Exploring Offloading Strategies in the Analysis of Vital Signs in Fog Computing

Thiago Henrique Thomas[1], **Rodrigo da Rosa Righi**[1]

[1]Universidade do Vale do Rio dos Sinos (UNISINOS)
São Leopoldo – RS – Brazil

thiagothomas@edu.unisinos.br, rrrighi@unisinos.br

Abstract. *This paper proposes the Decentralized Adaptive Priority-based Hierarchical Offloading (DAPHO) model for offloading strategies in fog computing applied to healthcare in smart cities. The model handles large volumes of IoT data by prioritizing critical signals in the fog while less critical ones are processed in the cloud. Unlike centralized or single-path approaches, DAPHO enables autonomous offloading decisions among fog nodes hierarchically organized (e.g., neighborhoods, cities). The flexibility of multiple destinations and dynamic thresholds improved response time and prioritized signal processing, although fixed thresholds showed better performance in some scenarios.*

1. Introduction

The Internet of Things (IoT) has transformed various domains, from Industry 4.0 to smart cities and healthcare, by enabling real-time data collection and intelligent decision-making. In healthcare, IoT devices such as smartwatches and sensors facilitate continuous monitoring of vital signs, helping in the prevention and management of critical conditions [Inzole and Sonwane 2024]. However, to ensure the effectiveness of these critical services, it is essential to minimize response time, and maintain reliability so that people are always assisted, especially in emergency situations [Das and Inuwa 2023].

Therefore, this work addresses the problem of efficient task offloading in Edge/Fog Computing environments for public healthcare and smart cities. Traditional cloud-based processing introduces latency, while existing offloading strategies often lack decentralization and multiple routes. To tackle this, we propose DAPHO (Decentralized Adaptive Priority-based Hierarchical Offloading), a model that optimizes resource allocation and prioritizes vital sign processing. Unlike previous approaches, DAPHO enables hierarchical and decentralized offloading, where Fog nodes dynamically transfer tasks to parent or sibling nodes based on workload and urgency. This enhances response times and system reliability, particularly in emergency scenarios.

2. Related Work

Several studies have explored task offloading in Edge and Fog Computing environments, focusing on optimization techniques and resource management. Reinforcement Learning (RL)-based approaches, such as [Robles-Enciso and Skarmeta 2023, Zare et al. 2023], utilize Deep RL agents to dynamically allocate tasks based on resource availability, latency, and priority constraints. These methods enhance adaptability by allowing nodes to learn from their environments, reducing reliance on static offloading strategies. Additionally, Federated Learning (FL) is employed to minimize data transmission overhead

while refining RL models locally. Other works prioritize deadline-aware scheduling and queue-based mechanisms. For instance, [Adhikari et al. 2020, Sarkar et al. 2022] propose heuristic-based task allocation to ensure time-sensitive tasks are processed in Fog nodes while deferring non-critical tasks to the cloud.

Latency and load-balancing strategies have also been extensively studied, focusing on real-time network conditions to optimize offloading decisions. For example, [Mattia et al. 2022, Phan et al. 2021] propose dynamic selection of offloading destinations based on factors such as hop distance, bandwidth, and node congestion levels. Some studies have also explored decentralized architectures for task offloading. [André Setti Cassel et al. 2024] introduces a tree-based model for prioritizing emergency cases through recursive vertical offloading.

3. DAPHO Model

In this chapter, a new model is proposed for offloading operations in fog computing environments to optimize response time and distribute the workload. The DAPHO model (Decentralized Adaptive Priority-based Hierarchical Offloading) will perform this operation in a decentralized manner on each node in the network. This enables both vertical offloading to a parent node and horizontal offloading to a sibling node within the tree structure, while also allowing dynamic resources thresholds and prioritization of both users and services to be executed

3.1. Overload Detection

To handle overloading in a fog node within the network, thresholds will be applied to the CPU and memory metrics of the nodes. This will be achieved using an algorithm inspired by the work of [da Rosa Righi et al. 2018], which introduces an approach related to TCP congestion control for dynamically adjusting thresholds, referred to as Live Thresholding. Periodic observations of resource utilization are collected, based on an adapted strategy from [da Rosa Righi et al. 2018]. These observations focus on a specific resource within a given node, applying techniques like Simple Exponential Smoothing (SES) or Weighted Moving Average.

With this approach, thresholds can now be dynamically adjusted according to the system's load variations. In the implementation of the DAPHO model, when the current system load is higher than the previous load, the Upper Threshold must be reduced, as this indicates an increasing workload. Conversely, when the current load is lower than the previous load, the workload is decreasing, allowing for an increase in the Lower Threshold. If the system load exceeds the Upper Threshold, all incoming messages will be offloaded until the overloaded resource drops below the Lower Threshold. At this point, the thresholds can be restored to their initial values. Similarly, if the system load falls below the Lower Threshold, both thresholds can be reset.

3.2. Offload Decision

To make offloading decisions, two heuristics are applied. The first heuristic consists of ranking signals based on user and service priority. If a signal's ranking exceeds 75% of all rankings on the node, it will be executed locally. If the ranking is below 25%, offloading is performed. Rankings between these thresholds result in inconclusive decisions. The

ranking calculation adapts the approach from [André Setti Cassel et al. 2024], incorporating a dynamic weight factor (γ) for users to prevent multiple offloads for the same user through time.

