Bidirectional Communication-Aware Offloading for Federated Prompt Learning at the Edge

Sumiko Miyata
School of Engineering
Institute of Science Tokyo
Tokyo, Japan
sumiko@ict.eng.isct.ac.jp

Takamichi Miyata
Faculty of Advanced Engineering
Chiba Institute of Technology
Chiba, Japan
takamichi.miyata@it-chiba.ac.jp

Irfanullah Khan
School of Computing Science
The University of Glasgow
Glasgow, UK
irfanullah.khan@glasgow.ac.uk

Paul Harvey
School of Computing Science
The University of Glasgow
Glasgow, UK
paul.harvey@glasgow.ac.uk

***Abstract*—Federated learning (FL) has been increasingly applied to smart city applications to enable distributed model training while preserving data privacy. However, conventional weight-sharing FL still incurs high communication costs in bandwidth-constrained edge environments. Recently introduced Federated Prompt Learning (FPL) significantly reduces communication bandwidth usage to alleviate this issue. Nevertheless, performance degradation persists due to the coexistence of inference and training tasks and *communication overhead*, defined as delays arising from the transfer of tasks and associated data. Managing priorities between latency-sensitive inference and delay-tolerant training tasks is a critical challenge.**

This paper models the offloading of inference and training tasks in FPL environments using queuing theory. Compared to the Task-Type Separation (TTS) method, we expand the control model and propose an improved offloading scheme that minimizes the mean sojourn time of training tasks while meeting delay requirements for inference tasks. This scheme allows for bidirectional offloading among multiple edge servers, reducing inference latency while maintaining training continuity. Analytical evaluations show that the proposed method satisfies inference delay requirements while maintaining acceptable training delays. This provides an automatic, scalable task control mechanism for FPL-based edge systems.

***Index Terms*—Federated Prompt Learning, Edge Computing, Task Offloading, Smart Cities, Queueing Theory, Priority Scheduling**

I. Introduction

In recent years, there has been a growing interest in constructing digital twins of entire cities using large-scale point cloud data [1]. As an increasing number of sensors and cameras are being deployed throughout urban areas, city-level service architectures can be envisioned. However, when such systems are implemented in a city context, the underlying network infrastructure is often insufficient, and the available bandwidth is limited [2]. Even under such resource-constrained conditions, Federated Learning (FL) remains a promising framework for enabling distributed learning while preserving data privacy [3]. In FL, each client device trains a local model and transmits the updated model weights to a cloud server, where they are aggregated to improve global performance. However, since FL requires transmitting entire model weights, it introduces substantial communication bandwidth usage, which becomes a major bottleneck in bandwidth-limited edge environments [4].

To overcome this limitation, Federated Prompt Learning (FPL) has been proposed [5], [6]. Instead of updating and transmitting the entire model, FPL shares only the optimized *prompts* used during training. This approach drastically reduces the communication cost: PromptFL [5], which targets image recognition tasks, has been reported to reduce the communication volume to approximately one hundredth of that in conventional FL. In contrast, FedPrompt [6], applied to natural language processing, achieves a reduction of about one ten-thousandth. By employing FPL, the communication cost associated with model exchange, the dominant bottleneck in federated learning, is significantly reduced.

However, in edge environments that handle large-scale data (e.g. point clouds) under limited bandwidth, delays arising from the need to transfer the task and associated data that occur during task offloading (*communication overhead*), remain non-negligible [7], [8]. In addition, because latency-sensitive inference tasks and delay-tolerant training tasks coexist, their priority levels must be carefully considered for faireness and progress [9]. Under such resource and operational constraints, it is difficult to process all tasks on a single edge server, and distributing or offloading tasks among multiple edge servers becomes a practical solution. To rigorously evaluate the delay characteristics of such edge environments, queueing theory is an effective analytical tool, as it can explicitly model task arrival rates, service times, and priority relationships [10]. Queueing-based analysis enables lightweight yet theoretically sound performance evaluation without relying on complex simulations or large-scale testbeds.

Most existing studies are based on this analytical framework [11]–[14]. Representative approaches include Queueing-based Task Assignment (QTA) [11], Delay-aware Sequential Offloading (DSO) [12], and our previous work, Task-Type Separation (TTS) [13]. Their characteristics are summarized in Table I. QTA addresses edge and task assignment across multiple servers without considering task heterogeneity, DSO considers communication overhead and offloading of low-priority tasks, while TTS handles offloading of both inference and training tasks. However, none of these methods can jointly optimize communication overhead and task prioritization, which remains a key limitation in existing research.

