Autonomous and Adaptive Scheduling Framework for Next-Generation Fog Computing Systems

Abhijeet Mahapatra [1], *Ganesh Khekare* [2]

[1]Postdoctoral Researcher, Lincoln University College, 47301, Petaling Jaya, Selangor Darul Ehsan, Malaysia; [2]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India.

Email IDs: pdf.abhijeet@lincoln.edu.my, khekare.123@gmail.com.

Abstract: Fog computing has emerged as a promising paradigm for supporting latency-sensitive Internet of Things (IoT) applications by extending Cloud capabilities closer to data sources. However, efficient task scheduling in Fog environments remains challenging due to heterogeneous resources, dynamic tasks, and strict Quality-of-Service (QoS) requirements. This work proposes an Autonomous Adaptive Scheduling Framework (AASF) that leverages Deep Reinforcement Learning (DRL) to intelligently allocate tasks across distributed Fog nodes. The framework incorporates a multi-objective optimization model that simultaneously minimizes latency, energy consumption, execution cost, and reliability risks. A comprehensive system model and analytical performance metrics are developed to guide scheduling decisions. The proposed framework aims to enable self-optimizing Fog ecosystems capable of adaptive resource management and scalable IoT service delivery.

Keywords: Autonomous and Adaptive Scheduling; Deep Reinforcement Learning; Fog-Cloud Computing; Multi-objective Optimization; Task Scheduling.

Introduction

Fog computing paradigm has emerged as an important layer for connecting the centralized Cloud systems with the edge devices by enabling data processing closer to its source. This results in addressing the issues related to high latency, limited scalability, and energy inefficiency [1]. These characteristics of Fog computing make it essential for latency-sensitive domains including healthcare monitoring, autonomous vehicles, industrial IoT systems, *etc.* [2]. However, the dynamic and resource-constrained nature of Fog environments necessitates the development of advanced scheduling mechanisms for efficient task allocation and resource utilization.

Modern IoT-Fog-Cloud environments are characterized by dynamic tasks, unstable network conditions, and frequent node failures, which makes traditional centralized scheduling approaches inadequate for real-time decision-making. Distant Cloud offloading cannot consistently ensure the low latency and hard real-time responses required by healthcare, autonomous vehicles, industrial automation, and AR/VR applications. This is made worse by the fact that Fog nodes differ greatly in terms of processing power, memory, energy capacity, and hardware capabilities, which calls for intelligent schedulers that can assign tasks independently and contextually.

The need for decentralized, self-optimizing scheduling solutions is needed as centralized control is becoming a bottleneck as the number of IoT devices increases. Fog scheduling demands adaptive real-time decision-making beyond static rule-based approaches, which must optimize multiple objectives simultaneously. Manual intervention can be greatly decreased by autonomous schedulers that actively respond to failures, learn from past choices, and continuously monitor system feedback [3]. However,

real-time responsiveness, reducing latency, and energy efficiency continue to be difficult to achieve, especially for critical real-time systems [4].

Proposed Methodology

- *System Architecture*:

  Figure 1 shows the 3—tier IoT-Fog-Cloud architecture consisting of IoT Devices, Fog Nodes and Cloud Datacenter. Fog layer includes the Autonomous Adaptive Scheduler (AASF) control panel which implements federated learning for decision making. In this architecture, IoT layer is responsible for task generation, execution and near-source aggregation are done in the Fog nodes and executing compute-intensive and non-real time tasks is carried out in the Cloud datacenter. This enables adaptive orchestration through the AASF that continuously observes system conditions and updates scheduling decisions using a learning-driven control loop.