When the ranking heuristic results in an inconclusive decision, a second heuristic evaluates service duration using the Exponential Recency Weighted Average (ERWA) method, prioritizing more recent observations. By giving greater weight to shorter-duration services, the system ensures faster processing for time-sensitive tasks. Therefore, dynamic thresholds for CPU and memory are evaluated first, followed by the ranking-based heuristic and, if necessary, the service duration heuristic refines the decision.

3.3. Destination Node Selection: vertical or horizontal offloading

Furthermore, is necessary to select an apropriate node. This selection follows a greedy approach, identifying the node with the most available resources. Periodically, the parent node gathers resource and metric information from available child nodes. Nodes that are not busy respond with their current resource availability. This information is then relayed to the child nodes. On the overloaded node, calculations are performed to determine which destination node would result in the greatest resource growth. That way the process can either perform vertical offloading (to the parent node) or horizontal offloading (to a sibling node), continuing this process recursively until the cloud is reached.

This approach ensures that the most suitable destination node is selected based on resource growth potential. The process evaluates critical resources such as CPU, memory, and bandwidth, assigning specific importance to each resource type. The overloaded node calculates a score for each potential destination, considering both the relative growth of available resources and their assigned importance.

4. Results

The experiments demonstrated that allowing offloading to multiple destinations (both sibling and parent nodes) reduced the average execution time and increased the number of tasks processed within the fog layer compared to the cloud. When offloading included sibling nodes, the average execution time remained below 0.1 seconds for all priority levels. In this configuration, most tasks were executed within the fog, achieving rates between 98.8%, 99.1%, 99.3%, 99.7% and 99.9% across priorities 1 to 5. Conversely, without offloading to sibling nodes, the average execution time ranged between 1 and 2 seconds, and fog execution rates dropped to between 83.6%, 85.0%, 88.8%, 92.3% and 96.9% for the same priorities.

The use of dynamic thresholds showed mixed results. While they did not outperform fixed thresholds, they prevented node overload, demonstrating greater resilience under heavy workloads. However, with fixed thresholds, some nodes became overloaded toward the end of the experiments, leading to higher offloading rates. If the tests were extended, the average execution time would likely increase, and the number of tasks processed in the fog layer would reduce. Additional tests with workloads evenly distributed across all nodes showed no significant differences in performance.

5. Conclusion

This work introduces the DAPHO model, which integrates decentralized offloading and dynamic thresholds for smart cities. Tasks can be offloaded both horizontally (to sibling

nodes) and vertically (to parent nodes), enabling faster processing of high-priority vital signs—critical in healthcare scenarios where timely responses to emergencies like cardiac arrests can be life-saving. This research is part of the Amazon AWS Universities VitalSigns++ project, coordinated by Professor Rodrigo da Rosa Righi. Future directions include refining dynamic thresholds, prioritizing critical cases, incorporating latency as a decision factor, and testing in larger topologies. Additionally, reducing communication overhead through distributed algorithms like the Gossip Protocol could enhance scalability.

References

Adhikari, M., Mukherjee, M., and Srirama, S. N. (2020). Dpto: A deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing. *IEEE Internet of Things Journal*, 7(7):5773–5782.

André Setti Cassel, G., da Rosa Righi, R., André da Costa, C., Rosecler Bez, M., and Pasin, M. (2024). Towards providing a priority-based vital sign offloading in healthcare with serverless computing and a fog-cloud architecture. *Future Generation Computer Systems*, 157:51–66.

da Rosa Righi, R., Rodrigues, V. F., Rostirolla, G., André da Costa, C., Roloff, E., and Navaux, P. O. A. (2018). A lightweight plug-and-play elasticity service for self-organizing resource provisioning on parallel applications. *Future Generation Computer Systems*, 78:176–190.

Das, R. and Inuwa, M. M. (2023). A review on fog computing: Issues, characteristics, challenges, and potential applications. *Telematics and Informatics Reports*, 10:100049.

Inzole, A. and Sonwane, S. (2024). Iot in healthcare: Applications challenges. In *2024 IEEE 3rd International Conference on Electrical Power and Energy Systems (ICEPES)*, pages 1–5.

Mattia, G. P., Magnani, M., and Beraldi, R. (2022). A latency-levelling load balancing algorithm for fog and edge computing. In *Proceedings of the 25th International ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '22, page 5–14, New York, NY, USA. Association for Computing Machinery.

Phan, L.-A., Nguyen, D.-T., Lee, M., Park, D.-H., and Kim, T. (2021). Dynamic fog-to-fog offloading in sdn-based fog computing systems. *Future Generation Computer Systems*, 117:486–497.

Robles-Enciso, A. and Skarmeta, A. F. (2023). A multi-layer guided reinforcement learning-based tasks offloading in edge computing. *Computer Networks*, 220:109476.

Sarkar, I., Adhikari, M., Kumar, N., and Kumar, S. (2022). Dynamic task placement for deadline-aware iot applications in federated fog networks. *IEEE Internet of Things Journal*, 9(2):1469–1478.

Zare, M., Elmi Sola, Y., and Hasanpour, H. (2023). Towards distributed and autonomous iot service placement in fog computing using asynchronous advantage actor-critic algorithm. *Journal of King Saud University - Computer and Information Sciences*, 35(1):368–381.