福建省中电海峡智能装备研究院
Fujian Zhongdian Straits Institute of Intelligent Equipment
Asia Pacific Science Press
Advances in Management and Intelligent Technologies
ISSN(O):3079-5869
ISSN(P):3079-5850
Vol. 2 No.2 (2026)

Research on Architecture Design and Optimization of Cloud-Edge Collaborative Emergency Communication System for Low-Latency Response

Guoli Ying*
Carnegie Mellon University, NASA Research Park, Moffett Field, CA, USA
Corresponding author: Guoli Ying, Stephenyingguoli@outlook.com

Abstract: In terms of the base station destruction, link disconnection and resource contention in the disaster environment of the traditional emergency communication system, there is a clear contradiction between the physical vulnerability of the communication link and its requirements for low-latency response. This paper proposes a cloud-edge collaborative emergency communication architecture to solve this problem. Based on edge collaborative computing and the ability of elastic expansion of the cloud, the architecture creates a communication network with fault self-healing ability by combining task replication scheduling and dynamic resource allocation. The task replication mechanism uses computing resources in exchange for communication efficiency. It can still ensure the continuous execution of tasks when key nodes fail. Dynamic resource allocation is to monitor the load of nodes, the quality of links and the state of energy consumption in real time, so as to achieve the purpose of adaptive distribution of tasks and on-demand scheduling of resources. On this basis, the improved ant colony optimization algorithm is used to complete the rapid deployment of emergency tasks. The efficiency of scheduling and the speed of convergence are improved by improving the pheromone update strategy, the design of heuristic function and the selection of nodes. From the results of theoretical analysis and simulation experiments, under a simulated disaster scenario with three edge node failures (out of a total of 10 edge nodes) and a concurrent task scale of 300-500, the proposed architecture reduces the average task response delay by approximately 40% compared to a baseline cloud-edge collaborative architecture without task replication. Furthermore, the system's task completion reliability reaches over 96% under these conditions, demonstrating significant performance advantages.which can well meet the needs of real-time and stability of communication in emergency situations.

Keywords: Emergency Communication; Cloud Edge Collaboration; Low Delay Response
Published: Apr 20, 2026
DOI: https://doi.org/10.62177/amit.v2i2.1311

1. Introduction

In the emergency communication scenario, because it is close to the data source and the transmission distance is short, edge computing can be used to reduce the service delay to a certain extent [1]. However, relying solely on edge nodes has the disadvantages of limited computing resources, small coverage, and poor power supply capacity. When the number of concurrent tasks increases sharply in the early stage of the disaster, it is easy to cause local overload. Relying solely on a centralized cloud center, although it has strong computing and storage capabilities, it is very sensitive to the stability of

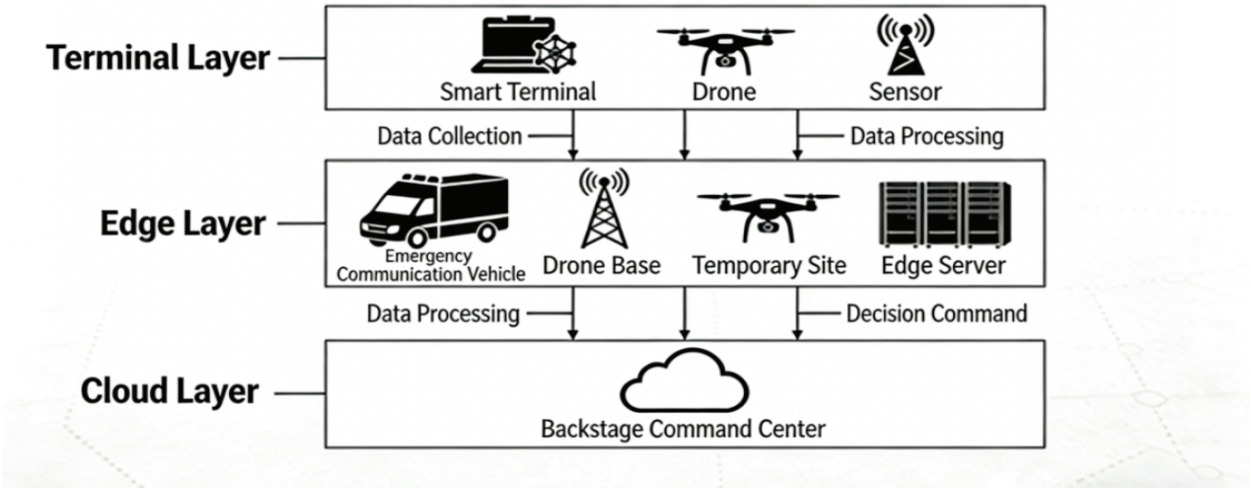
the network link [2]. Once the backbone network is interrupted or the base station is damaged, the entire system will be paralyzed. Therefore, edge and cloud have their own advantages, but they cannot independently meet the dual needs of high concurrent access and high reliable continuous service in emergency scenarios. Cloud-side collaboration is the main way to solve this problem [3]. This paper takes low-latency response as the design core, designs and optimizes from the cloud-edge collaborative architecture, mainly solves the fault tolerance problem under the conditions of node failure and resource fluctuation, and designs an emergency communication system with adaptive scheduling of task load fluctuation.