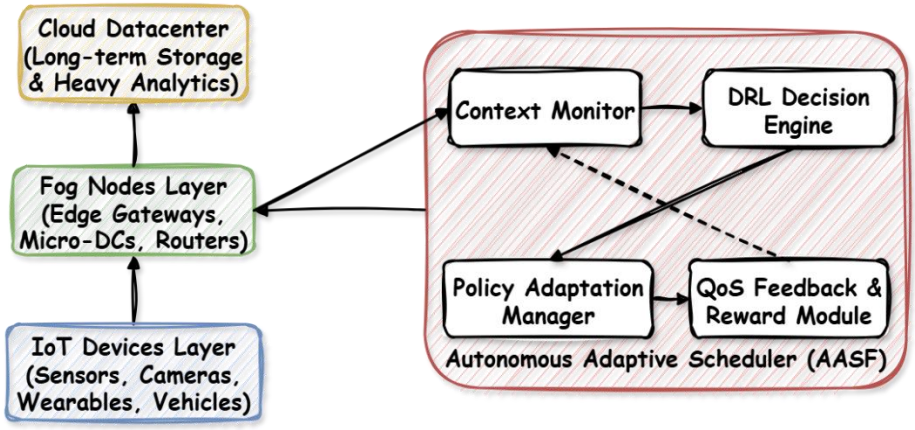

Figure 1. System Architecture

The IoT device layer is explicitly modeled as the task generator, where each task t_i is defined as $t_i = (C_i, D_i, M_i, P_i)$, where C_i: CPU demand, D_i: deadline, M_i: memory and P_i: priority. The Fog layer contains the Fog nodes, $F = \{f_1, f_2, \ldots, f_j\}$ with $f_j = (CPU_j, RAM_j, E_j, BW_j)$, where E_j: energy requirement by j^{th} Fog node and BW_j: bandwidth requirement by j^{th} Fog node. Fog layer is presented as the primary latency-sensitive execution tier, closer to data sources and thus suitable for deadline-sensitive tasks. However, Fog resources are heterogeneous and constrained; hence the scheduler must handle contention, dynamic link variability, and energy limitations, which makes static heuristics brittle.

The Cloud datacenter is assigned responsibilities that are deliberately non-real time including critical tasks, long-term storage and non-real time processing of tasks. Hence, it is treated as the back end with abundant compute/storage, but with higher communication latency and potentially higher cost. Here, Cloud layer is considered as the overflow tier for tasks that are not deadline-critical or too critical for Fog capacity.

- *Latency Modelling* [5]: The total latency experienced by a task consists of three main components, namely transmission delay, waiting/queuing delay and execution delay, which can respectively be calculated using eq. 1, eq. 2 and eq. 3:
  - Transmission Delay: Transmission delay represents the time required to transfer task data from the IoT device to the Fog node.

$$T_i^{trans} = \frac{S_i}{BW_j} \quad (1)$$

where, S_i is the input task size and BW_j is the available bandwidth of the selected Fog node. This parameter evaluates the overhead for network communication, which can significantly affect latency in bandwidth-constrained environments.

- Waiting/queuing Delay: Tasks arriving at Fog nodes may experience waiting delay if computing resources are occupied. Let the queue length at node f_j is represented by Q_j and the queueing delay can be evaluated as:

$$T_i^{wait} = \frac{Q_j}{R_j} \quad (2)$$

This delay shows resource contention in Fog nodes and is affected by task execution rate and scheduling policies.

- Execution Delay: Execution delay corresponds to the time required to process the task on the Fog node.

$$T_i^{exec} = \frac{C_i}{R_j} \quad (3)$$

Hence, the total latency experienced by task t_i can be calculated as:

$$L_i = T_i^{trans} + T_i^{wait} + T_i^{exec} \quad (4)$$

The average latency across all tasks is computed as:

$$L_{avg} = \frac{1}{N}\sum_{i=1}^{N} L_i \quad (5)$$

This is used to evaluate the overall responsiveness of the Fog computing system.

➢ *Energy Consumption Modelling* [6]: The total energy consumption of the system consists of computation energy and communication energy, calculated using eq. 7 and eq. 8, respectively:

- Computation Energy: This is modeled using a dynamic CPU power model:

$$P_{cpu} = \alpha f^3 \quad (6)$$

where, f represents CPU frequency and α is a hardware-specific coefficient.

Therefore, the energy consumed during task execution can be calculated as:

$$E_i^{comp} = P_{cpu} \times T_i^{exec} \quad (7)$$

- Communication Energy: This corresponds to the energy consumed during data transmission.

$$E_i^{comm} = P_{tx} \times T_i^{trans} \quad (8)$$

where P_{tx} represents the transmission power.

Therefore, total energy per task is represented as:

$$E_i = E_i^{comp} + E_i^{comm} \quad (9)$$

Total system energy can be represented as:

$$E_{total} = \sum_{i=1}^{N} E_i \quad (10)$$

This allows the scheduler to incorporate energy-aware optimization strategies.

➢ *Cost Modelling* [7]: In real-world Fog–Cloud infrastructures, resource utilization is associated with monetary cost. Therefore, the scheduling framework also considers execution cost.

Let c_{cpu} denote the cost per CPU second and c_{bw} denote the cost per bandwidth usage. Therefore, cost per-task and total execution cost can be calculated using eq. 11 and eq. 12 respectively:

$$Cost_i = c_{cpu} \times T_i^{exec} + c_{bw} \times S_i \quad (11)$$

$$Cost_{total} = \sum_{i=1}^{N} Cos\, t_i \quad (12)$$

The cost model ensures that the scheduler can optimize economic efficiency alongside system performance.

