Mobility-Aware Federated Offloading via Reinforcement Learning: Reliable and Energy-Fair Edge Intelligence

Woojin Jun Doyoung Kim Moon Gi Seok
Dept. of Computer Science and AI Engineering, Dongguk University
Seoul, Republic of Korea
{woojin.jun, doyoung.kim, mgseok}@dgu.ac.kr

Abstract— **Mobile agents in federated edge systems must decide not only whether to offload tasks but also how much computation to process locally before transmission. We propose a mobility-aware and energy-fair offloading framework driven by reinforcement learning (RL) with a feasibility filter. The RL policy jointly selects the offloading server and the DNN split point, while a feasibility filter excludes actions that are kinematically unsafe—based on the cosine between motion direction and the RSU bearing, the induced radial speed, and the round deadline. Energy fairness is encouraged by penalizing variance in residual battery levels across agents, sustaining long-term participation. By accounting for mobility-induced risks—frequent handovers and motion trends—our framework avoids wasted offloading attempts and improves reliability. Unlike binary or greedy heuristics, the proposed method supports fine-grained partial execution and mitigates both mobility risks and uneven energy depletion. Simulations with 150 mobile agents and 24 RSUs show clear improvements in latency, drop rate, and energy balance over Always-Local, Always-Offload, and mobility-unaware baselines. These results highlight the importance of mobility-aware and fairness-conscious offloading for reliable federated learning at the edge.**

I. Introduction

The proliferation of mobile agents such as autonomous guided vehicles (AGVs), unmanned aerial vehicles (UAVs), and connected vehicles in smart factories and intelligent transportation systems has intensified the demand for timely and energy-efficient computation at the edge. These agents often participate in federated learning (FL), where in each round they must decide whether to contribute updates and, if so, whether to process tasks locally or offload them to a nearby roadside unit (RSU). These decisions directly affect convergence speed, latency reliability, and cross-device energy balance, making efficient offloading a core challenge for practical FL deployment.

Task offloading has been widely studied as a means to overcome device limitations. Early work relied on static heuristics, greedy latency minimization, or simple signal-to-noise ratio (SNR) thresholds. These approaches remain brittle under mobility, neglecting **mobility-induced uncertainty** such as variable motion trends and frequent handovers, resulting in high task drop rates, degraded tail latency, and uneven energy depletion. Existing approaches either rely on heuristic rules (e.g., SNR thresholds, greedy latency minimization) or adopt unconstrained DRL policies. Both can be brittle in mobile offloading: heuristics ignore coverage dynamics and deadlines, whereas unconstrained DRL may propose actions that cannot finish within short contact times. We instead enforce feasibility up front with a feasibility filter and then apply a standard actor–critic policy (RL-Filter) on the reduced action set, yielding a practical controller under mobility and deadline constraints.

In this paper, we focus on the **local agent perspective** under dynamic mobility conditions and propose a **mobility-aware and energy-fair partial offloading policy** based on reinforcement learning (RL) with a feasibility filter. The RL agent jointly selects the offloading server and partition point of deep neural network (DNN) layers, enabling partial execution across device and edge. The feasibility filter excludes infeasible offloading actions that are unlikely to complete within the kinematic budget (derived from approach cosine, radial speed, and the round deadline), thereby directly reducing dropouts caused by mobility risks. To sustain long-term FL participation, energy fairness is incorporated by penalizing variance in residual battery levels across agents, ensuring that no single device is disproportionately drained. Unlike binary or greedy strategies, our framework explicitly accounts for both mobility risks and fairness while adapting to dynamic queueing and interference conditions.