2. Emergency Communication System Architecture Design

2.1 Three-tier Collaborative Overall Architecture

According to the main requirements of emergency communication low-latency response, this paper mainly proposes a three-tier collaborative architecture of end,edge and cloud. Each layer has clear responsibilities and cooperates with each other to lay the foundation for low latency from the architecture. The terminal layer completes data acquisition and preliminary processing, and uses a lightweight protocol to achieve local rapid response; the edge layer is the main computing node, which completes tasks such as real-time analysis, task scheduling, and nearby decision-making, which greatly shortens the distance of data transmission. The cloud layer performs global resource co-ordination, complex task processing, model training, etc., and can achieve elastic expansion and task takeover when the edge nodes are overloaded or failed. The three layers rely on dynamic link quality perception and adaptive offloading to achieve coordination, ensuring that tasks are performed with the lowest delay in the most suitable place. In a simulated weak-signal area with a radius of 500 meters where fixed edge node coverage is absent, a simulation experiment deploying a single UAV as an aerial edge node was conducted. The experimental results show that, for 200 delay-sensitive tasks (e.g., real-time video analysis), adding this UAV-assisted edge computing reduces the average task response time by approximately 49.6% (from 1.45 seconds to 0.73 seconds)compared to the scenario relying solely on a cloud center via a long-distance, low-bandwidth backup link. This solves the problem of incomplete coverage of fixed edge nodes and enhances the ability of the architecture to guarantee low latency in extreme environments. which solves the problem of incomplete coverage of fixed edge nodes and enhances the ability of the architecture to guarantee low latency in extreme environments. As shown in Figure 1:

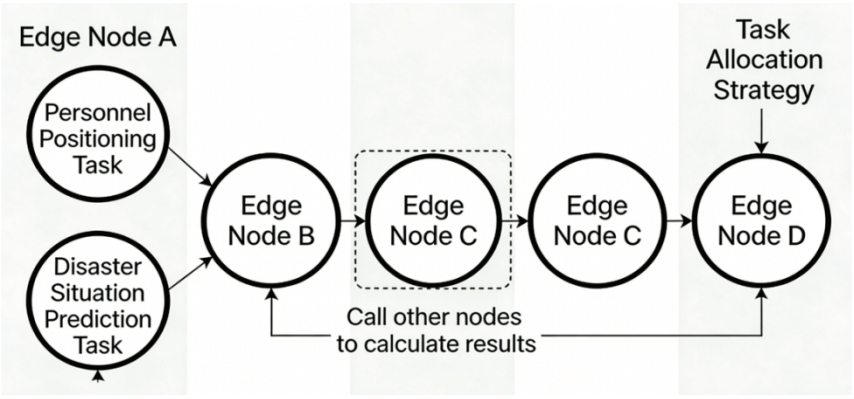
Figure 1: Three-tier collaborative overall architecture



2.2 Task Replication Scheduling Mechanism

In the emergency communication scenario, the dependencies between tasks are more complex [4], and the data transmission time is one of the main factors that cause low delay response. In order to overcome this core bottleneck, this architecture mainly uses the task replication scheduling strategy. The basic idea is to exchange computing for communication, that is, to a certain extent, to increase the consumption of computing resources, so as to achieve the purpose of greatly reducing communication delay. As shown in Figure 2.