- *System Reliability Model* [8]: Reliability represents the probability that tasks are successfully executed without failure. Let r_j represent the probability that Fog node f_j is operational. Hence, the task success probability of executing task t_i on node f_j can be represented as:

$$P_{success}(t_i) = r_j \quad (13)$$

The overall reliability of the system is computed as the average success probability, using eq. 14:

$$R = \frac{1}{N}\sum_{i=1}^{N} P_{success}\,(t_i) \quad (14)$$

Alternatively, reliability can also be expressed using failure probability, represented in eq. 15:

$$Failure = 1 - R \quad (15)$$

This parameter ensures that scheduling decisions consider node stability and fault tolerance.

- *Problem Statement*: The problem statement formulation allows the scheduler to simultaneously optimize latency, energy efficiency, cost, and reliability, which are the key QoS requirements in Fog computing environments. The task scheduling problem is formulated as a multi-objective optimization problem. The objective is to minimize the following expected cumulative costs, represented in eq. 16:

$$J(\pi) = E\left[\sum_{t=0}^{\infty} \gamma^t \,(w_1 L_t + w_2 E_t + w_3 C_t + w_4 U_t)\right] \quad (16)$$

where, π represents the scheduling policy, γ is the discount factor controlling the importance of future rewards, L_t represents the latency component, E_t represents the energy consumption component, C_t represents the execution cost component, and U_t represents system unreliability. The coefficients w_1, w_2, w_3, w_4 represent scalar weights controlling the trade-offs between performance metrics.

The optimization problem is subject to several practical system constraints as represented in constraints c1, c2, c3 and c4.

Each task must be assigned to at most one Fog node:

$$\sum_{j=1}^{m} x_{ij} \leq 1 \quad (c1)$$

where x_{ij} is a binary decision variable.

The total resource demand of tasks assigned to a node must not exceed its capacity, ensuring that nodes are not overloaded:

$$\sum_{i=1}^{n} x_{ij}\, C_i \leq R_j \quad (c2)$$

Each task must be completed before its deadline

$$L_i \leq D_i \quad (c3)$$

Alternatively, deadline violations may be penalized in the objective function.

The binary decision variables constraints ensure feasible and practical scheduling solutions in the Fog environment:

$$x_{ij} = \begin{cases} 1, & \text{if task } t_i \text{ is assigned to node } f_j \\ 0, & \text{otherwise} \end{cases} \quad (c4)$$

- *DRL-based Adaptive and Autonomous Scheduling Framework* [8]:

The proposed AASF module shown in Figure 2 includes monitoring, decision-making, policy adaptation, and feedback.

The *Context Monitor* is responsible for sensing and continuously collecting system state signals such as, Fog utilization (CPU usage, queue length, container availability), network conditions (uplink/downlink bandwidth, RTT, packet loss, jitter), energy consumption (battery level, power draw, thermal headroom), task features (arrival rate, priority, deadline), and SLA/QoS indicators from recent executions (deadline miss rate, latency, failure rate). It converts raw data into a structured state representation for the DRL agent. Importantly, the monitor supports the "adaptive" property without accurate and timely context because the DRL engine cannot maintain stable QoS under non-stationary task.

The *DRL Decision Engine* represents the learned scheduling policy. At each scheduling iteration, it maps the observed state s_t to an action a_t, where actions typically correspond to placement/offloading decisions by assigning t_i to f_j or Cloud, admission control by accepting/dropping/deferring tasks under overload, allocation of resources, and migration decisions by moving tasks between Fog nodes. Its decisions are evaluated through QoS feedback and can be adapted via the policy manager, creating a stable learning-control loop.

The *Policy Adaptation Manager* is responsible for stability and long-term robustness under dynamic task distributions, failures, mobility and bandwidth. It is used to maintain the exploration/exploitation dynamically, triggering re-training or fine-tuning when QoS degrades, switching between policies, and re-weighing reward components based on operator priorities. This helps in making the system autonomous in a practical sense because the scheduler can self-correct and gives efficient results without manual intervention in dynamic environments.

Finally, the *QoS Feedback & Reward Module* operationalizes the scheduler's objectives by converting runtime outcomes into a scalar (or vector) reward signal. The reward is typically dependent on end-to-end latency, SLA/Deadline adherence, task success/failure, and energy/cost metrices. The output of this module, r_t is used to update the DRL policy through context monitor module. The reward module is what aligns DRL behavior with multi-objective Fog scheduling. This module helps with learning-from-consequences model rather than a one-time trained model.