In this paper, we extend the TTS framework to propose

TABLE I: Comparison of existing works and our approach.

	Considering task assignment	Considering comm. overhead	Offloading low-priority task	Offloading high-priority task
QTA [Chin+ 2020] [11]	✓	–	–	–
DSO [Wang+ 2025] [12]	✓	✓	✓	–
TTS [Miyata+ 2025] [13]	✓	–	✓	✓
Ours	✓	✓	✓	✓

a task offloading scheme that explicitly considers communication delay in Federated Prompt Learning (FPL) environments. Specifically, the objective is to minimize the mean sojourn time of training tasks while satisfying the delay constraint of inference tasks. The offloading control problem is formulated based on queueing theory, from which the optimal offloading strategy is analytically derived. Furthermore, through comparison with the Delay-aware Sequential Offloading (DSO) method, analytical results demonstrate that the proposed scheme completely satisfies the inference delay constraint while keeping the training delay within an acceptable range.

The main contributions of this paper are as follows:

- We extend the TTS framework to explicitly incorporate communication overhead when considering an application communicating a large amount of data (e.g. point cloud data) and bandwidth-limited edge environments.
- We formulate the offloading control problem for inference and training tasks based on queueing theory, and derive the optimal on/off switching strategy under delay constraints through a simple brute-force search.
- We conduct a comprehensive performance comparison with existing methods and demonstrate that the proposed scheme keeps inference delay within a sustainable range while maintaining training performance, thereby clarifying the advantages and limitations of each approach.

The remainder of this paper is organized as follows. Section II reviews related studies on task offloading in edge environments. Section III presents the proposed extension of the TTS framework. Section IV introduces the analytical model based on queueing theory and evaluates task offloading performance considering communication overhead and priority control. Section V compares the proposed method with existing approaches. Finally, Section VI concludes the paper.

II. Related Work

Federated Learning and Federated Prompt Learning. Federated Learning (FL) has been widely adopted as a privacy-preserving machine learning framework for distributed environments [3], [4]. However, since FL transmits the entire model weights during aggregation, it requires significant communication bandwidth, particularly in bandwidth-limited edge environments. To address this issue, Federated Prompt Learning (FPL) has been proposed. FPL drastically reduces communication cost by sharing only the optimized prompts [5], [6]. Nevertheless, in FPL, analytical frameworks that account for communication delay and task prioritization have not yet been sufficiently established.

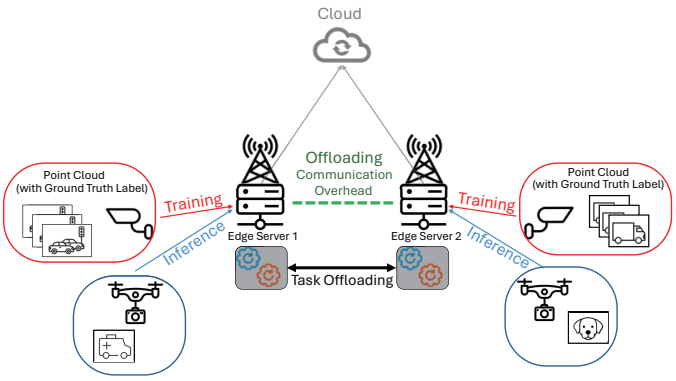

Fig. 1: System model. In Federated Prompt Learning (FPL), only optimized prompts are exchanged between the cloud and edge servers, making cloud–edge communication lightweight. Since both inference and model training are performed at edge servers, point cloud data must be exchanged between them for distributed processing. This study focuses on the communication overhead incurred by such inter-edge data transfer.

Edge Offloading and Communication overhead. In edge environments handling large-scale data such as point clouds, limited bandwidth makes communication overhead during task offloading non-negligible [7], [8]. Moreover, the coexistence of latency-sensitive inference tasks and delay-tolerant training tasks necessitates explicit consideration of task priorities [9]. Under these hardware, communication, and operational constraints, processing all tasks on a single edge server becomes impractical, and distributing or offloading tasks among multiple edge servers emerges as a necessary solution.

Queueing-Theoretic Modeling. To analytically evaluate the delay characteristics of edge environments, many studies have employed queueing theory [10]–[13]. Queueing-theoretic modeling enables rigorous evaluation of system performance with relatively low computational overhead by analytically capturing factors such as task arrival rates, service times, and priority levels, without relying on complex simulations or large-scale experimental testbeds [10].