Figure 2: Task replication scheduling mechanism



In a simulation with 10 edge nodes and 400 randomly generated tasks with complex dependencies, we compared a 'task replication'strategy (with a maximum replication factor of 3 for critical tasks) against a 'no replication'baseline. The experimental results show that a reasonable task replication strategy can increase the system's response speed by 1.45 times, i.e., reduce the average task completion delay by approximately 31% (from 820 ms to 566 ms).However, it is necessary to take into account the constraints of resources, and the competition of resources between edge nodes cannot be intensified due to excessive replication, thereby increasing the delay and achieving the balance between 'low latency'and 'resource efficiency.

2.3 Dynamic Resource Allocation Mechanism

In the emergency communication scenario, the task load has a strong suddenness and a high degree of unpredictability. In the early stage of the disaster, the field data will flow to the system in a very short period of time, and the task load will increase exponentially. If the system uses a fixed resource allocation method and cannot make corresponding adjustments according to the real-time state, it is easy to cause the computing resources of some edge nodes to be depleted instantly and overloaded, and the front-end backhaul link becomes very congested due to a sudden increase in traffic, which leads to a significant increase in task processing delay and has a serious impact on the timeliness of emergency response [5].

Therefore, this architecture designs a dynamic resource allocation algorithm, which does not rely on static preset strategies to work, it is based on the continuous perception and timely response of the system state. The algorithm uses distributed monitoring components to obtain the main state indicators such as computing resource utilization rate, task queue length, residual energy level, packet loss rate of communication links between nodes, bandwidth occupancy, and round-trip delay of each edge node. Nodes with small computing load, good link quality, and sufficient energy are first sent to high real-time or high-priority tasks. For nodes with load greater than the threshold or unstable links, tasks are pulled to adjacent available nodes to prevent local resource constraints from causing the overall system performance to decline. The algorithm adds a mechanism that relies on short-term load trends to predict, and makes resource reservation and task pre-distribution in advance before the peak load, thereby enhancing the forward-looking and smoothness of scheduling decisions. The system uses the closed-loop control mechanism of resource awareness and scheduling execution to control the response delay within an acceptable range under the emergency situation of large fluctuations in task load, and provides strong resource support for low-latency response. The main design points and quantitative results are shown in table 1:

Table 1. Dynamic resource allocation core elements and effect table

decisive variables	core aim	Specific allocation strategy/mechanism description	Quantitative benefits/description
Core optimization objectives	Minimize task response delay	As the highest priority of resource allocation, all decisions are made around reducing task waiting and processing time.	—
Key factor one	Current load rate of node	Prioritize task allocation to nodes with load rate<60% to avoid processing bottlenecks caused by high-load nodes.	Achieve load balancing to prevent a single point of overload

decisive variables	core aim	Specific allocation strategy/mechanism description	Quantitative benefits/description
Key factor two	The amount of data to be transmitted	Assign to local or adjacent nodes Assign to local or adjacent nodes to reduce data transmission time.	Reduce network transmission delay
Key factor three	link quality	Delay-sensitive tasks are preferentially allocated to nodes with link packet loss rate <5% to ensure transmission stability.	Ensure high reliability transmission
Key factor four	Node energy consumption state	Avoid assigning to nodes with energy consumption less than 20%, and prevent the node from failing in the middle of the task due to the exhaustion of power consumption.	Improve system robustness and reduce task interruption risk
auxiliary mechanism	Load forecasting and resource reservation	Based on historical data, combined with deep learning methods to predict future task arrival patterns, resource reservation is performed in advance.	The accuracy of resource allocation is increased by more than 35%.
comprehensive effect	Delay control and energy consumption optimization	Through the above strategy combination, the resource utilization efficiency is optimized while ensuring low latency.	The energy consumption of edge nodes is reduced by about 20%.