The diagrammatic representation for the flowchart for the proposed AASF mechanism is shown in figure 2 and the steps are stated as follows:

Step 1: Start

Step 2: Gathers information

Step 3: Incoming tasks are registered by adding tasks to the scheduling queue, and extracting task computation demand, deadline, memory and priority.

Step 4: Collected context and task parameters are encoded into a state representation for the DRL agent is states as $S_t = [CPU_1, \ldots, CPU_n, Energy_1, \ldots, QueueLen_1, \ldots, TaskFeatures]$, where S_t: state.

Step 5: Decision engine chooses an action based on the current state, we consider $A_t = \pi(S_t)$, where A_t: action taken, π: DRL policy.

Step 6: Selected action is applied by deploying tasks to chosen Fog nodes, allocating CPU/memory and starting execution.

Step 7: System performance is evaluated with respect to latency, energy consumption, cost, task success rate and reliability.

Step 8: Calculate scalar reward using $R_t = \alpha_1 \frac{1}{Latency} + \alpha_2 \frac{1}{Energy} + \alpha_3 Reliability - \alpha_4 Penalty$

Step 9: Update DRL policy through storing transition (S_t, A_t, R_t, S_{t+1}), backpropagation and updating neural network weights.

Step 10: Control check is performed at the end.

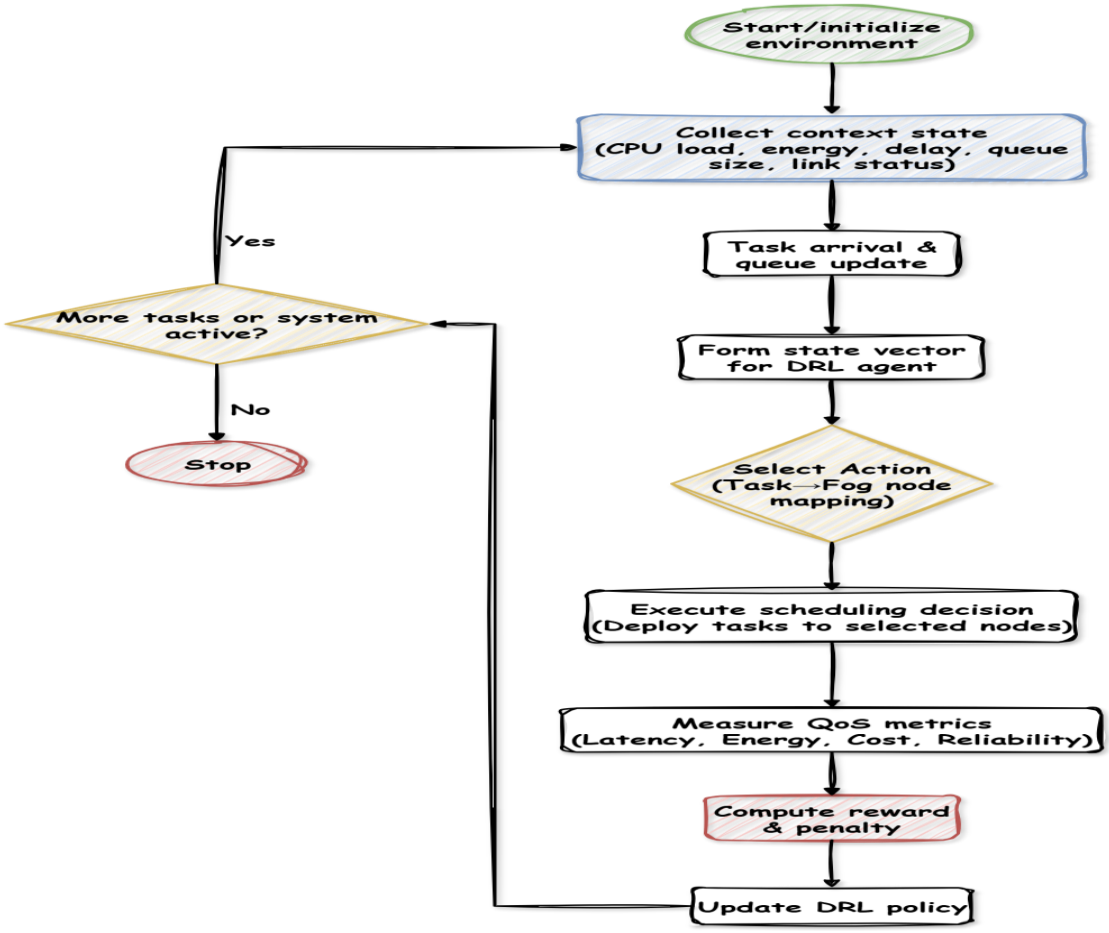

Figure 2. AASF Flowchart

Key Contribution

The main contributions of this work can be summarized as follows:

1. Proposed an Autonomous Adaptive Scheduling Framework (AASF) for Fog computing environments.
2. Developed an integrated Deep Reinforcement Learning-based task scheduling mechanism for real-time decision making and task scheduling.
3. Developed a comprehensive multi-objective optimization model for minimizing latency, energy, cost, and unreliability.
4. Constructed a detailed mathematical system model analyzing latency, energy consumption, execution cost, and reliability in Fog systems.
5. Proposed a scalable IoT–Fog–Cloud cooperative architecture enabling adaptive and decentralized resource management.

Conclusions

This article proposes an Autonomous Adaptive Scheduling Framework (AASF) that leverages Deep Reinforcement Learning (DRL) to intelligently allocate tasks across distributed Fog nodes by incorporating a multi-objective optimization model that simultaneously minimizes latency, energy consumption, execution cost, and reliability risks. Through this article we can conclude the following points while this also paves a way for various future research which can be carried out:

1. Proposed a DRL-based AAS framework for real-time adaptability of decentralized Fog nodes.
2. Integrated multi-objective optimization (latency, energy, cost and reliability) to address multiple issues in one objective.
3. Federated schedulers are used to achieve decentralized intelligence.
4. The proposed methodology is suitable for self-optimizing Fog ecosystems which are capable of dynamic resource management, energy conservation and scalability across various IoT applications.

References

1. M. Amadeo, C. Campolo, J. Quevedo, D. Corujo, A. Molinaro, A. Iera, R. L. Aguiar and A. V. Vasilakos, "Information-centric networking for the internet of things: challenges and opportunities", IEEE Network, vol. 30, no. 2, pp. 92–100, 2016. 10.1109/MNET.2016.7437030.
2. M. Kierzynka, L. Kosmann, M. vor dem Berge, S. Krupop, J. Hagemeyer, R. Griessl, M. Peykanu and A. Oleksiak, "Energy efficiency of sequence alignment tools—software and hardware perspectives", Future Generation Computer Systems, vol. 67, pp. 455–465, 2017. https://doi.org/10.1016/j.future.2016.05.006.
3. R. Mahmud, K. Ramamohanarao and R. Buyya, "Application management in Fog computing environments: A taxonomy, review and future directions", ACM Computing Surveys (CSUR), vol. 53, no. 4, pp. 1-43, 2020. https://doi.org/10.1145/3403955.
4. K. H. Abdulkareem, M. A. Mohammed, S. S. Gunasekaran, M. N. Al-Mhiqani, A. A. Mutlag, S. A. Mostafa, N. S. Ali and D. A. Ibrahim, "A review of Fog computing and machine learning: Concepts, applications, challenges, and open issues", IEEE Access, vol. 7, pp. 153 123–153 140, 2019. 10.1109/ACCESS.2019.2947542.
5. A. Mahapatra, R. Pradhan, S. K. Majhi and K. Mishra, "DELTa: Dynamic Energy-and-Latency-aware task scheduling for Fog-Cloud paradigm", IEEE Access, Vol. 13, pp. 74617–74633, 2025. 10.1109/ACCESS.2025.3563103.
6. A. Mahapatra, S. K. Majhi, K. Mishra, R. Pradhan, D. C. Rao and S. K. Panda, "An energy-aware task offloading and load balancing for latency-sensitive IoT applications in the Fog-Cloud continuum", IEEE Access, vol. 12, pp. 14334–14349, 2024. 10.1109/ACCESS.2024.3357122.
7. S. S. Mangalampalli, G. R. Karri, M. V. Ratnamani, S. N. Mohanty, B. A. Jabr, Y. A. Ali, S. Ali and B. S. Abdullaeva, "Efficient deep reinforcement learning based task scheduler in multi Cloud environment", Scientific Reports, vol. 14, no. 1, pp. 21850, 2024. https://doi.org/10.1038/s41598-024-72774-5.
8. P. Choppara and S. S. Mangalampalli, "Reliability and trust aware task scheduler for Cloud-Fog computing using advantage actor critic (A2C) algorithm", IEEE Access, vol. 12, pp. 102126–102145, 2024. 10.1109/ACCESS.2024.3432642.