III. Method

A. Problem Setup

Assuming system. This study targets edge environments that process large-scale point cloud and video data in real time, such as those used for digital twin construction and safety monitoring in urban areas. In such environments, numerous IoT cameras and LiDAR sensors continuously generate data. As illustrated in Fig. 1, only lightweight prompt exchanges occur between the cloud and edge servers in FPL, so cloud–edge communication cost is negligible. In contrast, when inference and model training are performed at edge servers, point cloud data must be exchanged between them for distributed

processing. Because the inter-edge link is bandwidth-limited, offloading tasks between edge servers incurs non-negligible communication overhead [15]. Therefore, an efficient inter-edge task distribution and offloading mechanism is essential. Furthermore, although each edge server is equipped with a GPU, it cannot execute multiple tasks simultaneously. Thus, inference and training tasks cannot be processed in parallel. To address this constraint, this study proposes an inter-edge offloading scheme that enables dynamic task allocation between edge servers. For analytical simplicity, we assume that both inference and training tasks arrive independently at each edge server (ES_1 and ES_2), and their characteristics are summarized as follows.

- **Inference tasks** take an image as input and output an estimated label. They have extremely short service time (e.g., a few milliseconds), very high occurrence frequency, and strict latency requirements.
- **Training tasks** take images with ground-truth (GT) class labels as input and output the trained prompts. They have extremely long service time, very low occurrence frequency, and relatively low latency requirements.
- A single edge server (with one GPU) can execute only one task at a time.

Modeling Communication overhead. Incorporating communication overhead into queueing-theoretic analysis of task offloading is generally non-trivial. Wang et al. [12] approximated the communication overhead as the total time required to transfer all task data and simply added it to the queueing and service delays. This is a reasonable assumption for inference tasks, since processing cannot begin until a complete image or input sample has been received.

However, assuming that training tasks wait for all data to be transmitted, potentially hundreds of megabytes, before any processing begins is unrealistic. In practice, training is typically performed in a mini-batch manner, where data are fetched and processed sequentially. As a result, the effective communication overhead for the training task can be approximated not by the total transmission time proportional to the data size, but by the time until the first mini-batch arrives and processing can begin, which is on the same order as that for an inference task.

Consequently, in this study, we uniformly model the communication overhead during task offloading as an initial transmission delay before task execution begins. After this delay, data transfer and computation are assumed to proceed in parallel. This assumption maintains analytical simplicity while providing a sufficiently accurate approximation of real system latency.

B. Objective Function

The objective of this study is to minimize the mean sojourn time of training tasks while satisfying the delay constraint of inference tasks. For inference tasks, a tolerable delay threshold W_{th} is introduced, and the maximum mean sojourn time over two edge servers $\mathrm{E}[W_{\text{infer}}] := \max_k \mathrm{E}[W_{\text{infer,k}}]$ is required to remain below this threshold.

TABLE II: Symbols and definitions.

Symbol	Definition
j	Index of task class ($j \in \mathcal{T} := \{\text{infer}, \text{train}\}$)
λ_j	Arrival rate of class j tasks [tasks/s]
b_j	Average service time for class j tasks [s]
$b_j^{(2)}$	Second moment of service time for class j [s^2]
k	Index of edge server ($k \in \mathcal{E} := \{1, 2\}$)
r	Proportion parameter used in $\rho_{\text{infer}} = r\rho$, $\rho_{\text{train}} = (1-r)\rho$
$\mathrm{E}[W_{j,k}]$	Mean sojourn time of class j tasks processed at ES_k
$\mathrm{E}[W_j]$	Mean sojourn time of class j tasks ,defined as the maximum average delay over two edge servers ($\max_k \mathrm{E}[W_{j,k}]$)
$S^{\text{infer}}_{(1\to2)}$, $S^{\text{train}}_{(2\to1)}$	Offloading control variables for inference ($ES_1 \to ES_2$) and training ($ES_2 \to ES_1$) tasks
$\boldsymbol{S} = \left(S^{\text{infer}}_{(1\to2)}, S^{\text{train}}_{(2\to1)}\right)$	Offloading strategy vector
$\boldsymbol{S}^*$	Optimal offloading strategy minimizing $\mathrm{E}[W_{\text{train}}]$ under $\mathrm{E}[W_{\text{infer}}] \le W_{\text{th}}$
d_{comm}	communication overhead [s]
W_{th}	Tolerable inference tasks delay threshold [s]