The main contributions of this work are summarized as follows:

- **Problem formulation:** We formulate offloading in FL rounds as a mobility-aware partial execution problem, where feasibility is assessed via a kinematic risk model that considers the approach angle and radial speed toward the serving RSU.
- **Feasibility-constrained RL framework:** We propose an RL policy constrained by feasibility masks, which prevents unrealistic actions while adapting to non-stationary mobility and wireless conditions.
- **Energy fairness:** We introduce fairness defined as variance reduction in normalized battery consumption, balancing long-term energy usage across heterogeneous agents.
- **Performance validation:** Simulations with 150 mobile agents and 24 RSUs demonstrate significant improvements in latency, drop rate, and energy balance compared to Always-Local, Always-Offload, and mobility-unaware baselines.

The remainder of this paper is organized as follows. Section II reviews related work. Section III defines the system model, including latency, energy, mobility constraints, and fairness formulations. Section IV presents the proposed mobility-aware and energy-fair offloading framework with RL and a feasibility filter. Section V reports the simulation setup, baseline policies, and evaluation results. Finally, Section VI concludes the paper and outlines directions for future research.

II. Related Work

A. Federated Learning under Mobility

Early FL methods such as FedAvg [1] assumed static clients, but mobile deployments introduce intermittent connectivity and heterogeneous resources. A comprehensive survey [2] highlights client availability and fairness as central challenges. Nishio and Yonetani [3] studied client selection under heterogeneous conditions in mobile edge FL. Building on these foundations, our work targets mobility-aware offloading with explicit feasibility constraints and fairness objectives.

B. Split and Partial Execution

Partitioning deep neural networks (DNNs) enables collaborative device–edge execution. *Neurosurgeon* [4] pioneered split inference between mobile and cloud, while Edgent [5] extended this with adaptive right-sizing at the edge. BranchyNet [6] demonstrated the use of early exits to trade accuracy for latency. Recent efforts such as CoopFL [7] extend DNN partitioning to federated learning and cooperative offloading across heterogeneous edge servers. Our framework adapts these ideas to FL rounds and augments them with a mobility-aware feasibility filter.

C. Fairness in Offloading and FL

Energy-aware offloading policies often reduce mean consumption but can cause unbalanced drain across devices. Fairness is typically measured by Jain's index [8], and more recent works emphasize variance reduction in residual battery levels to sustain participation. Our design incorporates fairness directly into the RL reward, encouraging balanced energy usage while maintaining latency reliability.

D. Surveys and Background

Surveys on mobile edge computing [9] and reinforcement learning for wireless resource management [10] provide broader background. These works motivate the need for adaptive, RL-driven policies in wireless and edge systems. Our study adds to this body of work by combining feasibility-aware RL with fairness objectives under mobile federated learning.

III. System Models

A. Network Architecture and FL Timeline

We consider a set of N mobile agents $\mathcal{N} = \{1, \dots, N\}$ participating in federated learning (FL) rounds with a set of edge servers $\mathcal{M} = \{1, \dots, M\}$. In each round r, agent i decides (i) whether to participate, (ii) whether to execute tasks fully locally or partially offload them, and (iii) if offloading, which server $m \in \mathcal{M}$ and split index k to select. All updates must arrive within the round deadline T_{bud} to be aggregated in the round. Candidate servers are determined by the agent's current position and proximity to RSUs.

B. Workload Model and Partition Granularity

Each workload is modeled as a DNN of K layers with computation cost $C_{i,\ell}$ and feature size $S_{i,\ell}$. Selecting partition index k executes layers $1{:}k$ locally and layers $(k{+}1){:}K$ at server m, where $k = 0$ denotes full offload (raw input upload) and $k = K$ denotes full local execution. The compute-based partition ratio is

$$\alpha_i(k) = \frac{\sum_{\ell=1}^{k} C_{i,\ell}}{\sum_{\ell=1}^{K} C_{i,\ell}}, \quad 0 \le \alpha_i \le 1,$$

where $k = 0$ means full offload and $k = K$ denotes full local execution.

