UDC 004. 056.55:004.312.2 **doi: 10.26906/SUNZ.2025.3.122-127**

Eduard Malokhvii

National Technical University “Kharkiv Polytechnic Institute”, Kharkiv, Ukraine

ADAPTIVE FILTERING AND DYNAMIC COMPUTATION OFFLOADING FOR RESILIENT TASK EXECUTION IN IIOT

Abstract. Relevance. The execution of time-sensitive tasks in Industrial Internet of Things (IIoT) systems requires decentralized and fault-tolerant computing models that can operate under dynamic workloads, unstable node availability, and limited resources. **Object of research:** task management and offloading processes in edge–fog IIoT environments. **Purpose of the article.** Development of a method for decentralized, latency-aware task processing using adaptive filtering and dynamic computation offloading, which ensures high availability, efficient resource usage, and responsiveness in distributed edge systems. **Research results.** The paper presents the AFDCO method, which integrates local data filtering, node availability evaluation, latency-based offloading, and lightweight redundancy into a unified framework. This approach enables autonomous and fault-resilient task execution without relying on centralized controllers. Simulation results confirm that AFDCO reduces response time, improves task deadline compliance, and minimizes network and energy overhead. **Conclusions.** Compared to static or centralized task allocation models, the proposed method demonstrates better adaptability and robustness under variable conditions by dynamically adjusting execution and replication strategies based on system feedback. Scope of application of the obtained results: distributed IIoT systems, edge–fog computing platforms, and low-latency industrial automation scenarios requiring decentralized execution and fault recovery.

Keywords: Industrial Internet of Things, Edge Computing, Task Offloading, Redundancy, Latency-Aware Scheduling, Fault Tolerance, Adaptive Filtering, Decentralized Systems.

Introduction

The rapid expansion of the Internet of Things (IoT) has led to an unprecedented increase in the number of interconnected devices and sensors. These devices generate massive amounts of data, which traditionally have been processed in centralized cloud servers [1–3]. However, this approach introduces significant latency issues due to the physical distance between data sources and data processing centers. As real-time data processing becomes increasingly critical for applications such as autonomous vehicles, smart cities, and healthcare monitoring, the limitations of cloud computing have necessitated the development of more efficient data management solutions [4, 5]. IoT edge computing has emerged as a viable alternative, offering a decentralized approach where data is processed closer to its point of origin, thereby reducing latency and improving response times [6, 7].

The evolution of the Industrial Internet of Things (IIoT) has revolutionized the way data is generated, transmitted, and processed across modern industrial infrastructures. With the proliferation of smart sensors, actuators, and intelligent control systems, industrial environments now produce vast volumes of heterogeneous, latency-sensitive data that must be processed in real time [1-2]. Traditional cloud-based processing paradigms struggle to meet the stringent timing and availability requirements of such systems due to their reliance on remote data centers, which introduce inherent network delays, potential bandwidth congestion, and limited reliability during connectivity disruptions [3].

To overcome these limitations, edge and fog computing architectures have emerged as decentralized models that enable computational tasks to be offloaded and executed closer to data sources [4]. Edge nodes deployed near the physical endpoints, reduce response latency, alleviate network loads, and support real-time analytics for mission-critical operations such as predictive maintenance, industrial automation, and remote supervision [5]. Fog nodes, situated between the edge and cloud, provide an intermediate layer for coordination, caching, and fallback computation. While this architectural shift significantly improves system responsiveness, it introduces new challenges related to task allocation, fault tolerance, and decentralized decision-making in highly dynamic and resource-constrained environments.

Maintaining system availability and reliability under conditions of partial node failure, resource instability, and unpredictable network performance remains a critical challenge for IIoT deployments. Many existing task scheduling and offloading methods depend on centralized control or assume static system behavior, which makes them prone to single points of failure and limits their applicability in scalable, distributed industrial settings [6]. In addition, most solutions do not incorporate latency sensitivity or perform real-time evaluation of node availability, both of which are essential for consistent operation under variable workloads and changing environmental conditions [7].

