Online Scheduling of Federated Learning with In-Network Aggregation and Flow Routing

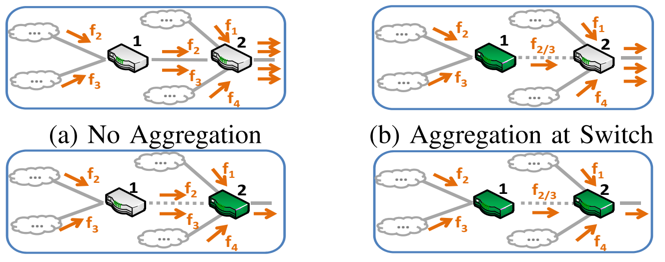
Mingtao Ji[1], Lei Jiao[2], Yitao Fan[1], Yang Chen[3], Zhuzhong Qian[1], Ji Qi[4], Gangyi Luo[4], Baoliu Ye[1]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, China [2]University of Oregon, USA
[3]Fudan University, China [4]China Mobile (Suzhou) Software Technology Co. Ltd., China

***Abstract*—Continuously orchestrating in-network model aggregations for federated learning faces fundamental challenges such as the combinatorial nature of traffic reduction, the dynamic trade-offs between system overhead and model convergence, and the unpredictable inputs from uncertain system environments. In this work, we model a nonlinear mixed-integer program to optimize the long-term total cost of federated learning computation overhead, traffic reduction, network delay, and programmable switch reconfigurations over time. To attack the lexicographic minimax, submodular, and online nature of this problem, we propose a polynomial-time algorithmic framework to judiciously designate the timing of reconfigurations, while designing and invoking a linearized transformation for selecting routing paths, a greedy sub-algorithm for selecting aggregation locations, and an online learning sub-algorithm for controlling federated learning convergence. We demonstrate our rigorous mathematical insights behind our algorithms, and prove the competitive ratio as the performance guarantee. Using trace-driven evaluations, we have validated our approach's superiority over existing methods.**

I. Introduction

Federated Learning (FL) [1, 2] typically consists of a group of client devices and a server that communicate via a network. In each round, each FL client updates the model using its local data and sends the updated model with related information to the FL server for aggregation; then, each client downloads the aggregated global model from the server, and continues to update that model in the next round. This paradigm involves excessive communication overhead between the FL clients and the FL server, especially when the model becomes huge today and needs many rounds of training [3]. Leaving such traffic unmanaged can incur large traffic footprint and delay upon the network and impact the performance of the entire FL system.

While different entities may use FL for different purposes, service providers and network operators who control the network infrastructures and conduct FL upon such infrastructures, e.g., for management tasks [4, 5], are uniquely positioned to address the above problem via *in-network aggregation* [6, 7]. In fact, with the deployment of programmable switches (e.g., P4 [8, 9]) and smart network interface cards, these service providers can choose to aggregate the models sent from the FL clients in the network before reaching the FL server, and due to the associative property of aggregations, further aggregate the already-aggregated models on the FL server to still obtain the same global model. Such in-network aggregation reduces the network traffic, alleviates the load on the FL server, improves the overall resource efficiency, and more importantly, achieves all these benefits by completing aggregation at line speed [10].



(a) No Aggregation (b) Aggregation at Switch 1
(c) Aggregation at Switch 2 (d) Aggregation at Both

Fig. 1: Different In-Network Aggregation Decisions

However, to enable service providers to optimally operate in-network aggregation jointly with federated learning is nontrivial, due to multiple fundamental challenges as follows.

First, in-network aggregation actually possesses a complex cost structure, making it difficult to precisely characterize its benefit [7, 10]. Consider *traffic reduction* in terms of the number of the reduced flows. In Fig. 1(d), there are two flows before and one flow after Switch 1, and three flows before and one flow after Switch 2. The traffic reduction for the two switches in Fig. 1(d) is thus (2−1)+(3−1)=3; analogously, the traffic reduction for Fig. 1(a), Fig. 1(b), and Fig. 1(c) are (2−2)+(4−4)=0, (2−1)+(3−3)=1, and (2−2)+(4−1)=3, respectively. Note that the traffic reduction of Fig. 1(d), with aggregation at both switches, is not equal to the sum of the traffic reduction of Fig. 1(b) with aggregation only at Switch 1 and that of Fig. 1(c) with aggregation only at Switch 2. That is, the traffic reduction of downstream switches depends on the in-network aggregation at upstream switches. A network with a complicated topology only makes such dependency trickier.

Second, as in-network aggregation and federated learning are intertwined, it is challenging to jointly control them in an online manner in the uncertain system environments. To achieve the global model convergence [1, 3], we need to ensure a sufficient number of global aggregations cumulatively. In an online setting, if we do not conduct the global aggregation currently, it removes the need to orchestrate in-network aggregations but can force us to do global aggregations later even if the network conditions (e.g., delay) become worse then; if we do the global aggregation now, it may turn out to be unnecessary if the future network performance becomes better. As how system environments vary in the future is unknown, it is not easy to make the control decisions on the fly.

Third, controlling in-network aggregations continuously on programmable switches incurs reconfiguration (a.k.a. switching) cost, e.g., the leading and initialization time for loading new programs and/or data [11], which is not straightforward to manage. Keeping existing in-network aggregation configurations stable may not be ideal, because with dynamic routing, the flows from the FL clients to the FL server that pass any specific switch may change as time goes; but adjusting the in-network aggregation configurations immediately in response to flow routing variations may incur excessive reconfiguration cost. Flow routing itself needs to be controlled as well, as it impacts not only flow aggregation but also network delay.

Existing research on FL and distributed machine learning with in-network aggregations falls short for addressing the aforementioned challenges. Those on FL [12–17] have neither characterized the traffic reduction impact in complex networks, nor addressed the dynamic control of FL convergence with the continuous reconfiguration of programmable switches in the long term. Those on distributed machine learning and neural networks [6–10, 18] are generally inapplicable, either, as their solutions often fail to consider the FL and the online settings of the problem. See Section VI for detailed discussions.

In this paper, we present a comprehensive mathematical and algorithmic study on the online optimization of network-assisted federated learning. We make multiple contributions:

We first model a long-term total cost minimization problem to jointly optimize the FL computational overhead, the programmable switches' reconfiguration cost, the traffic reduction due to in-network aggregation, and the maximum FL network delay. Our formulation preserves the FL convergence as specified and captures both the combinatorial nature of traffic reduction and the lexicographic minimax nature of network delay, while controlling FL, in-network aggregation, and flow routing with almost no assumption on the time-varying inputs from the system environments. Our problem is a non-linear mixed-integer program, which is, unsurprisingly, NP-hard.

We then design three polynomial-time algorithms that fuse together to solve this problem in an online manner. Our key idea of the online control algorithm is to balance the switching cost and the non-switching cost by judiciously postponing the switching operation (i.e., changing the locations of in-network aggregations) until the cumulative non-switching cost since the last switching operation surpasses a controllable constant times the switching cost of that last switching operation. Our online control algorithm invokes two sub-algorithms. The first sub-algorithm determines the routing paths by optimally solving a continuous linear program transformed from our original discrete min-max formulation, and chooses the network switches for in-network aggregations by greedily collecting the switches that bring the maximum increment to existing traffic reduction. The second sub-algorithm achieves federated learning convergence via online learning [19], using rectified primal-dual steps to decide online in each round the number of iterations for local training and whether to conduct global aggregation, without worrying about unknown future inputs.