3. Low-latency Task Scheduling Optimization Algorithm

3.1 Problem Modeling

The task scheduling problem in the emergency communication system is to minimize the task response delay while satisfying the resource constraints and task dependence constraints. In this paper, the problem is modeled as: given the emergency task set , there is a clear dependency between tasks, and the computing node set, including edge nodes and cloud nodes. Each task can be assigned to one or more edge nodes to perform. The goal is to minimize the overall task response delay of the system, taking into account the system energy consumption and resource overhead. The optimization objective is , where f_T represents the average task completion time, f_E represents the total energy consumption of the system, and f_R represents the resource overhead caused by task replication. Constraints include node computing power constraints, communication bandwidth constraints, task dependency constraints, and node energy consumption constraints. This problem is a NP-hard problem. When the task size is larger than 300, the exact optimal solution can not be solved in polynomial time. Designing efficient heuristic algorithms to achieve near-optimal scheduling under low-latency targets becomes the key.

3.2 Task Scheduling Algorithm Based on Improved Ant Colony Optimization

Ant colony optimization algorithm has good adaptability and optimization ability in task scheduling problem by simulating the path optimization behavior of ant foraging [6]. However, its standard version has some shortcomings, such as slow convergence speed, easy to fall into local optimum, and low delay demand in emergency scenarios. In this paper, according to the characteristics of emergency communication scenarios, the standard ant colony algorithm is improved, and a task scheduling optimization algorithm for low latency is designed. The core improvement points focus on low latency targets, as follows:

(1) Pheromone update strategy optimization: In the traditional ant colony algorithm, the pheromone update rules of all paths are consistent, which cannot reflect the priority difference of emergency tasks. In this paper, the task priority factor is introduced, and the tasks are classified according to the emergency priority,such as life rescue-related tasks,as the highest priority. The high priority tasks on the critical path will obtain higher pheromone enhancement coefficients, and guide the subsequent 'ants'to assign high quality edge nodes to high priority tasks to ensure the low delay execution of core tasks.

(2) Heuristic function design optimization: The expression of heuristic function can be expressed as:

$$\eta(i, j) = \alpha \cdot (1/(\text{load}_r\text{ate}(j))) + \beta \cdot (1/(\text{transmission}_d\text{elay}(i, j))) + \gamma \cdot (1/(\text{replication}_t\text{imes}(i)))$$

Where α, β, γ are the weight coefficients, and β is the highest priority to ensure low delay.

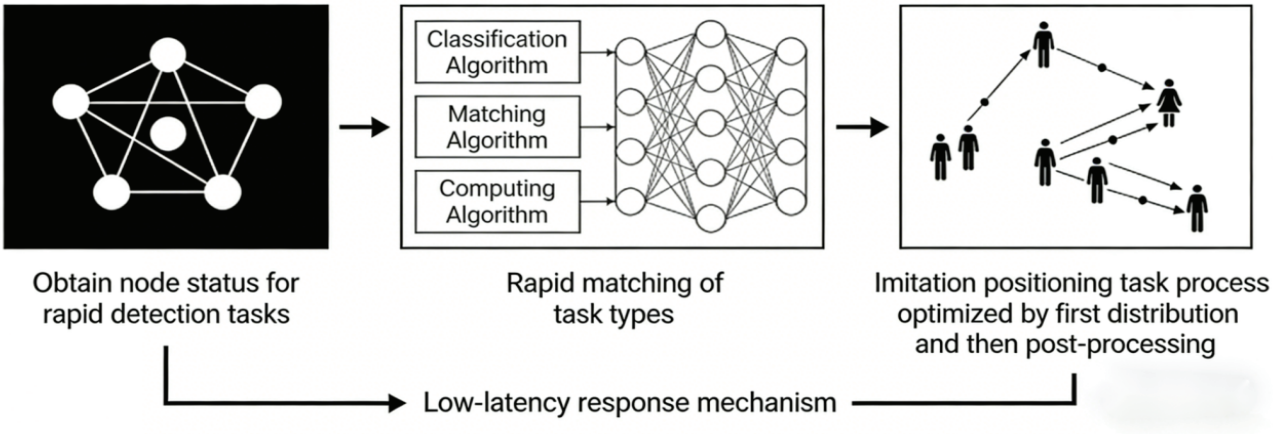
(3) The improvement of node selection mechanism. Theε-greedy algorithm is used to select nodes, which considers both the accuracy of optimization and the speed of convergence. In the first 30%iterations of the algorithm,εis set to be 0.8, and a new