To address these challenges, we propose a novel latency-aware and decentralized task management method designed specifically for edge–fog–cloud environments in IIoT systems. The proposed method integrates dynamic task offloading, adaptive backup allocation, and localized decision-making to enhance service continuity and processing efficiency. Each node independently evaluates its own availability, resource load, and communication latency, enabling it to select the most suitable execution paths for incoming tasks. A redundancy-aware model ensures that backup execution plans are in place in case of primary node failure, while the decentralized coordination protocol eliminates dependence on global control entities.

This paper presents the design, mathematical modeling of the proposed method, followed by simulation-based evaluation in a representative IIoT setting. The contributions of this work are relevant for industrial applications requiring high availability, low latency, and scalable computing strategies under real-world deployment conditions.

Related Works

The increasing complexity and scale of IIoT deployments have intensified the need for efficient task management frameworks capable of addressing latency sensitivity, system availability, and fault tolerance in resource-constrained and dynamic environments. Cloud-centric architectures, though powerful in terms of computation and storage, are often unsuitable for real-time applications due to long transmission delays, congestion, and lack of localized control [7]. Edge computing has emerged as a promising alternative, enabling data processing closer to the source and offering better responsiveness, bandwidth efficiency, and context-awareness [8].

Despite these advantages, edge environments pose new challenges such as limited computational capacity, energy constraints, heterogeneous hardware, and vulnerability to failure. Ensuring stable task execution under such conditions requires decentralized, lightweight, and fault-resilient mechanisms. Static or centrally managed systems often fall short in dynamic IIoT settings, particularly where mobility, connectivity disruption, or partial node failure are present [9].

To enhance system responsiveness and availability, various studies have proposed collaborative task execution through dynamic group formation among edge nodes. These approaches typically assign subtasks to multiple nearby nodes within execution clusters, which improves load balancing and resource utilization [10]. However, many of these techniques rely on centralized orchestration or assume a known and stable network topology, reducing their flexibility in mobile and failure-prone environments.

Fault tolerance is often addressed through task replication, where primary and backup nodes are statically assigned. While this enhances availability, static redundancy schemes frequently lead to inefficient resource usage, high energy consumption, and increased communication overhead. Moreover, they often do not account for the real-time availability or reliability of nodes, limiting their effectiveness in edge environments where conditions change rapidly [11].

Fog computing has been proposed as an intermediate layer between edge and cloud to support caching, aggregation, and fallback execution. Such architectures improve scalability and reduce the need for central coordination, but they often depend on periodic synchronization or rely on semi-centralized management entities, which introduce latency and complexity [12].

Recent efforts have introduced availability-aware and redundancy-controlled task scheduling. These approaches use metrics such as task priority, node failure probability, and resource availability to optimize offloading and backup selection [13]. However, most of them still rely on predefined thresholds or global state, and few implement adaptive redundancy levels that change based on observed network or node conditions.

Cluster-based coordination has also been investigated to improve stability under node mobility and intermittent connectivity. These solutions form temporary execution clusters based on proximity or link quality, enabling stable communication. Yet, they typically do not integrate tightly with runtime scheduling logic or support real-time failover [14].

In summary, existing works often address latency optimization, fault tolerance, or decentralization in isolation. Centralized models suffer from scalability limitations and single points of failure, while static redundancy schemes fail to adapt to runtime conditions. Decentralized approaches frequently lack latency awareness or coordinated backup planning. To overcome these gaps, this paper introduces a fully decentralized task management method that combines adaptive data filtering, dynamic node availability scoring, latency-sensitive offloading, and real-time redundancy adjustment. The proposed method operates without global coordination, enabling robust, responsive, and scalable task execution across heterogeneous IIoT systems.