C. Latency Model

For agent i with local CPU frequency f_i and server m with service rate F_m:

$$T_{\text{loc}}^{(i)}(k) = \sum_{\ell=1}^{k} \frac{C_{i,\ell}}{f_i}.$$

Transmission times are

$$T_{\text{tx}}^{(i,m)}(k) = \frac{U_{\uparrow}^{(i)}(k)}{R_{\uparrow}^{(i,m)}}, \qquad T_{\text{dl}}^{(i,m)}(k) = \frac{U_{\downarrow}^{(i)}(k)}{R_{\downarrow}^{(i,m)}},$$

where $R_{\uparrow,\downarrow}^{(i,m)}$ are uplink/downlink rates and $U_{\uparrow,\downarrow}^{(i)}$ are payload sizes.

Remote latency is

$$T_{\text{rem}}^{(i,m)}(k) = T_{\text{tx}}^{(i,m)}(k) + T_{\text{que}}^{(i,m)} + \sum_{\ell=k+1}^{K} \frac{C_{i,\ell}}{F_m} + T_{\text{dl}}^{(i,m)}(k), \tag{1}$$

with $T_{\text{que}}^{(i,m)}$ the queueing delay.

Total completion time:

$$T_{\text{off}}^{(i)}(k,m) = T_{\text{loc}}^{(i)}(k) + T_{\text{rem}}^{(i,m)}(k).$$

D. Mobility Model and Feasibility Constraint

Let $\mathbf{p}_i \in \mathbb{R}^2$ and $\mathbf{p}_m \in \mathbb{R}^2$ be the positions of agent i and RSU m, respectively. Define the relative vector $\mathbf{r}^{(i,m)} = \mathbf{p}_i - \mathbf{p}_m$, distance $d^{(i,m)} = \|\mathbf{r}^{(i,m)}\|$, and the unit bearing from m to i as $\mathbf{u}^{(i,m)} = \mathbf{r}^{(i,m)}/d^{(i,m)}$. Let $\mathbf{v}_i$ be the agent velocity, $v_i = \|\mathbf{v}_i\|$, and $\hat{\mathbf{v}}_i = \mathbf{v}_i/v_i$.

We quantify whether the agent is *approaching* the RSU via the approach cosine

$$c^{(i,m)} = \langle \hat{\mathbf{v}}_i, \ -\mathbf{u}^{(i,m)} \rangle \in [-1, 1],$$

which is positive when moving toward the RSU. The induced radial speed is

$$v_{\text{rad}}^{(i,m)} = v_i \, c^{(i,m)}.$$

We adopt a *trend-based kinematic budget* that scales the round deadline T_{bud} according to the motion direction:

$$\tau^{(i,m)} = T_{\text{bud}} \cdot \Big(1 + \alpha\,[c^{(i,m)}]_+ \;-\; \beta\,[-c^{(i,m)}]_+\Big),$$

where $[x]_+ = \max(x, 0)$, and $\alpha, \beta > 0$ tune the gain for approaching vs. receding motions (default α=0.6, β=0.3). This formulation captures whether the agent is moving toward the RSU sufficiently fast to sustain offloading within the round deadline, without requiring explicit coverage prediction.

A candidate (k, m) is feasible if

$$T_{\text{off}}^{(i)}(k, m) \le \eta \cdot \min\{\tau^{(i,m)}, T_{\text{bud}}\}, \tag{2}$$

where $\eta \in (0, 1)$ is a safety margin (default $\eta = 0.8$). Pairs that violate (2) are filtered out and excluded from the RL agent's action space.

To obtain realistic completion times, the simulator explicitly accounts for queueing delay, interference, and dynamic uplink integration. The offloading latency is computed by integrating the transmission rate over trajectory segments, together with local and remote computation times, so that execution time closely reflects mobility-induced channel variation.