We further perform the theoretical analysis for our approach and provide the mathematical insights behind our algorithm design. We derive a parameterized-constant *competitive ratio* for our entire approach, which indicates the multiplicative gap between the total cost of our online approach and the offline optimum. For the sub-algorithms, we prove the equivalence between our lexicographic min-max optimization of the network delay and a corresponding linear program [20, 21], and establish the monotonicity and submodularity of our traffic reduction function [22], which leads to a small *approximate ratio* against the optimal reduction. We also point out the sub-linear growth of the *regret* in terms of the cumulative non-switching cost compared to a sequence of one-shot optimizations and the *fit* in terms of the cumulative violation of the FL convergence.

We finally conduct large-scale evaluations using real-world data. We use NetworkX [23] to compose a network of 1,024 nodes, connecting an FL server with a varying number of FL clients at different locations [10] with real path delay [24], and implement an in-network aggregation protocol [7] upon real P4 programmable switches to collect the required inputs to our experiments. We adopt TensorFlow FL [25] to train a Support Vector Machine (SVM) and a Convolutional Neural Network (CNN) for the MNIST [26] and the CIFAR-10 [27] tasks. Overall, (i) our approach reduces the total cost by 23% in real time and by 19%$\sim$47% cumulatively on average, compared to combinations of existing methods; (ii) our approach performs consistently well for different client distributions, gradient sizes, and reconfiguration overhead; (iii) our approach finishes in seconds, meeting real-world needs; and (iv) our approach succeeds in training SVM and CNN models of high accuracy.

II. Modeling and Formulation

A. System Settings and Models

Federated Learning (FL): We consider a Federated Learning (FL) system that consists of one FL server and a set $\mathcal{N} = \{1, ..., |\mathcal{N}|\}$ of FL clients, where the FL server and the FL clients communicate via a network. To train the global model $\boldsymbol{w}$, we minimize the total "loss". We define the loss on the client $i \in \mathcal{N}$ as $F_i(\boldsymbol{w}) = \frac{1}{|\mathcal{D}_i|}\sum_{l\in\mathcal{D}_i} f_l(\boldsymbol{w})$, where $f_l(\cdot)$ is the loss function corresponding to the data sample l and $\mathcal{D}_i$ is the set of all the data samples on the client i, and also define the total loss as $F(\boldsymbol{w}) = \frac{1}{\sum_{i\in\mathcal{N}}|\mathcal{D}_i|}\sum_{i\in\mathcal{N}}(|\mathcal{D}_i|F_i(\boldsymbol{w}))$. Training the global model $\boldsymbol{w}$ is to solve the problem $\min_{\boldsymbol{w}} F(\boldsymbol{w})$.

Without loss of generality, in this paper, we target the FL approach as follows. We solve the problem $\min_{\boldsymbol{w}} F(\boldsymbol{w})$ using κ^G global iterations, starting with the initial value $\boldsymbol{w}_0$ and ending with the final value $\boldsymbol{w}_{\kappa^G}$. We desire $F(\boldsymbol{w}_{\kappa^G}) - F(\boldsymbol{w}^*) \leq \varepsilon \cdot (F(\boldsymbol{w}_0) - F(\boldsymbol{w}^*))$, where $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} F(\boldsymbol{w})$, and $\varepsilon \in (0, 1)$ is a parameter that we can call "global convergence accuracy". To make this inequality hold, we need to conduct the following number of global iterations [2, 3], where r_0 is a constant and x will be described next:

$$\kappa^G(\varepsilon, x) \geq r_0 \frac{\log(1/\varepsilon)}{1-x}.$$

In the k-th global iteration, on each client i, we need to solve the problem $\min_{\boldsymbol{w}_{i,k}} J_{i,k}(\boldsymbol{w}_{i,k})$ through κ^L local iterations, starting with the initial value $\boldsymbol{w}^0_{i,k} = \boldsymbol{w}_{k-1}$ and ending with the value $\boldsymbol{w}^{\kappa^L}_{i,k}$, by executing the gradient-descent steps as $\boldsymbol{w}^{\tau}_{i,k} = \boldsymbol{w}^{\tau-1}_{i,k} - \delta\nabla J_{i,k}(\boldsymbol{w}^{\tau-1}_{i,k})$. Here, δ is the step size,

A Federated Learning Algorithm

```
initialize $w_0$ and $\nabla F(w_0)$; ▷ Initialization on server
for $k = 1, ..., \kappa^G$ do
    ▷ Local training on each client $i$:
    download $w_{k-1}$ and $\nabla F(w_{k-1})$ from server;
    for $\tau = 1, ..., \kappa^L$ do
        $w_{i,k}^{\tau} = w_{i,k}^{\tau-1} - \delta \nabla J_{i,k}(w_{i,k}^{\tau-1})$;
    upload $w_{i,k} = w_{i,k}^{\kappa^L}$ and $\nabla F_i(w_{i,k})$ to server;
    ▷ Global aggregation on server:
    $w_k = \frac{1}{|\mathcal{N}|}\sum_{i \in \mathcal{N}} w_{i,k}$;
    $\nabla F(w_k) = \frac{1}{|\mathcal{N}|}\sum_{i \in \mathcal{N}} \nabla F_i(w_{i,k})$;
```

and $J_{i,k}(\cdot)$ is a dedicated function defined as $J_{i,k}(\boldsymbol{w}) = F_i(\boldsymbol{w}) - [\nabla F_i(\boldsymbol{w}_{k-1}) - \beta_1 \nabla F(\boldsymbol{w}_{k-1})]^{\mathsf{T}}(\boldsymbol{w} - \boldsymbol{w}_{k-1}) + \frac{\beta_2}{2}||\boldsymbol{w} - \boldsymbol{w}_{k-1}||^2$, where $\beta_1, \beta_2 \geq 0$ are constant parameters. We desire $\|\nabla J_{i,k}(\boldsymbol{w}_{i,k}^{\kappa^L})\| \leq x\|\nabla J_{i,k}(\boldsymbol{w}_{k-1})\|$, where $x \in (0,1)$ is a parameter that we call "local convergence accuracy". To make this inequality hold, we need to conduct the following number of local iterations on each client [2, 3], where q_0 is a constant:

$$\kappa^L(x) \geq q_0 \log_2 (1/x).$$

Note that in this paper we focus on synchronous FL, instead of asynchronous FL which could be of independent interest.

In-Network Aggregations: We consider the network that connects the FL clients and the FL server having a set of programmable switches, denoted as $\mathcal{J} = \{1, ..., |\mathcal{J}|\}$, where the in-network aggregations can happen. As indicated in the aforementioned FL algorithm, in each global iteration k, every FL client i sends a flow, containing $\boldsymbol{w}_{i,k}$ and $\nabla F_i(\boldsymbol{w}_{i,k})$, at the end of the local training to the FL server for aggregation. Due to programmable switches, flows that meet at a common switch can choose to do an early aggregation on the switch, and then the aggregated flow can continue to travel to the FL server for any further aggregation. For any switch, the number of outgoing flows is always 1 if the in-network aggregation occurs at this switch. If we use Q_j to represent the number of incoming flows for the switch j, then we can define "traffic reduction" for the switch j as $Q_j - 1$. In general, for a set S of switches, we define the function $\psi(S)$ as the traffic reduction on S, i.e., the total traffic reduction if every switch in S conducts in-network aggregation. Now, if we consider a switch $j \notin S$ which currently performs no in-network aggregation, then one and only one of the following cases must hold.