Accordingly, the optimization problem can be formulated as follows:

$$\boldsymbol{S}^* := \arg\min_{\boldsymbol{S}} \ \mathrm{E}[W_{\text{train}}] \quad \text{s.t.} \quad \mathrm{E}[W_{\text{infer}}] \le W_{\text{th}}, \tag{1}$$

Here,

$$\boldsymbol{S} = \left(S^{\text{infer}}_{(1\to2)}, S^{\text{train}}_{(2\to1)}\right), \quad \boldsymbol{S}^* = \left(S^{\text{infer}*}_{(1\to2)}, S^{\text{train}*}_{(2\to1)}\right),$$

where

- $S^{\text{infer}}_{(1\to2)} \in \{0, 1\}$: a control variable indicating whether edge server ES_1 offloads inference tasks to ES_2. When $S^{\text{infer}}_{(1\to2)} = 1$, offloading is performed; when $S^{\text{infer}}_{(1\to2)} = 0$, local processing is executed.
- $S^{\text{train}}_{(2\to1)} \in \{0, 1\}$: a control variable indicating whether edge server ES_2 offloads training tasks to ES_1. When $S^{\text{train}}_{(2\to1)} = 1$, offloading is performed; when $S^{\text{train}}_{(2\to1)} = 0$, local processing is executed.

Note that these offloading strategy variables are applied as a blanket policy for each traffic steady state, rather than being determined individually for each task.

In general, each edge server is denoted by ES_k ($k \in \{1, 2\}$).

Based on the objective and constraint defined in Eq. (1), all possible combinations of the strategy vector,

$$\boldsymbol{S} = \left(S^{\text{infer}}_{(1\to2)}, S^{\text{train}}_{(2\to1)}\right) \in \{0, 1\}^2,$$

the mean sojourn times $\mathrm{E}[W_{\text{infer}}]$ and $\mathrm{E}[W_{\text{train}}]$ are computed for each strategy. Among the candidates satisfying $\mathrm{E}[W_{\text{infer}}] \le W_{\text{th}}$, the strategy that minimizes $\mathrm{E}[W_{\text{train}}]$ is selected as the optimal strategy $\boldsymbol{S}^*$.

C. Modeling by using queueing theory

In this section, we employ queueing theory to analytically evaluate the impact of edge server behavior on the mean sojourn time of each task. Queueing theory enables the theoretical performance evaluation of systems without resorting to complex simulations, by probabilistically formulating task arrival rates, service times, and priority structures. This property makes it particularly effective for distributed systems such as federated prompt learning (FPL) environments, where heterogeneous tasks coexist and lightweight yet reliable analysis is required.

In this study, the system is modeled as an M/G/1 queue [16] and handles two types of tasks, *inference* and *training*. Since inference tasks require real-time responsiveness, they are assigned higher priority than training tasks.

Task arrivals are assumed to follow a Poisson process. For each class j ($j \in \{\text{infer}, \text{train}\}$), the mean arrival rate, mean service time, and second moment of the service time are denoted by λ_j [tasks/s], b_j [s], and $b_j^{(2)}$ [s^2], respectively. The traffic intensity for class j is defined as $\rho_j = \lambda_j b_j$.

This control mechanism allows the service of a training task to be *temporarily preempted* when an inference task arrives, so that the inference task can be processed immediately. This assumption reflects a realistic behavior in FPL environments, where real-time responsiveness must be guaranteed when inference requests occur.

Under this priority control, the theoretical expressions for the mean sojourn time of each task are derived in the following subsection.

D. Communication-Aware Task Offloading Model

Based on the preemptive-resume priority control described earlier, we now analytically derive the mean sojourn time of tasks under inter-edge offloading. For all strategy vectors, the mean sojourn times of inference and training tasks at each edge server ES_k are evaluated.

Strategy S=(0,0): Neither task is offloaded When no offloading is performed, all tasks are processed locally at each edge server. Therefore, the mean sojourn time is identical to the expression used in conventional preemptive-resume priority control. That is, the mean sojourn times of inference and training tasks at edge servers ES_1 and ES_2 are given by the following equations.

$$\mathrm{E}[W_{\text{infer},k}] = \frac{\lambda_{\text{infer}} b_{\text{infer}}^{(2)}}{2(1-\rho_{\text{infer}})} + b_{\text{infer}}, \tag{2}$$

$$\mathrm{E}[W_{\text{train},k}] = \frac{\sum_{j\in\mathcal{T}} \lambda_j b_j^{(2)}}{2(1-\rho_{\text{infer}})(1-\rho_{\text{infer}}-\rho_{\text{train}})} + \frac{b_{\text{train}}}{1-\rho_{\text{infer}}}. \tag{3}$$

Here, $\mathrm{E}[W_{\text{infer},k}]$ and $\mathrm{E}[W_{\text{train},k}]$ denote the mean sojourn times of inference and training tasks, respectively, at edge server ES_k.