E. Energy Model

Local energy consumption:

$$E_{\text{loc}}^{(i)}(k) = \kappa_i f_i^2 \sum_{\ell=1}^{k} C_{i,\ell}.$$

Communication energy:

$$E_{\text{comm}}^{(i,m)}(k) = P_{\text{tx}}^{(i)}\, T_{\text{tx}}^{(i,m)}(k) + P_{\text{rx}}^{(i)}\, T_{\text{dl}}^{(i,m)}(k).$$

Total energy:

$$E_{\text{total}}^{(i)}(k, m) = E_{\text{loc}}^{(i)}(k) + E_{\text{comm}}^{(i,m)}(k).$$

F. Fairness Metric

Let $B_i(r)$ be agent i's residual battery at round r. Normalized battery level:

$$b_i(r) = \frac{B_i(r)}{B_i^{\max}}.$$

Fairness is measured as variance of normalized battery levels:

$$\mathcal{V}(r) = \frac{1}{N}\sum_{j=1}^{N}\big(b_j(r) - \bar{b}(r)\big)^2, \quad \bar{b}(r) = \frac{1}{N}\sum_{q=1}^{N} b_q(r).$$

G. Decision Problem

At each round, agent i selects (m, k) from the feasible set defined by (2). The RL agent receives reward

$$R_i = -\Big(w_T \frac{T_{\text{off}}^{(i)}}{T_{\text{bud}}} + w_E \frac{E_{\text{total}}^{(i)}}{E_{\text{cap}}} + w_V \Delta\mathcal{V}(r)\Big) - \phi \cdot \mathbf{1}_{\{\text{drop}\}}, \tag{3}$$

where ϕ is a large drop penalty, and the weights w_T, w_E, w_V match the simulator's `fair_reward` function.

Thus, the optimization problem is to minimize latency and energy while sustaining fairness, subject to mobility feasibility.

IV. Proposed Method

In this section, we present the proposed mobility-aware and energy-fair offloading framework. Unlike prior designs that rely on an analytical index, our approach employs a reinforcement learning (RL) agent that directly selects both the target server m and the partition index k of the DNN layers. To ensure feasibility under mobility uncertainty, a feasibility filter excludes server–partition pairs that are unlikely to complete within the round deadline or the kinematic budget $\tau^{(i,m)}$. This combination prevents unrealistic offloading attempts while allowing the RL agent to adapt to dynamic queueing, interference, and mobility conditions.

A. Mobility-Aware Feasibility Filter

For agent i considering server m and partition index k, the estimated completion time is

$$\hat{T}^{(i,m)}(k) = T_{\text{loc}}^{(i)}(k) + T_{\text{rem}}^{(i,m)}(k), \tag{4}$$

where:

- $T_{\text{loc}}^{(i)}(k)$ is the local compute latency for layers 1:k,
- $T_{\text{rem}}^{(i,m)}(k)$ is the remote compute latency for layers (k+1):K at server m,
- $T_{\text{tx}}^{(i,m)}(k)$ is the uplink transmission time of intermediate features up to split k,
- $T_{\text{que}}^{(i,m)}$ is the expected queueing delay at server m.

We replace dwell-time prediction with the kinematic budget $\tau^{(i,m)}$ defined in Section III. The feasibility condition becomes

$$\hat{T}^{(i,m)}(k) \le \eta \cdot \min\{\tau^{(i,m)}, T_{\text{bud}}\}, \tag{5}$$

When the agent is receding from an RSU (i.e., with negative approach cosine), we do not enforce a hard exclusion. Instead, the feasibility budget is *shrunk* by the trend-based gain factor, making such offloading attempts unlikely to pass the feasibility check without being categorically ruled out.

If condition (5) is not satisfied, the pair (k, m) is filtered out and excluded from the RL action space.