path is randomly explored with high probability to prevent falling into local optimum in the early stage. In the later 70%of the iteration times of the algorithm, the number of iterations is gradually reduced fromεto 0.2, the probability of greedy selection is increased, and the approximate optimal solution is quickly converged to ensure that the scheduling algorithm can quickly respond to the sudden demand of emergency tasks.

3.3 Cold Start Optimization Strategy

In the early stage of the emergency scenario, the system has not accumulated enough historical data such as node load and link quality. The scheduling algorithm will encounter the problem of cold start, which is easy to cause unreasonable task allocation, thereby increasing the response delay, and ensuring that the system can achieve low-latency response at the beginning. The effect is shown in Figure 3:

Figure 3: Cold start optimization strategy of emergency scenario



4. Key Technology Implementation and Efficiency Analysis

4.1 Lightweight Containerized Deployment

This architecture regards the Docker container as the basic unit of application encapsulation and deployment. Compared with the virtual machine, in a comparative test on a standard edge node(4 vCPUs,8GB RAM), the average startup time for a Docker container running a lightweight emergency data processing application was measured at 350 ms, which is more than 80% faster than a traditional KVM-based virtual machine (which took over 2.1 seconds to boot the same application). Concurrently, the container's idle resource consumption was over 60% lower than that of the virtual machine. This allows for rapid deployment and migration of emergency applications, providing basic support for low-latency response. Container image management uses hierarchical storage to deploy the basic image to each edge node in advance. When the application is updated, only the incremental layer is transmitted, which greatly reduces the overhead of network transmission and thus reduces the delay. At the same time, the container orchestration tool is used to manage and dynamically schedule the edge node containers to ensure the rational allocation of resources.

4.2 Cross-node Data Synchronization

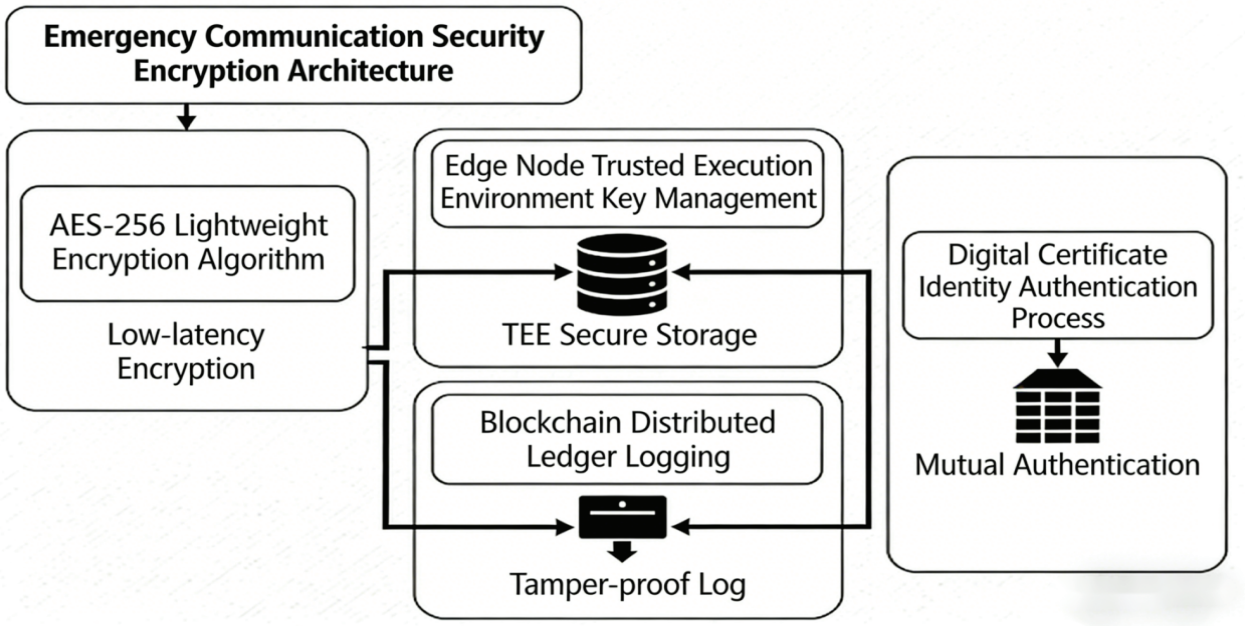
This architecture uses the final consistency model, and uses a distributed message queue (RocketMQ lightweight version) to achieve cross-node data synchronization. The key data uses synchronous replication to ensure that the data is returned after the completion of data writing. Confirmation, non-critical data uses asynchronous replication, does not need to wait for confirmation, and reduces latency overhead. For the problem of network interruption, the node adopts the method of local cache, and the data will be automatically synchronized when the network restarts, so as to prevent the repeated execution of tasks caused by data loss.