Strategy S=(1,1): Both inference and training tasks are offloaded

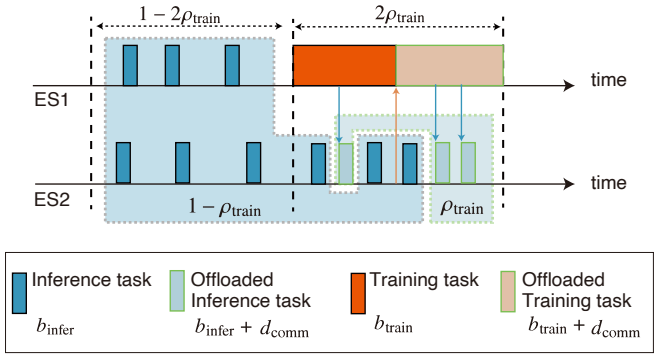

Fig. 2: The derivation of the third and fourth terms in Eq. (5). In addition to queueing delay (corresponding to the first and second terms in the equation), the processing time of inference task is d_{infer} with probability $(1-\rho_{\text{train}})$, and $d_{\text{infer}} + d_{\text{comm}}$, which includes the communication overhead caused by offloading, with probability ρ_{train}.

When bidirectional offloading is performed between edge servers, ES_1 primarily processes training tasks and handles inference tasks only when it is idle. Conversely, ES_2 only processes inference tasks, since all training tasks are offloaded to ES_1.

The mean sojourn time of inference task at ES_1 is same as Eq. (2). The mean sojourn time of training task is expressed as follows.

$$\mathrm{E}[W_{\text{train},1}] = \frac{\lambda_{\text{infer}} b_{\text{infer}}^{(2)} + 2\lambda_{\text{train}} b_{\text{train}}^{(2)}}{2(1-\rho_{\text{infer}})(1-\rho_{\text{infer}}-2\rho_{\text{train}})} + b_{\text{train}} + \frac{d_{\text{comm}}}{2}. \tag{4}$$

Here, d_{comm} represents the average communication overhead required before a task begins execution. Since training tasks can start learning sequentially before receiving all the data, their communication overhead can also be regarded as being approximately d_{comm}. Moreover, in this setting, since half of the training tasks are offloaded from ES_2 and the other half arrive originally at ES_1, the communication overhead is deterministically reduced by a factor $\frac{1}{2}$.

ES_2 must handle not only its local inference tasks but also those offloaded from ES_1. The probability of the latter case equals the probability that an inference task arrives at ES_1, i.e., $1/2$, multiplied by the probability that a training task is being executed at that time (including offloaded ones), i.e., 2ρtrain. Therefore, the probability becomes ρ_{train}. This derivation is also illustrated in Fig. 2. The overall mean sojourn time can be approximated as follows.

$$\mathrm{E}[W_{\text{infer},2}] = (1-2\rho_{\text{train}})\frac{\lambda_{\text{infer}} b_{\text{infer}}^{(2)}}{1-\rho_{\text{infer}}} + 2\rho_{\text{train}}\frac{2\lambda_{\text{infer}} b_{\text{infer}}^{(2)}}{1-2\rho_{\text{infer}}} + b_{\text{infer}} + \rho_{\text{train}} d_{\text{comm}}. \tag{5}$$

In this strategy, inference and training are processed in parallel on separate servers, enabling both real-time responsiveness for inference and continuity for training. However, if

the communication overhead $d$trans is large, the benefits of offloading may be offset.

Strategy S=(0,1) or S=(1,0): One type of task is offloaded

In strategy $\mathbf{S} = (0, 1)$, training tasks are offloaded to ES_1, concentrating the load on a single server. As the mean sojourn time of low-priority tasks in a preemptive-resume $M/G/1$ queue is a convex and increasing function of the total load, this concentration increases the maximum training delay, and communication overhead d_{comm} further worsens it. In strategy $\mathbf{S} = (1, 0)$, inference tasks are offloaded to ES_2, increasing its inference load and interrupting training more frequently. The convex load–delay relationship again leads to a larger maximum delay, and d_{comm} degrades $\mathrm{E}[W_{\mathrm{infer}}]$. Overall, single-type offloading creates unbalanced load and amplifies delay through convexity, so only strategy $\mathbf{S} = (0, 0)$ and $\mathbf{S} = (1, 1)$ are effective in this framework.

