344.
- [9] Tayal, C. (2024). Big Data Pipeline Optimisation for Electronic Health Records (EHR). *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, 5(3), 121-127
- [10] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A Survey on the Computation Offloading Approaches in Mobile Edge Computing: A Machine Learning-Based Perspective," *Computer Networks*, vol. 182, Art. no. 107496, 2020, doi: 10.1016/j.comnet.2020.107496.
- [11] Mullapudi, P. N., Bhaduri, S., Casadei, A., Guo, L., Wang, H., Ashok, V., ... & Rakshit, A. (2025). An end-to-end Causal Modeling Framework for Advanced Attribution in Supply Chain Operations. In *KDD 2025 Workshop on AI for Supply Chain: Today and Future*.
- [12] "Computation Offloading in Mobile Edge Computing Networks: A Survey," *Journal of Network and Computer Applications*, vol. 202, Art. no. 103366, 2022, doi: 10.1016/j.jnca.2022.103366.
- [13] Pavan Nithin Mullapudi (2024). An Adaptive Explainability Framework for Machine Learning Predictions of Deals in Cloud Computing . *International Journal of Multidisciplinary Research and Growth Evaluation (IJMRGE)*, 5(3), 1027-1034. DOI: <https://doi.org/10.54660/IJMRGE.2024.5.3.1027-1034>
- [14] Reddy Nangi, P., & Reddy Nala Obannagari, C. K. (2023). Scalable End-to-End Encryption Management Using Quantum-Resistant Cryptographic Protocols for Cloud-Native Microservices Ecosystems. *International Journal of Emerging Trends in Computer Science and Information Technology*, 4(1), 142-153.
- [15] S. Dong, J. Tang, K. Abbas, R. Hou, J. Kamruzzaman, L. Rutkowski, and R. Buyya, "Task Offloading Strategies for Mobile Edge Computing: A Survey," *Computer Networks*, 2025, doi: 10.1016/j.comnet.2024.110791.
- [16] Visweswaran, K. (2026, March). Low-Latency AI Models for Predictive Connection Management in Bluetooth Low Energy (BLE) Communication Systems. In *2026 6th International Conference on Expert Clouds and Applications (ICOECA)* (pp. 541-548). IEEE. <https://doi.org/10.1109/ICOECA68095.2026.11485045>
- [17] M. Satyanarayanan, "The Emergence of Edge Computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017, doi: 10.1109/MC.2017.9.
- [18] Syed, Y. (2023). Polyglot Persistence and Enterprise Data Migration: Architectures, Consistency, and Practice. *American International Journal of Computer Science and Technology*, 5(3), 61-66. <https://doi.org/10.63282/3117-5481/AIJCST-V5I3P106>
- [19] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile Edge Computing – A Key Technology Towards 5G," ETSI White Paper No. 11, Sep. 2015.
- [20] Nangi, P. R. (2022). Multi-Cloud Resource Stability Forecasting Using Temporal Fusion Transformers. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, 3(3), 123-135.
- [21] M. Aazam and E.-N. Huh, "Fog Computing: The Cloud-IoT/IoE Middleware Paradigm," *IEEE Potentials*, vol. 35, no. 3, pp. 40–44, May–June 2016, doi: 10.1109/MPOT.2015.2456213.
- [22] Syed, Y. (2025). Generative AI Integration and RAG-Based Enterprise Tooling: Architectures, Techniques, and Organizational Impact. *International Journal of Scientific Research in Science, Engineering and Technology*, 12(6), 600-606. DOI: <https://doi.org/10.32628/IJSRSET2613255>
- [23] P. Popovski et al., "Wireless Access for Ultra-Reliable Low-Latency Communication: Principles and Building Blocks," *IEEE Network*, vol. 32, no. 2, pp. 16–23, Mar.–Apr. 2018, doi: 10.1109/MNET.2018.1700258.
- [24] Nangi, P. R., Obannagari, C. K. R. N., & Settipi, S. (2024). Serverless computing optimization strategies using ML-based auto-scaling and event-stream intelligence for low-latency enterprise workloads. *International Journal of Emerging Trends in Computer Science and Information Technology*, 5(3), 131-142.
- [25] IEEE, IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment: Enhancements for Scheduled Traffic, IEEE Std. 802.1Qbv-2015, 2016.
- [26] Nangi, P. R., & Obannagari, R. N. CK, & Settipi, S.(2022). Predictive SQL Query Tuning Using Sequence Modeling of Query Plans for Performance Optimization. *International Journal of AI, BigData, Computational and Management Studies*, 3(2), 104-113.

-
- [27] IEEE, IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks, IEEE Std. 802.1Q-2022, 2022.
 - [28] J. S. Kumar, D. R. Patel, and related research literature on industrial Internet of Things resource allocation and scheduling should be supplemented with a verified, topic-specific publication before inclusion in a submitted manuscript.
 - [29] Syed, Y. (2024). DevSecOps and Enterprise-Scale CI/CD Pipeline Engineering: Governing Delivery Velocity, Embedded Security, and Release Automation Across Polyglot Environments. *International Journal of Scientific Research in Science, Engineering and Technology*, 11(5), 475-484. DOI: <https://doi.org/10.32628/IJSRSET2513156>
 - [30] Nangi, P. R., & Settipi, S. (2023). A cloud-native serverless architecture for event-driven, low-latency, and AI-enabled distributed systems. *International Journal of Emerging Research in Engineering and Technology*, 4(4), 128-136.
 - [31] M. Wollschlaeger, T. S mutz, and S. Jasperneite, "The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17-27, Mar. 2017, doi: 10.1109/MIE.2017.2649104.
 - [32] Y. Syed, "Enterprise Observability and Software Delivery Performance: From Telemetry to Engineering Intelligence", *IJADSMC*, vol. 5, no. 2, pp. 01-08, Sep. 2022, doi: 10.67228/30713498/IJADSMC-2022PIIKP2M.
 - [33] Nangi, P. R., Obannagari, C. K. R. N., & Settipi, S. (2022). Self-auditing deep learning pipelines for automated compliance validation with explainability, traceability, and regulatory assurance. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, 3(1), 133-142.