- (Case I) All Q_j outgoing flows of j go into some switch k in S: $\psi(\{j\} \cup S) = Q_j - 1 + (Q_k - (Q_j - 1)) - 1 + \psi(S \setminus \{k\}) = Q_k - 1 + \psi(S \setminus \{k\}) = \psi(S)$;
- (Case II) Some or all outgoing flows from S go into j: $\psi(\{j\} \cup S) = Q_j - 1 + \psi(S)$;
- (Case III) None of Q_j outgoing flows of j go into any switch in S, and also no outgoing flows from S go into j: $\psi(\{j\} \cup S) = Q_j - 1 + \psi(S)$.

That is,

$$\psi(\{j\} \cup S) = \begin{cases} \psi(S) & \text{Case I;} \\ Q_j - 1 + \psi(S) & \text{Cases II and III.} \end{cases} \tag{1}$$

We envisage $\exists j$, such that $Q_j > 1$; otherwise, we would have $Q_j = 1, \forall j$, with no need for in-network aggregation.

We study the system over a series of consecutive time slots $\mathcal{T} = \{1, 2, ..., |\mathcal{T}|\}$. At each time slot $t \in \mathcal{T}$, we impose a threshold, denoted as M_t, as the maximum total number of switches that are allowed to perform in-network aggregations [10, 28]. This can be set by the service provider to restrict the in-network aggregation overhead, or reserve programmable switches for other purposes such as for non-FL workload.

Reconfiguration Cost: For any switch, at the time slot t, if the in-network aggregation is conducted, then the system needs to reconfigure this switch if it was not configured for any in-network aggregation at the previous time slot $t - 1$. Reconfiguration incurs overhead, denoted as b, including the delay for sending the flow table entries to the programmable switch and uploading the register states in the programmable switch [11]. At t, if we continue with whatever the configuration was at $t - 1$, the reconfiguration cost at t will be 0.

Flow Routing: For any FL client i to communicate with the FL server, it needs to choose and use a path to go through the network. Aligned with lots of existing studies [29, 30], we focus on the case of unsplittable flows. We use $\mathcal{R}_i$ to denote the set of the available paths for the FL client i. For the path $p \in \mathcal{R}_i$, we denote by $d_{p,t}$ the network delay of this path at the time slot t. We also use $m_{p,j} \in \{1, 0\}$ to imply whether or not the switch j is on the path p. A switch could be simultaneously on multiple paths; but it is impossible for one path to be simultaneously used by two or more FL clients, as every path starts with an FL client and ends at the FL server.

Control Decisions: We focus on making four types of decisions in this paper. $x_t \in (0, \chi]$ denotes the local convergence accuracy of the FL clients at the time slot t, where we consider $\chi < 1$ as the upper bound for the local convergence accuracy pre-specified by the service provider. $y_t \in \{1, 0\}$ denotes whether or not to conduct the global model aggregation for FL at the end of the time slot t. $z_{j,t} \in \{1, 0\}$ denotes whether or not to conduct the in-network aggregation at the switch j at the time slot t, where we also denote $S_t \triangleq \{j | z_{j,t} j = j, \forall j\}$. $w_{i,p,t} \in \{1, 0\}$ denotes whether or not to use the path p to route the flow of the FL client i at the time slot t.

Total Cost: The total cost at the time slot t has multiple components. First, the computation overhead of local training on all FL clients is $\sum_{i \in \mathcal{N}} \kappa^L(x_t)|\mathcal{D}_i|a_t$, where a_t is the overhead for performing one local training iteration on a single data sample on each client. Next, the reconfiguration cost for configuring switches for the in-network aggregations is $\sum_{j \in \mathcal{J}} [z_{j,t} - z_{j,t-1}]^+ b$, where $[\cdot]^+ \triangleq \max\{0, \cdot\}$. Then, the total traffic reduction, treated as inverse cost, is $y_t \cdot \frac{1}{\psi(S_t)c}$, where c is size of the model plus the gradient information going from each client to the server. To make it non-negative, we do not use $y_t(-\psi(S_t)c)$. Finally, the network delay, depending on the maximum path delay [6, 7], is $y_t \max_{i \in \mathcal{N}, p \in \mathcal{R}_i} w_{i,p,t} d_{p,t}$.

B. Problem Formulation and Algorithmic Challenges

Control Problem: Based on the aforementioned models, we formulate the optimization problem $\mathbb{P}$ for minimizing the total cost over the entire time horizon:

$$\begin{aligned}\min\quad & \mathcal{P} \triangleq \sum_{t\in\mathcal{T}} \Big\{ \kappa^L(x_t) \sum_{i\in\mathcal{N}} |\mathcal{D}_i| a_t + \sum_{j\in\mathcal{J}} [z_{j,t} - z_{j,t-1}]^+ b \\ & + \tfrac{y_t}{\psi(S_t)c} + y_t \max\nolimits_{i\in\mathcal{N}, p\in\mathcal{R}_i} w_{i,p,t} d_{p,t} \Big\} \\ s.t.\quad & C_1 : \textstyle\sum_{p\in\mathcal{R}_i} w_{i,p,t} = 1, \forall i\in\mathcal{N}, \forall t\in\mathcal{T}, \\ & C_2 : z_{j,t} \le \textstyle\sum_{i\in\mathcal{N}, p\in\mathcal{R}_i} w_{i,p,t} m_{p,j}, \forall j\in\mathcal{J}, \forall t\in\mathcal{T}, \\ & C_3 : \textstyle\sum_j z_{j,t} \le M_t, \forall t\in\mathcal{T}, \\ & C_4 : \textstyle\sum_{t\in\mathcal{T}} y_t \ge \kappa^G(\varepsilon, \max_{t\in\mathcal{T}} x_t), \\ var.\quad & x_t \in (0,\chi], y_t\in\{0,1\}, z_{j,t}\in\{0,1\}, w_{i,p,t}\in\{0,1\}, \\ & \forall j\in\mathcal{J}, \forall p\in\mathcal{R}_i, \forall i\in\mathcal{N}, \forall t\in\mathcal{T}. \end{aligned}$$

The optimization objective is the total cost. Constraint C_1 ensures that at each time slot each FL client uses only one path. Constraint C_2 ensures that at each time slot the in-network aggregation can only be done on a switch if there is at least one path that contains this switch is used. Constraint C_3 ensures that at each time slot the total number of switches to perform in-network aggregations does not exceed the pre-specified threshold. Constraint C_4 ensures that a sufficient number of FL aggregations are conducted, so that the global model is trained to the desired global convergence accuracy ε. This problem is NP-hard due to the submodularity of the function $\psi(\cdot)$ [22], which will also be elaborated later.

As in the above, to align with numerous existing research, we treat multi-objective optimization here as a weighted sum of the different objectives, transforming it into single-objective optimization. That is, we actually maintain a weight for each term in the objective, also because the units of such different terms could be different. We can control the optimization by adjusting such weights based on the service provider's needs.

Algorithmic Goal: For the problem $\mathbb{P}$, we aim to design a polynomial-time online approach while rigorously satisfying $\mathcal{P}(\{\tilde{x}_t, \bar{y}_t, \bar{\boldsymbol{z}}_t, \bar{\boldsymbol{w}}_t, \forall t\}) \le r\cdot\mathcal{P}^*$, where the objective function $\mathcal{P}$ is evaluated with the solutions produced by our online approach which only knows the inputs gradually on the fly as time goes; $\mathcal{P}^*$ is the offline optimal objective function value assuming all the inputs over time are known in prior at once; and r is a constant called the *competitive ratio*.