Proposed Method

The proposed method, Adaptive Filtering and Dynamic Computation Offloading (AFDCO) is designed to support low-latency and highly available task execution in decentralized IIoT environments. It operates within a multi-layer architecture composed of sensor-level devices, edge nodes, and fog/cloud servers. Unlike traditional cloud-centric systems or statically replicated edge models, AFDCO emphasizes autonomy, responsiveness, and resilience through localized decision-making, real-time evaluation of execution conditions, and adaptive redundancy mechanisms. Each component of the method contributes to efficient task management in dynamic settings. Adaptive data filtering at the device level reduces computational and network load. Node availability evaluation allows the system to prioritize stable and underutilized nodes. Latency-aware offloading ensures timely task execution by selecting the most suitable execution path. The redundancy mechanism provides localized failover recovery without centralized coordination. Finally, runtime adaptability enables the system to continuously reconfigure itself in response to workload fluctuations and node behavior changes.

A. Adaptive Data Filtering

To minimize the volume of redundant or non-critical data entering the processing pipeline, AFDCO applies adaptive filtering at the data generation level. This mechanism is particularly important in IIoT scenarios with high-frequency telemetry and constrained communication channels. Each device or edge node evaluates incoming data points using a threshold-based filter:

$$d_t \in T_i \Leftrightarrow f(d_t, \phi_t) = true,$$

where d_t is a data point at time t, and ϕ_t is a dynamically adjusted threshold reflecting the current system state. The filtering function determines whether the data point should trigger task creation based on its deviation from expected patterns, the criticality of the source, and the node’s current load.

The threshold ϕ_t increases in response to elevated queue lengths, network congestion, or high CPU utilization. Under stable or low-load conditions, the threshold is relaxed, allowing more data to pass through. This dynamic behavior ensures that task generation adapts to the system's ability to process and deliver results in real time. By filtering data early, AFDCO reduces transmission delays, conserves bandwidth, and prevents processing units from being overwhelmed by non-urgent computations. This mechanism plays a crucial role in maintaining overall system responsiveness, especially under bursty traffic or degraded network conditions.

B. Node Availability Evaluation

To ensure that tasks are assigned to the most reliable and capable nodes, AFDCO includes a decentralized mechanism for evaluating node availability in real time. This component enables each node to make informed scheduling decisions based on the operational health and workload status of itself and its neighbors, without requiring global system knowledge.

Each node computes an availability score A_j for every candidate node N_j using a weighted model:

$$A_j = \alpha \cdot U_j + \beta \cdot (1 - Q_j) + \gamma \cdot (1 - F_j) ,$$

where U_j represents the normalized uptime ratio of node N_j over a defined observation window, Q_j denotes the current task queue utilization ($Q_j \in [0,1]$), F_j is the recent failure rate, calculated as the ratio of failed or expired tasks to total assigned tasks, $\alpha, \beta, \gamma \in [0,1]$ are application-specific weight coefficients satisfying $\alpha + \beta + \gamma = 1$.

This scoring model reflects both short-term fluctuations and long-term reliability trends. Nodes with higher scores are interpreted as more available, stable, and responsive, making them more favorable for task execution or backup assignment.

Each node maintains a local state table that includes availability scores for itself and neighboring nodes. These scores are updated periodically based on internal monitoring and lightweight status exchanges. This decentralized update mechanism avoids the overhead and fragility associated with centralized state synchronization.

The availability score plays a critical role in two decision-making stages. First, it influences the selection of execution nodes during task offloading, balancing latency performance with node reliability. Second, it is used to rank backup candidates, ensuring that redundancy mechanisms rely on nodes with a low probability of failure and sufficient spare capacity.

By quantifying node stability and workload pressure in real time, this component enables AFDCO to dynamically adapt task placement strategies to the current system context. As a result, task distribution becomes both resilient to failures and efficient in terms of resource utilization.

C. Latency-Aware Task Offloading

AFDCO incorporates a decentralized offloading mechanism that selects the most suitable execution node for each task based on expected latency and availability. This ensures that computational tasks are completed within application-specific deadlines while maintaining system responsiveness under dynamic conditions.