IV. Experiments

In this section, the performance of the proposed method is analytically evaluated based on the queuing model formulated in Section III. All results are derived from closed-form solutions, and neither simulations nor implementation experiments are used. We assume an FPL-based setting, where the communication overhead of model exchange can be ignored. Task-switching overhead on the GPU is also not considered.

A. Experimental Setup

OpenAI's CLIP is used as the vision–language model, and PromptFL [5] is employed for the implementation of Federated Prompt Learning. The mean service time of inference tasks (class j = infer) is set to $b_{\mathrm{infer}} = 3$ [ms] with a standard deviation of 1 [ms], while the mean service time of training tasks (class j = train) is set to $b_{\mathrm{train}} = 1$ [hour] with a standard deviation of 0.5 [hour].

The overall system traffic intensity is fixed at ρ, and using a proportional coefficient $r \in [0, 1]$, the traffic intensities for each class are defined as follows, $\rho_{\mathrm{infer}} = r\rho$ and $\rho_{\mathrm{train}} = (1 - r)\rho$. To maintain system stability, the condition $2r\rho < 1$ must hold.

In this setting, the arrival rate λ_j for each class is automatically determined by $\lambda_j = \rho_j / b_j$. For example, when $\rho = 0.3$ and $r = 0.9$, inference tasks arrive every $1/\lambda_{\mathrm{infer}} = 0.01$ [s], while training tasks arrive every $1/\lambda_{\mathrm{train}} = 33.33$ [hours].

As a baseline for comparison, we adopt the method proposed by Wang et al. [12]. Both approaches share the same objective of minimizing the mean sojourn time of low-priority tasks (i.e., training tasks in this study), but differ in several assumptions. Wang et al. assume an environment with three or more edge servers (ESs), where an ongoing training task can be interrupted and migrated to another ES upon the arrival of an inference task. They also do not consider offloading of inference tasks. However, such mid-execution migration can increase the overall delay when inference task arrivals are frequent. Therefore, this study does not assume such migration.

It should be noted that in all existing methods, once offloading is permitted, both inference and training tasks tend to be offloaded simultaneously. In other words, they consistently adopt the strategy corresponding to $(S^{\mathrm{infer}}_{(1\to2)}, S^{\mathrm{train}}_{(2\to1)}) = (1, 1)$. When comparisons are made within this region, the method of Wang et al. tends to exceed the allowable delay threshold W_{th} for inference tasks, whereas the proposed method in this study maintains superiority in that respect.

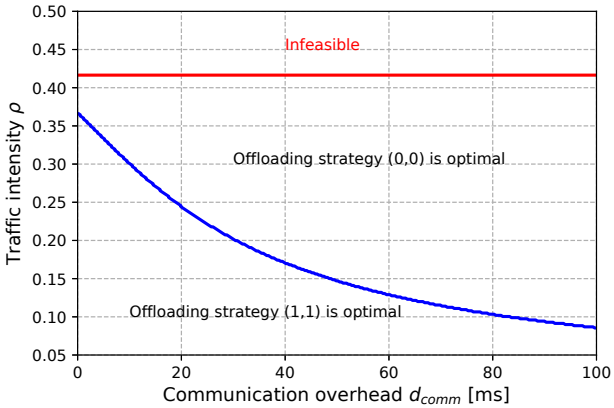

Fig. 3: Visualization of the boundary between different optimal offloading strategies.

A key feature of this study is that it derives the optimal offloading strategy $\boldsymbol{S}^*$ that strictly satisfies the delay constraint for inference tasks, $\mathrm{E}[W_{\mathrm{infer}}] \le W_{\mathrm{th}}$, while minimizing the mean sojourn time of training tasks, $\mathrm{E}[W_{\mathrm{train}}]$.