Algorithmic Challenges: Solving $\mathbb{P}$ online faces multiple challenges. First, $\psi(\cdot)$ and $\max$ are both nonlinear and discrete functions, which is difficult to address, as we need to explore the combinations of the different decisions in a large search space. Second, for the function $[\cdot]^+$, we note that at $t-1$, as we have no idea about what decision we will make for $\boldsymbol{z}_t$ at t, it is difficult to choose a right value for $\boldsymbol{z}_{t-1}$ at $t-1$ to minimize $[\boldsymbol{z}_t - \boldsymbol{z}_{t-1}]^+$. Third, due to the long-term constraint C_4, choosing a small y_t early could force a large y_t later in the future, in order to respect C_4; yet, as we have no idea how S_t and $d_{p,t}$ in the objective will vary over time, a large y_t later can lead to the sub-optimum overall. Choosing a large y_t early may alleviate this concern, but can incur a large objective value if, for example, $d_{p,t}$ becomes small in the future.

III. Online Algorithm Design

A. Algorithms Overview

To tackle the aforementioned challenges, we design three algorithms. Our overall online control algorithm is Algorithm 1, which invokes Algorithms 2 and 3. Algorithm 1 dynamically balances the switch reconfiguration cost with all the other types of cost in the problem $\mathbb{P}$'s objective function. Algorithm 2 selects the paths for routing FL flows and the switches for in-network aggregations. Algorithm 3 controls FL local training and global aggregations. We elaborate these algorithms with their rationales in the following sections sequentially. To that end, we decompose the problem $\mathbb{P}$ into multiple sub-problems as follows, and our algorithms work with these sub-problems to produce the solution online to the original problem $\mathbb{P}$.

First, we focus on path selection and from $\mathbb{P}$, we extract the following problem $\mathbb{P}_{t,1}$, which centers on the $\max$ function:

$$\begin{aligned}\min\quad & \mathcal{P}_{t,1} \triangleq \max\nolimits_{i\in\mathcal{N}, p\in\mathcal{R}_i} w_{i,p,t} d_{p,t} \\ s.t.\quad & C_1, \\ var.\quad & w_{i,p,t}\in\{0,1\}, \forall p\in\mathcal{R}_i, \forall i.\end{aligned}$$

Second, we focus on switch selection and from $\mathbb{P}$, we extract the following problem $\mathbb{P}_{t,2}$, which centers on the $\psi(\cdot)$ function:

$$\begin{aligned}\max\quad & \mathcal{P}_{t,2} \triangleq \psi(S_t) \\ s.t.\quad & z_{j,t} \le \textstyle\sum_{i\in\mathcal{N}, p\in\mathcal{R}_i} \bar{w}_{i,p,t} m_{p,j}, \forall j, \\ & C_3, \\ var.\quad & z_{j,t}\in\{0,1\}, \forall j,\end{aligned}$$

where the solution $\bar{\boldsymbol{w}}_t$ from $\mathbb{P}_{t,1}$ is placed in the constraints.

Third, we split the objective function of the problem $\mathbb{P}$ as $\mathcal{P} = C^t_{\neg S}(x_t, y_t, \boldsymbol{z}_t, \boldsymbol{w}_t) + C^t_S(\boldsymbol{z}_t, \boldsymbol{z}_{t-1})$:

$$\begin{aligned} C^t_{\neg S}\left(x_t, y_t, \boldsymbol{z}_t, \boldsymbol{w}_t\right) &\triangleq \kappa^L(x_t) \textstyle\sum_{i\in\mathcal{N}} |\mathcal{D}_i| a_t + \tfrac{y_t}{\psi(S_t)c} \\ &+ y_t \max\nolimits_{i\in\mathcal{N}, p\in\mathcal{R}_i} w_{i,p,t} d_{p,t}, \\ C^t_S\left(\boldsymbol{z}_t, \boldsymbol{z}_{t-1}\right) &\triangleq \textstyle\sum_j [z_{j,t} - z_{j,t-1}]^+ b,\end{aligned}$$

which we call the *non-switching cost* and the *switching cost*, respectively. Then, we place the solution $\bar{\boldsymbol{z}}_t$ from $\mathbb{P}_{t,2}$ and the solution $\bar{\boldsymbol{w}}_t$ from $\mathbb{P}_{t,1}$ in $\mathcal{P}$, and define the following problem $\mathbb{P}_3$ for the FL control:

$$\begin{aligned}\min\quad & \textstyle\sum_{t\in\mathcal{T}} \mathcal{P}_{t,3} \triangleq \sum_{t\in\mathcal{T}} C^t_{\neg S}\left(x_t, y_t, \bar{\boldsymbol{z}}_t, \bar{\boldsymbol{w}}_t\right) \\ s.t.\quad & C_4, \\ var.\quad & x_t\in(0,1), y_t\in\{0,1\}, \forall t.\end{aligned}$$

Note that $\mathbb{P}_{t,1}$ and $\mathbb{P}_{t,2}$ are defined for every time slot $t\in\mathcal{T}$; and $\mathbb{P}_3$ is defined over the entire time horizon $\mathcal{T}$.

B. Lazy-Switch-Based Online Control

Algorithm 1: Algorithm 1 postpones the switching operation that changes where to conduct the in-network aggregations until the cumulative non-switching cost since the last switching operation surpasses a parameter η times the switching cost of that last switching operation. Line 2 checks whether or not the condition for performing a potentially new switching operation holds. If such condition holds, Lines 3$\sim$4 invoke the sub-algorithms to find the new control decisions for the

Algorithm 1 Overall Online Control Algorithm

```
Input: initialize $t' = 0$, $\bar{\boldsymbol{z}}_0 = \bar{\boldsymbol{w}}_0 = \boldsymbol{0}$, $\bar{\boldsymbol{z}}_{-1} = \boldsymbol{0}$;
 1: for $t = 1, 2, 3, ..., |\mathcal{T}|$ do
 2:     if $C_S^{t'}(\bar{\boldsymbol{z}}_{t'}, \bar{\boldsymbol{z}}_{t'-1}) \leq \frac{1}{\eta}\sum_{v=t'}^{t-1} C_{\neg S}^{v}(\widetilde{x}_v, \bar{y}_v, \bar{\boldsymbol{z}}_v, \bar{\boldsymbol{w}}_v)$ then
 3:         obtain $\bar{\boldsymbol{z}}_t$, $\bar{\boldsymbol{w}}_t$ via Algorithm 2;
 4:         obtain $\widetilde{x}_t, \widetilde{y}_t$ via Algorithm 3, given $\bar{\boldsymbol{z}}_t$, $\bar{\boldsymbol{w}}_t$;
 5:         if $\bar{\boldsymbol{z}}_t \neq \bar{\boldsymbol{z}}_{t-1}$, then set $t' = t$;
 6:     end if
 7:     if $t' < t$ then
 8:         set $\bar{\boldsymbol{z}}_t = \bar{\boldsymbol{z}}_{t-1}$, $\bar{\boldsymbol{w}}_t = \bar{\boldsymbol{w}}_{t-1}$;
 9:         obtain $\widetilde{x}_t, \widetilde{y}_t$ via Algorithm 3, given $\bar{\boldsymbol{z}}_t$, $\bar{\boldsymbol{w}}_t$;
10:     end if
11:     round $\widetilde{y}_t$ to $\bar{y}_t$, while ensuring $\mathbb{E}[\bar{y}_t] = \widetilde{y}_t$;
12: end for
```

current time slot. In Line 5, if the new in-network aggregation locations are indeed different from the existing locations, then we record the current time slot in t' to mark it as the latest switching operation. In Line 7, if there is no switching at the current time slot, we just reuse the existing decisions of the switches $\bar{\boldsymbol{z}}_{t-1}$ for in-network aggregations and the paths $\bar{\boldsymbol{w}}_{t-1}$ for routing FL flows and then update the FL control decisions $\widetilde{x}_t$ and $\widetilde{y}_t$ in Lines 8$\sim$9. In Line 11, we conduct randomized rounding, i.e., set $\bar{y} = 1$ with the probability $\widetilde{y}_t$ and $\bar{y} = 0$ with the probability $1 - \widetilde{y}_t$. That is, $\mathbb{E}[\bar{y}_t] = 1 \cdot \widetilde{y}_t + 0 \cdot (1 - \widetilde{y}_t) = \widetilde{y}_t$.