When a new task T_i is generated, the originating node N_k estimates its local processing delay using:

$$L_i^k = \tau_{comm}^{i,k} + \tau_{proc}^{i,k} ,$$

where $\tau_{comm}^{i,k}$ is the expected communication delay and $\tau_{proc}^{i,k}$ is the estimated time to process the task locally. If L_i^k exceeds the task's latency constraint, the node initiates offloading by evaluating a subset of candidate nodes N_j. Each candidate is scored using a composite suitability function:

$$S_i^j = \theta \cdot \left(1 - L_i^j / L_{\max}\right) + (1 - \theta) \cdot A_j ,$$

where L_i^j is the estimated latency on node N_j, A_j is the availability score of node N_j, $L_{\max}$ is the maximum tolerated latency for normalization, $\theta \in [0,1]$ controls the trade-off between latency sensitivity and reliability.

The node N_p with the highest score S_i^p is selected as the primary executor. To increase fault tolerance, a backup node N_b is also selected if it satisfies the proximity condition:

$$| S_i^p - S_i^b | < \delta ,$$

where δ is a redundancy margin threshold. This condition ensures that the backup node is sufficiently comparable to the primary in terms of responsiveness and stability.

This latency-aware offloading scheme enables AFDCO to avoid bottlenecks, bypass overloaded or unstable nodes and maintain compliance with real-time execution deadlines. The decision process is lightweight and fully decentralized, requiring only local estimation and limited peer communication.

Moreover, the offloading logic naturally adapts to network conditions. When local latency is low, tasks are executed without offloading. As latency or queue depth increases, the system shifts execution to more suitable nodes, balancing load and reducing the probability of deadline violations.

D. Redundancy and Localized Recovery

To ensure continuity of task execution in the presence of partial node failures or disconnections, AFDCO incorporates a lightweight redundancy mechanism with localized failover control. This mechanism improves system availability without requiring centralized orchestration or global state monitoring.

For each task T_i, once the primary execution node N_p is selected, a backup node N_b is assigned based on its suitability score and its proximity to the primary. The backup remains in standby mode, maintaining a minimal copy of the task context, including task parameters, metadata, and priority level. No active processing is performed on the backup unless failure is detected on the primary path.

Failure detection relies on timeout-based monitoring. After dispatching a task to N_p, the origin node or an organizer node expects periodic acknowledgments confirming task receipt and execution progress. If no acknowledgment is received within a predefined window τ_{fail}, the backup node is activated and resumes execution from the preserved task context.

This decentralized failover process reduces recovery latency and eliminates the need for complex checkpointing or rollback protocols. The replication strategy adapts to runtime conditions by selecting backups dynamically using availability scores, rather than through static assignments. Unlike traditional high-availability models that reserve resources continuously for backup tasks, AFDCO's on-demand redundancy preserves edge resources by activating backups only when necessary. This is particularly important in constrained IIoT environments where computational and energy budgets are limited. Additionally, if the original primary node recovers after failover, conflict resolution is handled through task versioning and coordination between nodes. The task instance with the most recent execution state is retained, and duplicate execution is avoided.

Overall, the redundancy and recovery mechanism in AFDCO ensures robust operation in unstable and failure-prone networks. It allows tasks to be completed despite hardware faults, connectivity loss, or overload, thus supporting high application-level availability across distributed edge–fog infrastructures.

E. Runtime Adaptability

AFDCO is designed to operate effectively under the variable and unpredictable conditions typical of industrial IoT environments. To this end, the method incorporates runtime adaptability, allowing its key components to dynamically reconfigure in response to system state changes without requiring external intervention or centralized control.

All critical operational parameters within AFDCO are adaptive. The filtering threshold , used to determine the relevance of incoming data, is continuously updated based on CPU load, queue length, and task urgency. This enables the system to prioritize critical inputs during peak demand periods while reducing overhead during stable conditions. Similarly, the availability score weights α, β, γ, which influence node selection, can be rebalanced over time to reflect changing performance characteristics or evolving reliability patterns. For example, in scenarios with increased task failure rates, more emphasis can be placed on historical success metrics to avoid overloading unstable nodes.