B. Result

We evaluate performance using mean sojourn time. As shown in Fig. 3, the optimal strategy switches depending on the network delay d_{comm} and the traffic intensity ρ. When d_{comm} is small, bidirectional offloading $\boldsymbol{S} = (1, 1)$ is effective, allowing the constraint on inference tasks, $\mathrm{E}[W_{\mathrm{infer}}] \le W_{\mathrm{th}}$, to be satisfied while significantly reducing the mean sojourn time of training tasks, $\mathrm{E}[W_{\mathrm{train}}]$. Conversely, as d_{comm} increases, the communication overhead due to offloading becomes dominant, and local processing $\boldsymbol{S} = (0, 0)$ becomes optimal.

As shown in the numerical results in Table III, when $d_{\mathrm{comm}} = 5$ ms, the optimal strategy is $\boldsymbol{S} = (1, 1)$, whereas when $d_{\mathrm{comm}} = 60$ ms, $\mathrm{E}[W_{\mathrm{infer}}] > W_{\mathrm{th}}$, and $\boldsymbol{S} = (0, 0)$ becomes the superior option. Therefore, the boundary curve represents the trade-off between satisfying the inference delay constraint and minimizing the training delay.

Table IV compares the average performance of the baseline method (DSO) [12] and the proposed method. Here, $\bar{\mathrm{E}}[W_{\mathrm{infer}}]$ and $\bar{\mathrm{E}}[W_{\mathrm{train}}]$ denote the mean values of $\mathrm{E}[W_{\mathrm{infer}}]$ and $\mathrm{E}[W_{\mathrm{train}}]$, respectively, averaged over the feasible parameter region of ρ and d_{comm} shown in Fig. 3. As shown in Table IV, the proposed method fully satisfies the delay constraint for inference tasks (achieving 100% compliance with $\le W_{\mathrm{th}}$) while maintaining the mean sojourn time of training tasks, $\mathrm{E}[W_{\mathrm{train}}]$, at an almost equivalent level to that of Wang et al. [12].

TABLE III: Mean sojourn times. The mean sojourn times of inference and training tasks are denoted as $\mathrm{E}[W_{\mathrm{infer}}]$ and $\mathrm{E}[W_{\mathrm{train}}]$, respectively, with $W_{\mathrm{th}} = 0.004$ s. **Bold** values indicate those obtained with the optimal strategy. Underlined values denote violations of the tolerable delay threshold W_{th}.

	$d_{\mathrm{comm}} = 0.005$ s		$d_{\mathrm{comm}} = 0.060$ s	
Offloading strategy	$\mathrm{E}[W_{\mathrm{infer}}]$	$\mathrm{E}[W_{\mathrm{train}}]$	$\mathrm{E}[W_{\mathrm{infer}}]$	$\mathrm{E}[W_{\mathrm{train}}]$
ES_1: $\boldsymbol{S} = (1, 1)$	**0.0036**	**3876**	0.0036	3876
ES_2: $\boldsymbol{S} = (1, 1)$	0.0038	–	0.0055	–
$\mathrm{ES}_{1,2}$: $\boldsymbol{S} = (0, 0)$	0.0036	5064	**0.0036**	**5064**

TABLE IV: Comparison results. The achievement rate of the tolerable delay for inference tasks ($\leq W_{\mathrm{th}}$) [%], and the mean sojourn times of inference and training tasks, $\mathrm{E}[W_{\mathrm{infer}}]$ and $\mathrm{E}[W_{\mathrm{train}}]$ averaged over the feasible parameter region of ρ and d_{comm}. $W_{\mathrm{th}} = 0.004$ s.

Method	$\leq W_{\mathrm{th}}$ [%]	$\bar{\mathrm{E}}[W_{\mathrm{infer}}]$	$\bar{\mathrm{E}}[W_{\mathrm{train}}]$
DSO: Wang et al. [12]	34	0.0047	**3813**
Ours ($\boldsymbol{S}^*$)	**100**	**0.0036**	4564

In contrast, the method of Wang et al. satisfies the inference delay constraint in only 34% of cases, and the mean delay for inference tasks, $\bar{\mathrm{E}}[W_{\mathrm{infer}}] = 0.0047$ s, exceeds the threshold. This is because their model does not account for inference offloading and ignores the communication delay caused by interruptions of training tasks. By contrast, the proposed method achieves more practical scheduling through the combination of bidirectional offloading and strict delay constraint satisfaction.

V. Conclusions

In this paper, we have explored the use of queuing theory as a lightweight evaluation mechanism for automatic task offloading of inference and training tasks in federated prompt learning at the edge. Motivated by the use case of smart cities, where FPL is used to evaluate video streams to capture dynamic, region-specific, or anomalous behavior. In FPL, the primary bottleneck shifts from model exchange communication to the efficient coexistence of training and inference at the edge. We have presented the problem encoding in queuing theory. Initial experimental results are presented where the approach is compared to non-preemptive and preemptive scheduling approaches on two edge servers. Results show that queuing theory combines the advantages of both approaches with less overhead.