4.3 Security and Privacy Protection

Emergency communication includes sensitive data such as the location of rescuers and medical information of the wounded. The security standard is high, but encryption protection will increase the time of data transmission and processing. This architecture uses a lightweight encryption mechanism to minimize latency while ensuring security.The AES-256 lightweight encryption algorithm is used to transmit data and store data. In a benchmark test encrypting 1 MB data blocks (simulating a

typical emergency data packet), the AES-256 algorithm achieved an average encryption/decryption latency of 15 ms. This represents a delay reduction of approximately 25% compared to the widely-used but more computationally intensive RSA-2048 algorithm (which averaged 20 ms for the same operation), while maintaining a comparable level of security. As shown in Figure 4:

Figure 4: Emergency communication security architecture



4.4 Effectiveness Analysis

In order to verify the performance of the architecture, a simulation environment is built based on CloudSim Plus (v6.4.0), and a custom module for task replication and dynamic resource allocation is added. In the experiment, the hardware configuration is : 1 cloud center (32 vCPU, 64 GB RAM) and 10 edge nodes (4 vCPU, 8 GB RAM), the network baseline bandwidth is 100 Mbps, the round-trip delay between the cloud and the edge is 10 ms, the round-trip delay between the edge nodes is 5 ms, and the cloud backup link delay is set to 50 ms in the UAV weak signal simulation. In terms of task model, each simulation randomly generates 500 emergency tasks, and the dependency relationship is represented by a directed acyclic graph. The task size is between 10 MI and 500 MI, and the task arrival obeys a Poisson process with an average of 50 / s to simulate disaster bursts. In the fault scenario, the ‘ node failure ‘ test will randomly select three edge nodes at the 30 th second of the simulation run to make them fail. The comparison baselines include : pure cloud architecture (all tasks are offloaded to the cloud center), pure edge architecture (tasks are only processed locally at the edge), and cloud-edge collaboration without task replication (based on load scheduling but not copying tasks). The evaluation indicators are : average task response delay (the average of 30 runs, given a 95% confidence interval), system reliability (the proportion of tasks completed before the deadline, the deadline is set to 2 times the average completion time under a pure cloud architecture), average node load rate (average CPU utilization of all edge nodes during the simulation), and resource overhead (additional computing time consumed by task replication, measured in CPU seconds).. The comparison objects are three typical schemes: pure cloud architecture, pure edge architecture and cloud edge collaboration architecture without task replication. The performance of cloud edge collaboration with cloud centralized processing, edge independent processing and no task redundancy mechanism under actual emergency conditions is investigated respectively. The evaluation index is mainly considered from three aspects: task response delay, system reliability and resource overhead. Task response delay reflects the degree of satisfaction of the architecture for low latency requirements. System reliability is used to measure the ability of task completion in the case of node failure. Resource overhead is used to measure the computing and communication costs paid by the architecture in order to achieve low latency and high reliability. The above settings are used to compare the comprehensive performance of the architecture in different comparison objects, different task loads,and fault scenarios.