The offloading strategy, governed by the trade-off parameter θ, is also adjustable at runtime. When low-latency execution becomes more critical than node reliability (e.g., during alarm conditions), θ can be increased to prioritize speed over availability. Conversely, in degraded network conditions, lowering θ shifts preference toward stable execution paths.

In terms of redundancy, the threshold δ that governs backup selection can be tightened or relaxed depending on observed failure trends or energy constraints. This ensures that resources are not wasted on unnecessary replication under stable conditions, while maintaining fault coverage during periods of instability.

The adaptability of AFDCO is achieved through localized monitoring and feedback. Each node independently tracks system conditions and tunes its behavior accordingly, enabling a fully decentralized form of self-optimization. No global coordination is required, and system-wide adaptation emerges from the aggregation of local decisions.

This runtime flexibility allows AFDCO to maintain high levels of responsiveness, reliability, and efficiency even as network topology, workload intensity, or device availability change over time. It ensures that the system remains resilient and effective in both normal and adverse operating conditions, making it well-suited for real-world IIoT deployments.

Results and Discussion

To evaluate the performance of the proposed AFDCO method, a series of simulations were conducted in a virtual IIoT environment modeled after a hierarchical edge–fog–cloud architecture. The simulation environment included 72 edge nodes with heterogeneous CPU and memory configurations, 8 organizer nodes responsible for group coordination, and 1 central cloud server. End devices generated time-sensitive tasks at variable intervals, simulating dynamic workloads common in industrial monitoring and control system. Each edge node was initialized with resource parameters (CPU: 2–8 cores, RAM: 2–16 GB) randomly selected from a defined distribution to reflect real-world heterogeneity. Node failure events were introduced probabilistically, with availability rates ranging from 60% to 100%, to test the method's fault resilience. The network latency between nodes was modeled using a Gaussian distribution with a mean of 15 ms and a standard deviation of 5 ms. Each experiment was repeated 20 times, and average results were recorded. The performance of AFDCO was compared against three baseline strategies: cloud-only task execution, edge execution without redundancy, and static redundancy assignment with no latency awareness. All models were evaluated using the same workload profiles and network conditions. Metrics of interest included average response time, task success rate, network utilization, and energy consumption.

Results demonstrated that AFDCO significantly reduces application latency by minimizing unnecessary data transmission and enabling localized execution. While cloud-based models experienced delays exceeding 6000 ms under high task loads, AFDCO consistently maintained latency below 1500 ms, even with multiple concurrent sources. This improvement is attributed to adaptive filtering at the device level, which limits non-critical task generation, and latency-aware offloading, which prioritizes nodes in optimal conditions. The performance trend is illustrated in Fig. 1. In terms of reliability, AFDCO achieved a higher task deadline success rate under varying system loads and node failure rates. Even with 30% of edge nodes intentionally deactivated, AFDCO maintained a deadline compliance rate above 90%, outperforming the baseline methods. This resilience is made possible by continuous evaluation of node availability and dynamic backup selection based on real-time system feedback, as shown in Fig. 2.