B. Reinforcement Learning Formulation

Each agent makes an offloading decision at the beginning of an FL round.

a) State.: The RL observation includes SINR estimates, kinematic features $c^{(i,m)}, v_i, v_{\text{rad}}^{(i,m)}, d^{(i,m)}$ for candidate RSUs, local CPU frequency f_i, residual battery $B_i(r)$, and queueing/drop EMAs.

b) Action.: The action space is

$$a_i = (m, k), \quad m \in \{0, \ldots, M\}, \quad k \in \{0, \ldots, K\},$$

including a special token for local execution.

c) Reward.: The reward function follows the definition in (3), combining latency, energy, and fairness terms with a strong penalty ϕ for dropped updates.

d) Update.: The policy is trained with an actor–critic algorithm (A2C). The feasibility filter ensures that only feasible actions are passed to the policy, while the critic evaluates long-term performance considering latency, energy, fairness, and drop penalties.

C. Connection to Federated Learning

In federated learning, dropped updates not only waste local resources but also directly harm global convergence by reducing the number of contributions per round. By explicitly penalizing drops in (3) and constraining actions with (2), our framework aligns per-agent offloading decisions with the long-term goal of reliable and fair FL training.

D. Workflow

At the beginning of each FL round:

1) For each agent i, candidate pairs (k, m) are evaluated using (4).
2) Feasibility filter (2) excludes infeasible choices.
3) The RL policy samples an action $a_i = (m, k)$ from the feasible set.
4) Selected tasks are executed; latency, energy, and success outcomes update the reward (3).
5) Successful updates are transmitted and aggregated by the server.

This design ensures that mobility-awareness and fairness are embedded into each decision while preserving adaptability under non-stationary conditions.

V. Experimental Results

A. Simulation Setup

We evaluate the proposed framework via simulation with mobile agents traveling along a road segment covered by multiple RSUs (edge servers). Agents follow random trajectories across overlapping RSU coverage regions. All updates must be completed within the round deadline T_{bud} to be included in FL aggregation. Table I summarizes the main simulation parameters.

In our simulation configuration, the result payload size is set to zero (`result_kb=0`), hence downlink transmission and reception energy are negligible. Accordingly, the performance is dominated by uplink communication, and we omit downlink terms in our experimental evaluation.

TABLE I: Simulation Parameters.

Parameter	Value
Number of agents (N)	150
Number of RSUs (M)	24 (grid 6×4)
RSU coverage radius	300 m
DNN layers (K)	20
Round deadline (T_{bud})	3000 ms
Round period	5000 ms
Agent speed	4–7 km/h
Battery capacity	108,000 J

The spatial environment is illustrated in Fig. 1, which compares rounds 20 and 60 and shows how mobility and the SINR landscape evolve over time.

We adopt a two-phase evaluation pipeline. First, the A2C policy is trained under multiple random seeds. Then, fixed checkpoints are evaluated on held-out mobility traces with `eval_only=true` to report latency, drop, energy, and fairness metrics. The main text reports evaluation performance, while training curves are deferred to the appendix.

B. Baseline Policies

We compare the following policies:

- **Always-Local**: All computation is executed locally on the device.
- **Always-Offload**: All computation is fully offloaded to the nearest RSU.
- **Greedy-Latency**: Selects the partition index k that minimizes estimated latency only.
- **SNR-Only (Mobility-Unaware)**: Offloads if SNR is above a fixed threshold, otherwise runs locally.
- **Analytical Index**: Uses a heuristic index (latency/SNR/queue estimates) to decide partitioning, without RL.
- **RL-Filter (ours)**: An RL agent chooses server and partition index, while a feasibility filter excludes infeasible or unsafe options (e.g., moving away from the RSU or insufficient $\tau^{(i,m)}$).

C. Evaluation Metrics

We report:

- **p95 Latency**: 95th percentile completion time (ms).
- **Drop Rate**: Ratio of failed tasks (e.g., due to leaving RSU coverage or budget violation).
- **Energy**: Mean energy consumed per agent per round (arbitrary units).
- **Fairness**: Variance of normalized battery levels (lower is better).

D. Results

We report results over 300 rounds of simulation. Table II and Figs. 2–3 summarize the comparison.