Table 2. Experimental comparison table

Scenario	Evaluation Metric	Proposed Architecture	Cloud-only Architecture	Edge-only Architecture	Cloud-Edge Collaboration without Task Replication
Node Failure (3 edge nodes down)	Average Task Response Delay	Baseline (≈40% reduction)1060ms→636ms	≈120% higher1400ms	≈73% higher1100ms	Baseline1060ms
	System Reliability (On-time Completion Rate)	≥96%	72%	80%	88%
Normal Condition (Fault-free, 200 tasks)	Communication Delay Reduction vs. Cloud-only	≥30%	—	—	—
	Computing Resource Overhead (vs. No Replication)	+15% ~ +20%	—	—	—
Large-scale Concurrency (500 tasks)	Average Node Load Rate (Balance)	≤60% (Balanced Load)	≤20% (Centralized Bottleneck)	>85% (Local Overload)	70% ~ 90% (Uneven Distribution)
	Task Response Delay (P99)	≤500ms	>1500 ms (Severe Backlog)	>1200 ms (Resource Contention)	>900 ms (High Latency)

5. Conclusions and Prospects

Based on the rigid requirements of emergency communication for low-latency response, this paper proposes an emergency communication architecture with end,edge and cloud collaboration. From the perspective of task deployment, computing resources are used to exchange communication efficiency, and dynamic resource allocation is used to achieve real-time matching of node load, link quality and energy consumption status. In the aspect of scheduling optimization, improved ant colony optimization algorithm and cold start strategy are added to ensure that the system can quickly generate a near-optimal scheduling scheme in the early stage of emergency without historical data accumulation. From the perspective of engineering implementation, lightweight containerized encapsulation and cross-node data synchronization technology are used to improve the deployment efficiency of applications and the consistency of multi-copy data, providing support for the actual landing of the architecture. In the future, deep reinforcement learning will continue to be used to improve the adaptability of scheduling strategies, expand the network coverage in extreme environments on the basis of multi-UAV collaboration, and use federal learning to ensure data privacy while achieving multi-node collaborative modeling. Continuously improve the system's intelligent scheduling capabilities and security service levels in complex emergency scenarios.

Funding

No

Conflict of Interests

The authors declare that there is no conflict of interest regarding the publication of this paper.

Reference

[1] Zhang, W., Miao, H. (2023). Cloud-edge-collaboration-based flexibility scheduling strategy considering communication and computation delay. CSEE Journal of Power and Energy Systems, 11(4): 1858 - 1869.

[2] Kambala, G. (2024). Emergent architectures in edge computing for low-latency application. International Journal Of Engineering And Computer Science, 13(09): 26597-26607.

[3] Obenobe, I. O. (2025). Hybrid Edge–Cloud Collaboration for Addressing Bandwidth Limitations and Latency in IoT

Applications. Faculty of Natural and Applied Sciences Journal of Computing and Applications, 2(4), 95-104.

[4] Bachoumis, A., Andriopoulos, N., Plakas, K., et.al. (2021). Cloud-edge interoperability for demand response-enabled fast frequency response service provision. IEEE Transactions on Cloud Computing, 10(1), 123-133.

[5] Sundaram, R., Thangavel, S., Narukulla, K. (2022). Edge-Enabled Distributed Computing for Low-Latency IoT Applications: Architectures, Challenges, and Future Directions. International Journal of Emerging Research in Engineering and Technology, 3(1), 28-41.

[6] Patel, C. M. (2024). Edge computing for low-latency loT applications in smart cities. Smart internet of things, 1(4), 282-288.