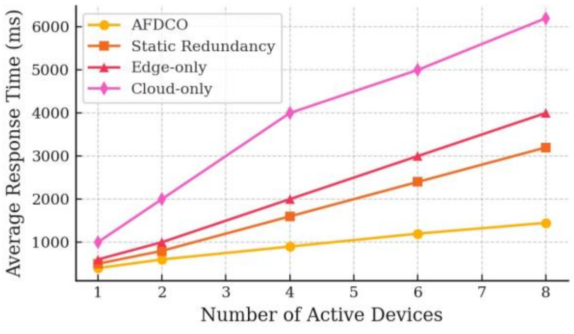

Fig. 1. Average response time under increasing task load

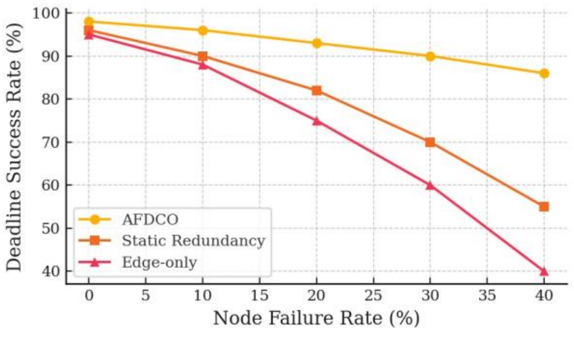

Fig. 2. Deadline success rate under varying node availability

Network utilization was also reduced due to early-stage data suppression and selective task replication. Compared to static redundancy models, AFDCO lowered overall data transfer volume by up to 30%, while cloud-only systems generated more than twice the traffic. This reduction is illustrated in Fig. 3. Additionally, the reduced communication overhead contributed to lower energy consumption, as backup nodes in AFDCO remain inactive unless triggered by failure, avoiding unnecessary standby power use common in static high-availability systems. The system exhibited strong fault recovery capabilities. As node failures increased to 40%, task success rates remained above 85% without requiring centralized coordination. AFDCO adjusted the level of redundancy dynamically, increasing replication when instability was detected and relaxing it under stable conditions, thereby optimizing resource usage. This behavior is demonstrated in Fig. 4.

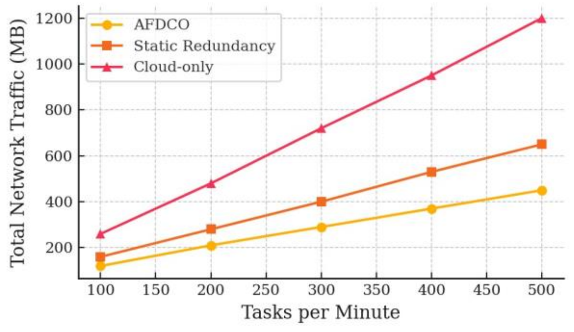

Fig. 3. Network traffic volume across task generation rate

Overall, the experimental results confirm that AFDCO provides a balanced trade-off between responsiveness, fault tolerance, and resource efficiency, making it suitable for deployment in decentralized IIoT systems where latency constraints, dynamic environments, and limited resources must be addressed simultaneously.

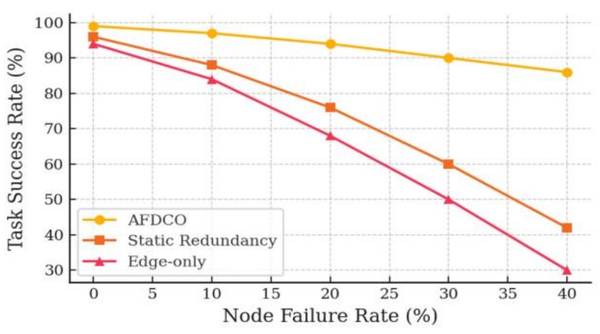

Fig. 4. Task success rate under increasing node failure rate

Conclusion

The research presents AFDCO, a decentralized method for adaptive filtering and dynamic task offloading in edge–fog–cloud environments for industrial IoT systems. The method addresses key challenges associated with latency sensitivity, fluctuating resource availability, and fault tolerance in highly distributed and dynamic networks. By integrating locally applied data filtering, real-time node evaluation, latency-aware task allocation, and lightweight redundancy mechanisms, AFDCO enables efficient and resilient execution of tasks without relying on centralized control.

Unlike traditional cloud-based or statically replicated edge models, AFDCO empowers each node to autonomously evaluate its workload, availability, and response time to make informed scheduling decisions. The adaptive filtering component significantly reduces the amount of non-essential data entering the system, helping to avoid congestion and allowing nodes to prioritize critical computations. The task offloading mechanism selects execution candidates based on a composite score that balances latency estimates and availability metrics, ensuring that each task is processed in the most suitable location with minimal delay.