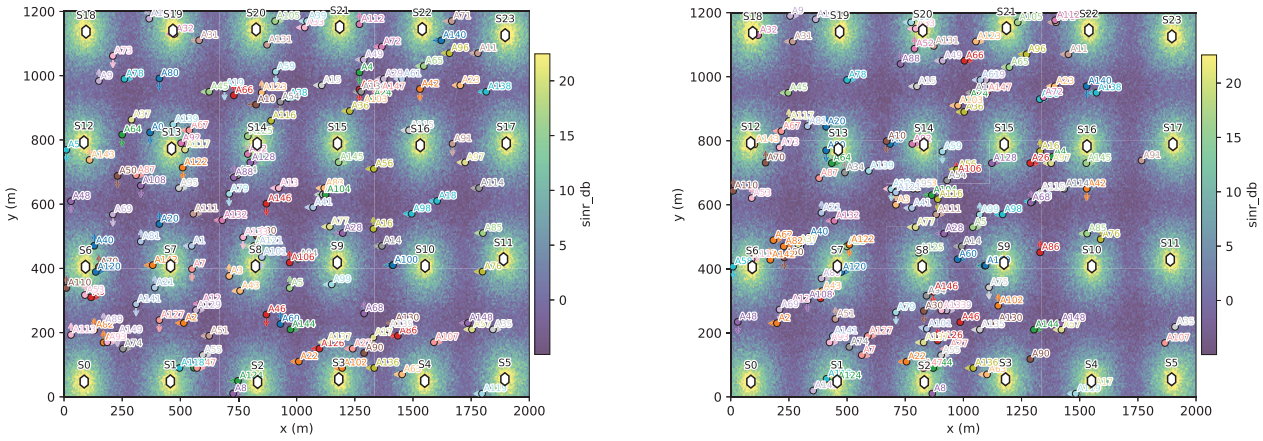
a) Latency and reliability.: RL-Filter achieves the lowest tail latency among all policies. Compared to Always-Local (4910.9 ms), RL-Filter reduces the p95 latency to 3186.9 ms, corresponding to a 35.1% reduction. It also lowers the drop rate from 12.7% to 7.1%, demonstrating that the feasibility mask effectively prevents offloads that would otherwise fail during handovers.

b) Energy and fairness.: Energy consumption is reported in raw (unnormalized) units. Always-Local consumes the highest energy (51.08), while Always-Offload uses the least (5.10) due to reduced local computation, despite high failure rates. RL-Filter consumes 29.10 on average, which is higher than Always-Offload but significantly lower than Always-Local. Importantly, RL-Filter maintains fairness with a variance of 0.0062, sustaining long-term participation.

c) Partition adaptivity.: Fig. 4 illustrates how RL-Filter distributes its choices across partition indices k, selecting earlier splits when mobility or SINR budgets are tight and deeper splits when conditions are favorable. This adaptive behavior underlies its joint gains in latency, reliability, and fairness.

E. Discussion

Two key insights emerge. First, **mobility-awareness via action filtering** is critical for tail protection and reliability:



(a) Round 20

(b) Round 60

Fig. 1: SINR heatmap with RSU and agent snapshots. Background shows `sinr_db`; red triangles denote RSUs (`S*`), and circles denote agents (`A*`). Snapshots are taken at Round 20 and Round 60.