C. Linearized Path and Submodular Switch Selection

To select the routing path for each FL client toward the FL server at the time slot t, we convert $\mathbb{P}_{t,1}$, which is actually a discrete lexicographic min-max optimization problem, into a linear program $\mathbb{P}'_{t,1}$ solvable in polynomial time by standard linear program solvers. The problem $\mathbb{P}'_{t,1}$ is as follows:

$$\begin{array}{ll} \min & \mathcal{P}'_{t,1} \triangleq \sum_{i \in \mathcal{N}} \sum_{p \in \mathcal{R}_i} \left\{ \left((\sum_{i \in \mathcal{N}} |\mathcal{R}_i|)^{d_{p,t}} - 1 \right) \cdot w_{i,p,t} \right\} \\ s.t. & C_1, \\ var. & 0 \leq w_{i,p,t} \leq 1, \forall p \in \mathcal{R}_i, \forall i. \end{array}$$

Solving $\mathbb{P}'_{t,1}$ will automatically result in the optimal *integer* solution of $\mathbb{P}_{t,1}$. Though not obvious, these two problems are actually equivalent, as shown by Theorem 1 in Section IV.

To select network switches for in-network aggregations, we need to solve $\mathbb{P}_{t,2}$. We motivate our algorithm design by Fig. 1. In Fig. 1(a)$\sim$Fig. 1(d), denoting the set of the switches performing in-network aggregations as $S_1 = \{\}$, $S_2 = \{1\}$, $S_3 = \{2\}$, and $S_4 = \{1, 2\}$, respectively, we have $\psi(S_1) = 0$, $\psi(S_2) = 1$, $\psi(S_3) = 3$, and $\psi(S_4) = 3$. We see the following:

- While $S_2 \subset S_4$ and $S_3 \subset S_4$, we have $\psi(S_2) \leq \psi(S_4)$ and $\psi(S_3) \leq \psi(S_4)$;
- While $S_1 \subset S_2$, if we add the switch "2" to S_1 and S_2, the additional traffic reductions are $\psi(S_1 \cup \{2\} = S_3) - \psi(S_1) = 3$ and $\psi(S_2 \cup \{2\} = S_4) - \psi(S_2) = 2$, respectively, i.e., $\psi(S_1 \cup \{2\}) - \psi(S_1) \geq \psi(S_2 \cup \{2\}) - \psi(S_2)$. This is also aligned with Equation (1).

Shown by Theorem 2 in Section IV, the above two phenomena correspond to monotonicity and submodularity, which hold in general for the function $\psi(\cdot)$. Based on this, we can "grow" the set of switches for in-network aggregations in an iterative manner, where in each iteration we add the one switch that brings the largest increment to the existing traffic reduction.

Algorithm 2 Path and Switch Selection Algorithm, $\forall t$

```
Input: initialize $\boldsymbol{z}_t = \boldsymbol{0}$, $S = \emptyset$;
 1: obtain $\bar{\boldsymbol{w}}_t$ by solving $\mathbb{P}'_{t,1}$ via a linear program solver (e.g.,
    CVXPY [31]);
 2: obtain $\mathcal{J}_t \triangleq \{j | \bar{w}_{i,p,t} m_{p,j} = 1, \exists p \in \mathcal{R}_i, \exists i \in \mathcal{N}\}$;
 3: for $l = 1, 2, 3, ..., |\mathcal{J}_t|$ do
 4:     find the switch $j^* = \arg\max_{j \in \mathcal{J}_t} \{\psi(\{j\} \cup S) - \psi(S)\}$
        using Equation (1);
 5:     set $S = \{j^*\} \cup S$, $\mathcal{J}_t = \mathcal{J}_t \setminus \{j^*\}$;
 6:     set $z_{j,t} = 1, \forall j \in S$ , and $z_{j,t} = 0, \forall j \notin S$;
 7:     if $\boldsymbol{z}_t$ violates Constraint $C_3$, then break;
 8:     set $\bar{\boldsymbol{z}}_t = \boldsymbol{z}_t$;
 9: end for
```

Algorithm 3 Federated Learning Control Algorithm, $\forall t$

```
Input: initialize step sizes $\alpha$, $\mu$;
 1: observe $f_t(\boldsymbol{I}_t)$, $g_t(\boldsymbol{I}_t)$ given $\bar{\boldsymbol{z}}_t$, $\bar{\boldsymbol{w}}_t$, based on (2) and (3);
 2: obtain $\tilde{\boldsymbol{I}}_t$ by solving (4);
 3: update $\lambda_{t+1}$ by (5);
```

Algorithm 2: With the above insights, Algorithm 2 selects the routing paths and the network switches on such routing paths. Line 1 can invoke any standard linear program solver, which, for example, can find the ε-accurate optimal solution in $O((\sum_i |\mathcal{R}_i|)^2 \log(1/\varepsilon))$ iterations via the interior point method. Line 2 finds out the set of all the switches, each of which is on at least one path used the FL flow. In Line 3, we conduct up to $|\mathcal{J}_t|$ iterations as we add exactly one switch in each iteration. For Lines 4$\sim$5, this is our greedy step based on submodularity. In Lines 6$\sim$7, given $\boldsymbol{z}_t$, we will check whether the threshold for the total number of switches that conduct in-network aggregations is still satisfied. If so, we continue to update $\bar{\boldsymbol{z}}_t$ to select more switches; otherwise, we just keep $\bar{\boldsymbol{z}}_t$ in the previous iteration. Overall, compared to a brutal force approach of $\mathcal{O}(2^{|\mathcal{J}_t|})$, our Lines 3$\sim$9 take only $\mathcal{O}(|\mathcal{J}_t|^2)$.

D. Online-Learning-Based Federated Learning Control

We introduce the following notations regarding $\mathbb{P}_3$:

$$f_t = \mathcal{P}_{t,3} = C_{\neg S}^{t}(x_t, y_t, \bar{\boldsymbol{z}}_t, \bar{\boldsymbol{w}}_t), \tag{2}$$

$$g_t = \kappa^G(\varepsilon, \chi)/|\mathcal{T}| - y_t. \tag{3}$$

If we denote $\boldsymbol{I}_t = (x_t, y_t)^\top$ and define $\mathcal{I}_t \triangleq (0, \chi] \times [0, 1]$, we can write $\mathbb{P}_3$, with $y_t, \forall t$ in the continuous domain, as follows:

$$\min \ \sum_t f_t(\boldsymbol{I}_t), \ s.t. \ \sum_t g_t(\boldsymbol{I}_t) \leq 0, \ var. \ \boldsymbol{I}_t \in \mathcal{I}_t, \forall t.$$

In the following, we also denote $\tilde{\boldsymbol{I}}_t = (\widetilde{x}_t, \widetilde{y}_t)^\top$ as the solution produced by Algorithm 3 at the time slot t.