To improve reliability, AFDCO introduces dynamic backup assignments and decentralized failover management. Rather than relying on pre-allocated resources, the system continuously re-evaluates backup candidates during runtime. Backup nodes remain idle until failure is detected on the primary path, allowing for rapid recovery without unnecessary energy consumption or communication overhead. This design enables high task completion rates even under node failure scenarios, as demonstrated in experimental evaluations. The results show that AFDCO consistently outperforms baseline configurations in terms of application latency, deadline compliance, fault resilience, and resource efficiency. In scenarios with high task concurrency or intermittent edge availability, the method maintained low response times and sustained execution success rates above 85%, while reducing network traffic and energy usage. These findings highlight the method's suitability for real-world IIoT deployments, where responsiveness, reliability, and scalability are simultaneously required.

Future research will aim to enhance AFDCO by incorporating predictive models that utilize historical workload patterns to improve scheduling accuracy under uncertainty. Additional focus will be placed on supporting node mobility and context-aware task migration in highly dynamic topologies. The method may also be extended to operate across federated fog infrastructures, enabling coordination between multiple administrative domains. Finally, deploying AFDCO on physical edge hardware and evaluating its performance under real industrial workloads will provide deeper insight into its practical applicability, particularly in terms of real-time responsiveness, energy efficiency, and integration with existing IIoT platforms.

References

1. R. P. Singh, M. Javaid, A. Haleem, and R. Suman, “Internet of things (IoT) applications to fight against COVID-19 pandemic Diabetes & Metabolic Syndrome”, Clinical Research & Reviews, vol. 14, no. 4, pp. 521-524, 2020, doi: https://doi.org/10.1016/j.dsx.2020.04.041
2. C. Gong, F. Lin, X. Gong, and Y. Lu, “Intelligent Cooperative Edge Computing in Internet of Things”, IEEE Internet of Things Journal, vol. 7, no. 10, pp. 9372-9382, 2020, doi: https://doi.org/10.1109/JIOT.2020.2986015
3. L. Cui, S. Yang, Z. Chen, Y. Pan, Z. Ming, and M. Xu, “A Decentralized and Trusted Edge Computing Platform for Internet of Things”, IEEE Internet of Things Journal, vol. 7 no. 5, pp. 3910-3922, 2019, doi: https://doi.org/10.1109/JIOT.2019.2951619
4. M. S. Elbamby, C. Perfecto, C. F. Liu, J. Park, S. Samarakoon, X. Chen, and M. Bennis, “Wireless Edge Computing with Latency and Reliability Guarantees”, Proc. IEEE, vol. 10 (8), pp. 1717-1737, 2019, doi: https://doi.org/10.48550/arXiv.1905.05316
5. H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, and C. T. Lin, “Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment”, IEEE Access, vol 6, pp. 1706-1717, 2017, doi: https://doi.org/10.1109/ACCESS.2017.2780087
6. A. Javed, K. Heljanko, A. Buda, and K. Främling, “CEFIoT: A Fault-Tolerant IoT Architecture for Edge and Cloud”, 2018 IEEE 4th world forum on internet of things (WF-IoT), pp. 813-818, 2018, doi: https://doi.org/10.48550/arXiv.2001.08433
7. G. Premsankar, M. D. Francesco, and T. Taleb, “Edge Computing for the Internet of Things: A Case Study”, IEEE Internet of Things Journal, vol. 5, no. 2, pp. 1275-1284, 2018, doi: https://doi.org/10.1109/JIOT.2018.2805263
8. W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, “A Survey on the Edge Computing for the Internet of Things”, IEEE Access, vol. 6, pp. 6900-6919, 2017, doi: https://doi.org/10.1109/ACCESS.2017.2778504
9. Y. Zhang, X. Chen, Y. Chen, Z. Li, and J. Huang, “Cost Efficient Scheduling for Delay-Sensitive Tasks in Edge Computing System”, 2018 IEEE Int. Conf. on Services Computing, pp. 73-80, 2018, doi: https://doi.org/10.1109/SCC.2018.00017
10. Y. Sahni, J. Cao, and L. Yang, “Data-Aware Task Allocation for Achieving Low Latency in Collaborative Edge Computing”, IEEE Internet of Things Journal, vol. 6 no. 2, pp. 3512-3524, 2018, doi: https://doi.org/10.1109/JIOT.2018.2886757
11. C. Avasalcai, C. Tsigkanos, and S. Dustdar, “Decentralized Resource Auctioning for Latency-Sensitive Edge Computing”, 2019 IEEE Int. Conf. on Edge Computing, pp. 72-76, 2019, doi: https://doi.org/10.1109/EDGE.2019.00027
12. M. Mudassar, Y. Zhai, L. Liao, and J. Shen, “A Decentralized Latency-Aware Task Allocation and Group Formation Approach With Fault Tolerance for IoT Applications”, IEEE Access, 8, pp. 49212-49223, 2020, doi: https://doi.org/10.1109/ACCESS.2020.2979939
13. S. Latre, P. Leroux, T. Coenen, B. Braem, P. Ballon, and P. Demeester, “City of things: An integrated and multi-technology testbed for IoT smart city experiments”, Proc. IEEE Int. Smart Cities Conf. (ISC), pp. 1–8, Sep. 2016, doi: https://doi.org/10.1109/ISC2.2016.7580875
14. H.-L. Truong, and S. Dustdar, “Principles for Engineering IoT Cloud Systems”, IEEE Cloud Comput., vol. 2, no. 2, pp. 68–76, Mar. 2015, doi: https://doi.org/10.1109/MCC.2015.23