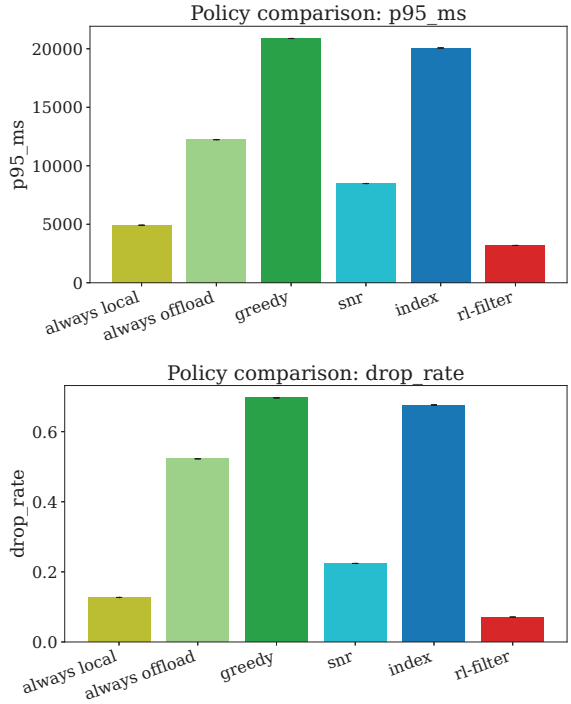


Fig. 2: Latency and reliability metrics across policies. RL-Filter achieves the lowest tail latency and drop rate.

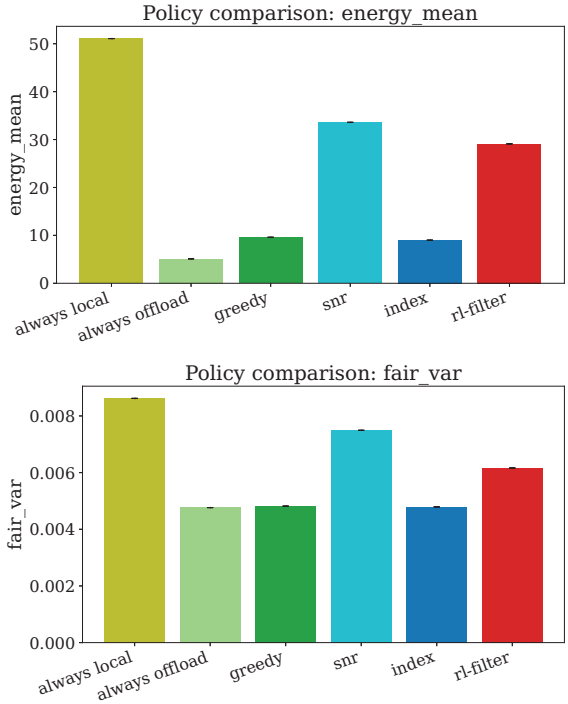


Fig. 3: Energy consumption and fairness across policies. RL-Filter reduces energy while balancing battery usage.

without a feasibility filter, offloaded tasks often fail during handovers, inflating p95 latency and increasing drop rates. By enforcing feasibility constraints, RL-Filter ensures that only actions likely to succeed are explored, directly reducing wasted transmissions. Second, the reward formulation—combining latency, energy, and fairness with strong penalties for dropped updates—encourages the RL agent to balance load across time and agents. This results in lower average energy consumption and reduced variance in battery levels, sustaining long-term participation without requiring an analytical index or handcrafted hybrid logic.

It is also worth noting that fairness objectives may introduce trade-offs with other metrics. For example, aggressively equalizing battery usage could sometimes increase latency or reduce short-term task throughput. Our results suggest that RL-Filter strikes a good balance between fairness and performance, but systematically exploring this trade-off remains an important direction for future work.

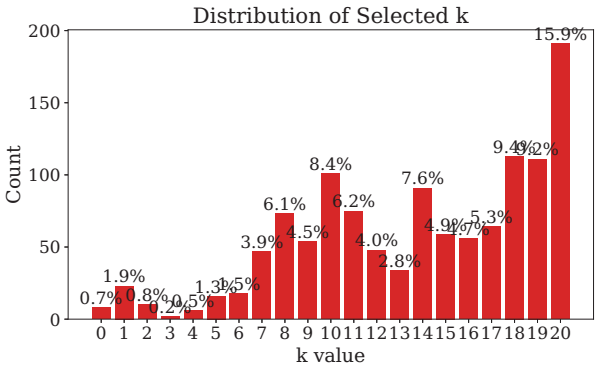


Fig. 4: Distribution of selected partition indices k under RL-Filter, highlighting adaptive partitioning behavior.