Algorithm 3: Algorithm 3 solves $\mathbb{P}_3$ on the fly using alternate primal and dual steps as time moves forward. Specifically, note that $\mathbb{P}_3$ can be equivalently written as

$$\min_{\boldsymbol{I}_t \in \mathcal{I}_t, \forall t} \max_{\lambda \geq 0} \sum_t \left(f_t(\boldsymbol{I}_t) + \lambda g_t(\boldsymbol{I}_t) \right),$$

where λ is the Lagrange multiplier. Denoting $\mathcal{L}_t(\boldsymbol{I}_t, \lambda) = f_t(\boldsymbol{I}_t) + \lambda g_t(\boldsymbol{I}_t)$, we alternate between (4) and (5):

$$\min_{\boldsymbol{I}_t \in \mathcal{I}_t} \nabla f_{t-1}(\tilde{\boldsymbol{I}}_{t-1})(\boldsymbol{I}_t - \tilde{\boldsymbol{I}}_{t-1}) + \lambda_t g_{t-1}(\boldsymbol{I}_t) + \frac{||\boldsymbol{I}_t - \tilde{\boldsymbol{I}}_{t-1}||^2}{2\alpha}, \tag{4}$$

$$\lambda_{t+1} = [\lambda_t + \mu \nabla_\lambda (\mathcal{L}_t(\tilde{\boldsymbol{I}}_t, \lambda_t))]^+ = [\lambda_t + \mu g_t(\tilde{\boldsymbol{I}}_t)]^+, \tag{5}$$

with the initial value $\tilde{\boldsymbol{I}}_0$, the initial value $\lambda_1 = 0$, and the preset step sizes α and μ. We perform the primal step by solving (4) efficiently via any standard convex optimization solver. Note that our primal descent step is a rectified variant which is different from that in standard primal-dual algorithms. In our version here, we directly penalize the constraint itself, instead of the first-order approximation, with the carefully-designed regularization term against the previous control decision, i.e., $\frac{||\boldsymbol{I}_t - \tilde{\boldsymbol{I}}_{t-1}||^2}{2\alpha}$. We perform the dual ascent step following the calculation as in (5), where we take $\tilde{\boldsymbol{I}}_t$ from the primal step.

We highlight that our design described in the above falls into the paradigm of *online learning*, which we leverage to control federated learning at each time slot t with provably guaranteed performance as shown in Section IV, without worrying about the long-term convergence requirement in $\mathbb{P}_3$'s constraint and the unknown future inputs beyond t in $\mathbb{P}_3$'s objective.

IV. Theoretical Analysis

We prove Theorems 1 and 2 as the motivation for Algorithm 2. We also state Propositions 1 and 2 and prove Theorem 3 to characterize the performance of all our algorithms.

Theorem 1. *Our problems $\mathbb{P}_{t,1}$ and $\mathbb{P}'_{t,1}$ are equivalent, i.e., their optimal solutions are the same.*

Proof. See Appendix A. To prove this equivalence, we first show that $\mathbb{P}'_{t,1}$ as a continuous linear program automatically has the discrete integer optimal solution due to total unimodularity [20], and then show that the linear objective function as in $\mathbb{P}'_{t,1}$ has preserved the optimal solution of $\mathbb{P}_{t,1}$ through separable convex functions and "λ-representation" [21]. □

Theorem 2. *Our traffic reduction function $\psi(\cdot)$ satisfies both monotonicity and submodularity: $\forall S \subset T \Rightarrow \psi(S) \leq \psi(T)$; and $\forall S \subset T, \forall j \notin T \Rightarrow \psi(S \cup \{j\}) - \psi(S) \geq \psi(T \cup \{j\}) - \psi(T)$.*

Proof. See Appendix B. To prove this, we exploit our previous Equation (1), and also analyze the different cases of the switch j versus the sets S and T. □

Proposition 1. *Algorithm 2 achieves an approximate ratio of $1 - 1/e$ for the problem $\mathbb{P}_{t,2}$ [22].*

Proposition 2. *Algorithm 3 achieves the regret and the fit for the problem $\mathbb{P}_3$ as follows [19]:*

$$\textstyle\sum_t \{f_t(\tilde{\boldsymbol{I}}_t) - f_t(\tilde{\boldsymbol{I}}^*_t)\} \leq \mathcal{O}(|\mathcal{T}|^{\alpha_1}), ||[\sum_t g_t(\tilde{\boldsymbol{I}}_t)]^+|| \leq \mathcal{O}(|\mathcal{T}|^{\alpha_2}),$$

where $\tilde{\boldsymbol{I}}_t$ is the control decision produced by Algorithm 3 at t; $\tilde{\boldsymbol{I}}^*_t \in \arg\min_{\boldsymbol{I} \in \{\boldsymbol{I} | g_t(\boldsymbol{I}) \leq 0; \boldsymbol{I} \in \mathcal{I}_t\}} f_t(\boldsymbol{I})$; and $\alpha_1, \alpha_2 \in (0, 1)$ are constants.

We skip the proofs of these two propositions as they directly follow our Theorem 2 and Algorithm 3, respectively, which have also been proved in existing literature as indicated.

Theorem 3. *Algorithm 1 achieves the following competitiveness, where θ and ξ are constants:*

$$\begin{aligned} E[\mathcal{P}(\{\tilde{x}_t, \tilde{y}_t, \bar{\boldsymbol{z}}_t, \bar{\boldsymbol{w}}_t\}, \forall t)] &= \mathcal{P}(\{\tilde{x}_t, \tilde{y}_t, \bar{\boldsymbol{z}}_t, \bar{\boldsymbol{w}}_t, \forall t\}) \\ &\leq \theta(1 + \xi) \cdot \mathcal{P}^*. \end{aligned}$$

Proof. See Appendix C, where we repetitively use the relation between the switching cost of each switching operation and the cumulative non-switching until the next switching operation, as enforced in Algorithm 1, and then handle the last switching operation in the entire time horizon separately. □

V. Experimental Evaluations

A. Evaluation Settings

Federated Learning: We adopt MNIST [26] and CIFAR-10 [27], which are among the most representative datasets for machine learning. MNIST contains 70k images, where we use 60k for training and 10k for testing; CIFAR-10 has 60k images, where we use 50k for training and 10k for testing. The task for MNIST is digit recognition and for CIFAR-10 is image classification. The training data are evenly distributed to the FL clients. We use TensorFlow Federated [25] to train a Support Vector Machine (SVM), and a Convolutional Neural Network (CNN) which consists of nine layers, including convolutional, max pooling, and local response normalization layers.

Network Topology: We construct a simulated network with 1,024 nodes using NetworkX [23] based on the U.S. backbone network [32]. For our evaluations, we envisage that every node in this network is a programmable switch. This network has 128 leaf nodes, where we use one of them to connect the FL server and all the rest to connect the FL clients. This is to simulate the scenario of using the cloud server as the FL server and the edge servers as the FL clients. We vary the number of the FL clients for each leaf node [10]: (i) *uniform*, where each leaf node randomly selects an integer from the range $[1, 9]$ as the number of the FL clients; (ii) *power-law*, where the numbers of the FL clients for the leaf nodes follow a power-law distribution within the integer range $(1, 63)$, with a variance of 97; (iii) *skewed*, where the numbers of the FL clients for the leaf nodes vary significantly; (iv) *mixed*, where each leaf node uses one of the above three strategies regarding the number of the FL clients and different leaf nodes may use different strategies. By default, we consider 4 available paths from each FL client, where each path to the FL server contains about 10 nodes en route with real-world path delay [24].