Received (Надійшла) 02.05.2025
Accepted for publication (Прийнята до друку) 23.07.2025

Відомості про авторів / About the Authors

Малохвій Едуард Едуардович – аспірант, кафедра комп’ютерної інженерії та програмування, Національний технічний університет “Харківський політехнічний інститут”, Харків, Україна;
Eduard Malokhvii – PhD candidate, Department of Computer Engineering and Programming, National Technical University “Kharkiv Polytechnic Institute”, Kharkiv, Ukraine;
e-mail: malokhvii.ee@gmail.com; ORCID Author ID: http://orcid.org/0009-0008-0311-6400;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=59179728600.

Adaptive Filtering and Dynamic Computation Offloading for Resilient Task Execution in IIoT

Е. Е. Малохвій

Анотація. Актуальність. Виконання задач з жорсткими часовими вимогами в системах промислового Інтернету речей (IIoT) потребує децентралізованих та відмовостійких обчислювальних моделей, здатних функціонувати за динамічних навантажень, нестабільної доступності вузлів і обмежених ресурсів. **Об’єкт дослідження:** процеси керування виконанням та оффлоудингом задач у середовищах крайових-туманних архітектури IIoT. **Мета статті.** Розробка методу децентралізованої обробки задач з урахуванням затримки, який ґрунтується на адаптивній фільтрації та динамічному оффлоудингу обчислень, що забезпечує високу доступність, ефективне використання ресурсів і швидке реагування у розподілених системах. **Результати дослідження.** Запропоновано метод AFDCO, який об’єднує локальну фільтрацію даних, оцінювання доступності вузлів, оффлоудинг з урахуванням затримки та легкий механізм резервування в єдину децентралізовану схему. Метод забезпечує автономне та відмовостійке виконання задач без залучення централізованого керування. Результати моделювання підтверджують зниження затримок, покращення дотримання термінів виконання задач і зменшення навантаження на мережу та споживання енергії. Висновки. У порівнянні з централізованими або статичними моделями розподілу задач, запропонований метод краще адаптується до змін у системі та забезпечує стабільну роботу завдяки динамічному регулюванню виконання та реплікації. Сфера застосування отриманих результатів: розподілені системи IIoT, платформи крайових-туманних обчислень та сценарії промислової автоматизації з низькою затримкою, що потребують децентралізованого виконання та відновлення після збоїв.

Ключові слова: промисловий Інтернет речей (IIoT), кордонні обчислення, передача задач на виконання, резервування, планування з урахуванням затримки, відмовостійкість, адаптивна фільтрація, децентралізовані системи