TABLE II: Corrected performance (means over 300 rounds). Energy is reported in raw units.

Policy	p95 Latency (ms)	Drop Rate	Energy Mean	Fairness Var
Always-Local	4910.9	12.7%	51.08	0.0086
Always-Offload	12236.4	52.3%	5.10	0.0048
Greedy-Latency	20879.5	69.7%	9.60	0.0048
SNR-Only	8481.8	22.4%	33.59	0.0075
Analytical Index	20066.2	67.6%	9.00	0.0048
RL-Filter (ours)	3186.9	7.1%	29.10	0.0062

Together, these effects demonstrate that mobility-aware filtering and fairness-aware rewards are sufficient to achieve robust and adaptive offloading.

VI. Conclusion

In this work, we studied mobility-aware and energy-fair task offloading for federated learning with mobile agents. Unlike generic edge offloading methods, our framework is explicitly tailored for FL rounds, where dropped updates directly harm global convergence. We proposed a reinforcement learning (RL) based policy constrained by a feasibility filter, which filters out infeasible server–partition choices that would exceed the kinematic budget $\tau^{(i,m)}$ or the round deadline. Energy fairness is embedded by penalizing variance in residual battery levels, preventing premature depletion of specific agents. This design yields an adaptive and mobility-aware offloading policy that improves reliability and sustainability.

Through simulations with 150 mobile agents and 24 RSUs, we demonstrated that RL-Filter outperforms conventional baselines in terms of tail latency, drop rate, and fairness-aware energy usage. The results highlight that explicit mobility-awareness is essential for protecting against handover-induced failures, while fairness-aware rewards sustain long-term participation across heterogeneous devices.

Future Work. Future extensions include two main directions. First, refining feasibility estimation with trajectory-aware contact-time models or map-aware predictors could improve mobility robustness. Second, validating the framework on real-world vehicular datasets and hardware testbeds will be critical to demonstrate practicality beyond simulation.

ACKNOWLEDGMENT

This research was supported by the "Regional Innovation System Education (RISE)" program through the Seoul RISE Center, funded by the Ministry of Education (MOE) and the Seoul Metropolitan Government (2025-RISE-01-007-05), and by the Institute of Information Communications Technology Planning Evaluation (IITP) grant funded by the Ministry of Science and ICT (MSIT), Republic of Korea, under the Artificial Intelligence Convergence Innovation Human Resources Development program (IITP-2025-RS-2023-00254592) and the IITP-ICAN (ICT Challenge and Advanced Network of HRD) program (IITP-2025-RS-2023-00260248).

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, ser. Proceedings of Machine Learning Research, vol. 54. PMLR, 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[2] P. Kairouz, H. B. McMahan, B. Avent, and et al., "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[3] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proceedings of IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[4] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proceedings of the 22nd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. ACM, 2017, pp. 615–629.

[5] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proceedings of the ACM SIGCOMM 2018 Workshop on Mobile Edge Communications (MECOMM'18)*. Budapest, Hungary: ACM, 2018, pp. 31–36.

[6] S. Teerapittayanon, B. McDanel, and H. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2464–2469.

[7] T. Liu, Z. Zhang, R. He, F. Liu, and C. Xu, "Coopfl: Accelerating federated learning with dnn partitioning and offloading in heterogeneous edge computing," *Computer Networks*, vol. 228, p. 109510, 2023.

[8] R. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," Digital Equipment Corporation, Hudson, MA, Tech. Rep. DEC-TR-301, 1984. [Online]. Available: https://www.cse.wustl.edu/ jain/papers/fairness.htm

[9] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[10] N. C. Luong, D. T. Hoang, D. T. Gong, D. Niyato, P. Wang, and Y.-C. Liang, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.