System Parameters: We implement an existing in-network aggregation protocol [7] on our real-world P4 device, i.e., a Wedge100BF switch. We can thus measure the average leading time $b \approx 3$s for the switch reconfiguration. We also have an Inspur Rack SN5160M4 edge server, equipped with a GeForce RTX 2080Ti GPU and an Intel(R) Xeon(R) Silver 4210R CPU, on which we measure the time-varying per-unit training computation latency. We get $a_t \approx 0.001$s on average for SVM

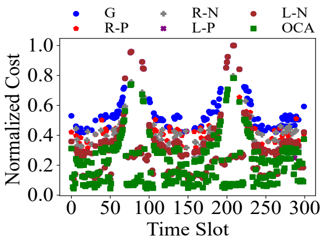


Fig. 2: Real-Time Cost

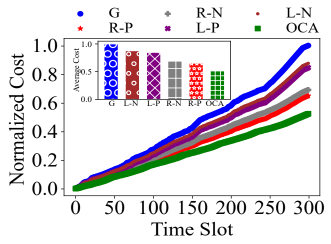


Fig. 3: Cumulative Cost

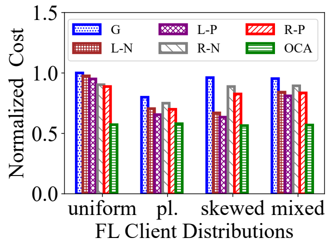


Fig. 4: Impact of Distribution

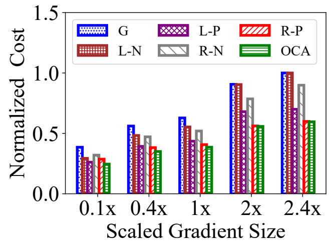


Fig. 5: Impact of Gradient Size

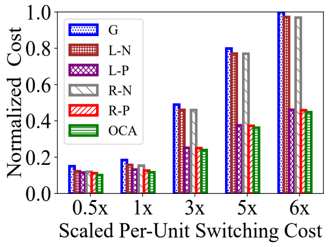


Fig. 6: Impact of Switching Cost

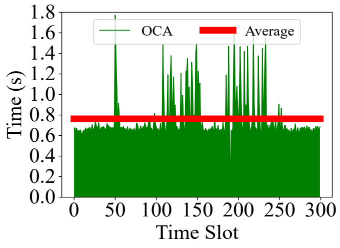


Fig. 7: Running Time

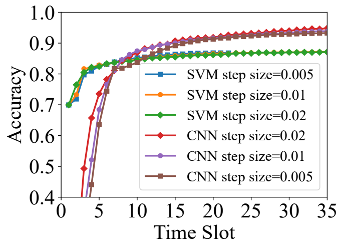


Fig. 8: MNIST Accuracy

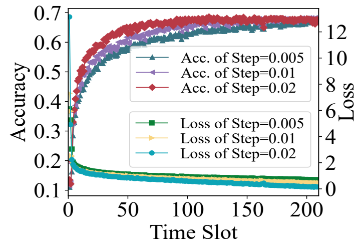


Fig. 9: CIFAR-10 Accuracy

and $a_t \approx 0.016$s on average for CNN. We set other parameters as $r_0 = 15$, $q_0 = 4$, $\varepsilon = 0.001$, $c = 500$ MB, $M_t = 128$, $\chi = 0.9$, and $\eta = 2$ [5]. As we have $1024/10 \approx 102$ paths on average, the value of M_t is set to cover all the paths.

Algorithms: Our proposed approach is represented as *OCA*, invoking the CVXPY [31] tool to solve the sub-problems as specified by our algorithms. For comparison, we implement the combinations of different algorithms as follows[1]. The following algorithms select switches for in-network aggregations:

- *L-*$*$ chooses leaf switches, i.e., the switches that directly connect the FL clients;
- *R-*$*$ chooses N switches randomly, where N is calculated by our online learning algorithm.

The following algorithms select paths for routing flows:

- $*$*-P* chooses the path for each flow from the paths that have been used before;
- $*$*-N* chooses the path with the best delay for each flow at each time slot.

We also consider a state-of-the-art FL algorithm:

- *G* represents the FedAvg algorithm [1], which ignores the effect or control of in-network aggregations.

B. Evaluation Results

Cost Advantage: Fig. 2 and Fig. 3 visualize the real-time total cost at each time slot and the cumulative total cost until each time slot, respectively, incurred by different algorithms. In the former figure, *OCA* outperforms others, with an average cost reduction of about 23%. Compared to *G*, *OCA* reduces at least 40% total cost. The cost fluctuations are mainly due to the dynamic path delay and per-unit training latency. In the latter figure, *OCA* possesses a slower growth than others.

Performance Consistency: Fig. 4, Fig. 5, and Fig. 6 depict the cumulative total cost over the entire time horizon for different approaches under different settings. In Fig. 4, for the skewed FL client distribution, compared to *L-** algorithms, the average cost reduction of *OCA* is only 10.1%. In the skewed distribution, some leaf switches connect to a large number of clients; then, *L-** that use such leaf switches for in-network aggregations achieve large traffic reduction. Even so, *OCA* consistently has the best performance with the average cost reduction of about 30.6%. In Fig. 5, as the size of the model and the gradient grows, the total cost grows. *OCA* achieves the average cost reduction of about 33%. In Fig. 6, as we increase the per-unit reconfiguration cost, the total cost grows. **-P* algorithms show the benefits when the unit reconfiguration cost is high, as they use the same paths (but may not use the same switches), resulting in only a slight reconfiguration cost.

Execution Efficiency: Fig. 7 exhibits the execution time of *OCA* at each time slot. The average execution time per time slot is merely 0.76s, which is sufficiently fast and efficient to meet the need in many real-world scenarios.

Inference Accuracy: Fig. 8 and Fig. 9 show the inference accuracy of the different models trained through *OCA*. As time goes, the accuracy grows. Both SVM and CNN models have high accuracy as expected on MNIST (SVM is not specifically designed for images). On CIFAR-10, we show the accuracy and the loss of CNN. Compared to MNIST, the classification task is more complex, leading to slower accuracy growth.

VI. Related Work

Federated Learning with In-Network Aggregation: Su *et al.* [12] designated FL clients to upload quantized model updates to programmable switches with limited memory for aggregation. Chen *et al.* [13] optimized FL over wireless mesh networks through model aggregation on routers with routing and spectrum allocation. Dinh *et al.* [14] minimized the FL aggregation latency with routing and resource management in multi-layer edge networks. Luo *et al.* [15] designed protocols for the edge nodes en route to eliminate duplicated model downloads and pre-aggregate gradients in FL. Sacco *et al.* [16] defined an in-band protocol to cache model parameters and

[1]The mark of * serves as a wildcard to match different switch selection algorithms and path selection algorithms.

adapted P4 switches to aggregate gradients for FL. Pan *et al.* [17] exploited both in-network and in-storage homomorphic encryption to expedite cross-silo FL aggregations.

This line of research could be the closest to our work. Yet, none of them have rigorously analyzed the traffic reduction impact in complex networks. Also, none have addressed the dynamic control of FL together with in-network aggregations and flow routing, not to mention reconfiguration cost incurred at programmable switches. Unlike our work, some have even overlooked the issue of attaining provable convergence in FL.

Distributed Training with In-Network Aggregation: Sapio *et al.* [6] designed programmable switch processing with end-host protocols and machine learning frameworks to accelerate distributed parallel training. Lao *et al.* [7] exploited in-network aggregation at rack switches in the multi-rack, multi-job distributed deep neural network training settings. Segal *et al.* [10] reduced network congestion via in-network traffic aggregation for distributed machine learning workloads. Qiu *et al.* [8] placed training tasks and selected switches for in-network aggregations using Steiner-Tree-based algorithms to reduce communication time and traffic. Bao *et al.* [18] controlled batch sizes and in-network aggregation locations to reduce training time. Liu *et al.* [9] addressed the case of multiple parameter servers, using in-network aggregations to improve the efficiency of distributed training.