Future work will seek to expand both the scale and complexity of the evaluation to explore the limits of queuing theory in this context. We will consider expanding this method to multiple edge servers by applying existing research that has proposed algorithms using game theory to determine the amount of offloading [17] or using kmeans++ to model it [18].

Acknowledgment

This research is partially supported by JSPS KAKENHI 23K03871, TAF, the Mazda Foundation, the Micron Foundation, a commissioned research project from the NICT (24101), and EPSRC grant EP/Z533221/1.

References

[1] K. Kanai, H. Kanemitsu, Y. Taku, S. Mori, A. Mine, S. Miyata, H. Imamura, and H. Nakazato, "D2EcoSys: Decentralized Digital Twin EcoSystem Empower Co-Creation City-level Digital Twins Date of Evaluation," *IEICE Transactions on Communications*, vol. E107-B, no. 1, pp. 50–62, 2024.

[2] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Ultra-Reliable Distributed Cloud Network Control with End-to-End Latency Constraints," *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, pp. 2505–2520, 2022.

[3] P. Zhou, Q. Lin, D. Loghin, B. C. Ooi, Y. Wu, and H. Yu, "Communication-efficient Decentralized Machine Learning over Heterogeneous Networks," in *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, 2021, pp. 384–395.

[4] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," in *Proceedings of the Conference on Neural Information Processing Systems (NIPS) Workshop*, 2016.

[5] T. Guo, S. Guo, J. Wang, X. Tang, and W. Xu, "PromptFL: Let Federated Participants Cooperatively Learn Prompts Instead of Models – Federated Learning in Age of Foundation Model," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 5179–5194, 2024.

[6] H. Zhao, W. Du, F. Li, P. Li, and G. Liu, "FedPrompt: Communication-Efficient and Privacy-Preserving Prompt Tuning in Federated Learning," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[7] E. H. Makiyah and N. N. Khamees, "Emulation of Point Cloud Streaming over 5G Network," *International Journal of Information Technology*, vol. 16, no. 4, pp. 2099–2113, 2024.

[8] P. Peng, W. Lin, W. Wu, H. Zhang, S. Peng, Q. Wu, and K. Li, "A Survey on Computation Offloading in Edge Systems: From the Perspective of Delay and Energy Trade-off," *Computer Science Review*, vol. 53, no. C, p. 100656, 2024.

[9] G. Chen, S. Subramaniyan, and X. Wang, "Latency-Guaranteed Co-Location of Inference and Training for Reducing Data Center Expenses," in *2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*, 2024, pp. 473–484.

[10] B. Wang, D. Irwin, P. Shenoy, and D. Towsley, "INVAR: Inversion Aware Resource Provisioning and Workload Scheduling for Edge Computing," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2024, pp. 1511–1520.

[11] T.-L. Chin *et al.*, "Queuing Model Based Edge Placement for Work Offloading in Mobile Cloud Networks," *IEEE Access*, vol. 8, pp. 46 763–46 775, 2020.

[12] J. Wang, W. Zhang, Z. Lin, and Y. Zheng, "Optimal Sequential Computation Offloading and Migration in Mobile Edge Computing with Multi-priority Tasks," *Computing*, 2025.

[13] S. Miyata, T. Miyata, P. Harvey, and I. Khan, "On the Effectiveness of Task Off-loading in Edge based Federated Learning Environment," in *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS) International Workshop on Autonomous Network Management Systems (ANMS)*, 2025.

[14] K. Shibata and S. Miyata, "Edge Server Placement and Task Allocation for Maximum Delay Reduction," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 6207–6217, 2025.

[15] 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.

[16] D. Bertsekas and R. Gallager, *Data Networks (2nd ed.)*. USA: Prentice-Hall, Inc., 1992.

[17] Y. Yokota and S. Miyata, "Latency Focused Load Balancing Method between Cloudlets using Game Theory," *IEICE Nolta Journal*, vol. 15, no. 2, pp. 473–484, 2024.

[18] K. Ogawa, S. Miyata, and K. Kanai, "Transfer Probability-Based Job Reallocation Method for Heterogeneous Edge Clouds," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 4549–4562, 2025.