These works are on distributed machine learning in general, rather than for FL specifically. Their settings are thus typically different from FL, and their models and formulations do not apply to FL with in-network aggregations. Their algorithms and methods are not directly applicable, either, because they have not captured the online, combinatorial, and lexicographic minimax nature of the problem as in our work.

VII. Conclusion

In-network aggregation provides a tremendous opportunity for service providers to address the traffic footprint and performance issues of federated learning. While most existing works have not systematically exploited it from a rigorous online algorithmic perspective, this paper fills the gap. We study the mathematical nature of this problem, and jointly schedule in-network aggregations, flow routing, and federated learning. We conduct both theoretical analysis and practical experiments to validate the advantages of our approach. For future work, we intend to continue to investigate in-network aggregations, with other related issues such as security and privacy.

Acknowledgment

This work was supported in part by the Nanjing University-China Mobile Communications Group Co., Ltd. Joint Institute, by the U.S. National Science Foundation (CNS-2047719 and CNS-2225949), by the National Natural Science Foundation of China (No. 62072115), and by the Shanghai Science and Technology Innovation Action Plan Project (No. 22510713600). The corresponding authors are Lei Jiao (ljiao2@uoregon.edu), Zhuzhong Qian (qzz@nju.edu.cn), and Ji Qi (qiji@cmss.chinamobile.com).

Appendix

A. Proof of Theorem 1

First, note the following proposition:

Proposition 3. *For the problem* $\min \boldsymbol{c}^\top \boldsymbol{x}, s.t.\ A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}$, *its optimal solution is integral if* A *is a totally unimodular matrix [20].*

For the matrix "A" from Constraint C_1, every entry is either 0 or 1, and every column has exactly one 1 entry. That is, A is totally unimodular [20]. Due to Proposition 3, the optimal solution of $\mathbb{P}'_{t,1}$ is integral.

Second, note the following two propositions:

Proposition 4. *For a problem with the* $\max$ *objective function, e.g.,* $\max\{r_1, r_2, ..., r_\kappa\}$, *the optimal solution keeps unchanged after replacing the objective by the separable convex functions, i.e.,* $\sum_{i=1}^{\kappa} \kappa^{r_i}$ *[21].*

Proposition 5. *Separable convex functions can be transformed into a single linear function via "λ-representation":* $f(x)$ *can be written as* $f(x) = \sum_{h\in\mathcal{H}} f(h)\lambda_h$, *where* $\sum_{h\in\mathcal{H}} h\lambda_h = x$, $\sum_{h\in\mathcal{H}} \lambda_h = 1$, *and* $\lambda_h \geq 0, \forall h \in \mathcal{H}$. $\mathcal{H}$ *is a set of all the possible values of* x *[21].*

Using Proposition 4, $\mathbb{P}_{t,1}$'s objective $\max_{i\in\mathcal{N}, p\in\mathcal{R}_i} w_{i,p,t}d_{p,t}$ can be transformed to $\sum_{i\in\mathcal{N}}\sum_{p\in\mathcal{R}_i}\{(\sum_{i\in\mathcal{N}}|\mathcal{R}_i|)^{w_{i,p,t}d_{p,t}}\}$. Afterwards, applying Proposition 5, we can write $\mathbb{P}_{t,1}$ as

$$\begin{aligned}
\max\quad & \textstyle\sum_{i\in\mathcal{N}}\sum_{p\in\mathcal{R}_i}\sum_{h\in\mathcal{H}}\{(\sum_{i\in\mathcal{N}}|\mathcal{R}_i|)^{h\cdot d_{p,t}}\lambda_{i,p,h}\}\\
s.t.\quad & \textstyle\sum_{h\in\mathcal{H}} h\lambda_{i,p,h} = w_{i,p,t}, \forall p\in\mathcal{R}_i, \forall i,\\
& \textstyle\sum_{h\in\mathcal{H}} \lambda_{i,p,h} = 1, \forall p\in\mathcal{R}_i, \forall i,\\
& C_1,\\
var.\quad & 0 \leq w_{i,p,t} \leq 1, \lambda_{i,p,h} \geq 0, \forall p\in\mathcal{R}_i, \forall i.
\end{aligned}$$

Due to $\mathcal{H} = \{0, 1\}$ in our case, the above formulation can be further expanded as the following:

$$\begin{aligned}
\max\quad & \textstyle\sum_{i\in\mathcal{N}}\sum_{p\in\mathcal{R}_i}\{1 - \lambda_{i,p,1} + (\sum_{i\in\mathcal{N}}|\mathcal{R}_i|)^{d_{p,t}}\lambda_{i,p,1}\}\\
s.t.\quad & \lambda_{i,p,1} = w_{i,p,t}, \forall p\in\mathcal{R}_i, \forall i,\\
& C_1,\\
var.\quad & 0 \leq w_{i,p,t} \leq 1, \lambda_{i,p,1} \geq 0, \forall p\in\mathcal{R}_i, \forall i.
\end{aligned}$$

After removing the constant 1 from the objective and the variable $\lambda_{i,p,1}$ from the entire formulation, we have $\mathbb{P}'_{t,1}$.

B. Proof of Theorem 2

First, we prove monotonicity. Note that $\psi(S) \leq \psi(S \cup \{j\})$ always holds, due to Equation (1). Then, $\forall S \subset T$, we take $T \setminus S \triangleq U = \{j_1, j_2, ..., j_{|U|}\}$. We thus have $\psi(S) \leq \psi(S \cup \{j_1\}) \leq \psi(S \cup \{j_1\} \cup \{j_2\}) \leq ... \leq \psi(S \cup U) = \psi(T)$.

Next, we prove submodularity. Consider a switch $j \notin T$, where j currently performs no in-network aggregation, and also consider $S \subset T$. One of the following cases must hold.

- All Q_j outgoing flows go into some switch k in S: $\psi(\{j\} \cup S) - \psi(S) = Q_j - 1 + (Q_k - (Q_j - 1)) - 1 + \psi(S \setminus \{k\}) - \psi(S) = \psi(S) - \psi(S) = 0$, and $\psi(\{j\} \cup T) - \psi(T) = Q_j - 1 + (Q_k - (Q_j - 1)) - 1 + \psi(T \setminus \{k\}) - \psi(T) = \psi(T) - \psi(T) = 0$;

- All Q_j outgoing flows go into some switch k in $T \setminus S$: $\psi(\{j\} \cup S) - \psi(S) = Q_j - 1 + \psi(S) - \psi(S) = Q_j - 1$, and $\psi(\{j\} \cup T) - \psi(T) = Q_j - 1 + (Q_k - (Q_j - 1)) - 1 + \psi(T \setminus \{k\}) - \psi(T) = \psi(T) - \psi(T) = 0$;
- None of Q_j outgoing flows go into T, or all other cases: $\psi(\{j\} \cup S) - \psi(S) = Q_j - 1 + \psi(S) - \psi(S) = Q_j - 1$, and $\psi(\{j\} \cup T) - \psi(T) = Q_j - 1 + \psi(T) - \psi(T) = Q_j - 1$.

That is, it is always $\psi(S \cup \{j\}) - \psi(S) \geq \psi(T \cup \{j\}) - \psi(T)$.

C. Proof of Theorem 3

In this proof, we define $\bar{\boldsymbol{I}}_t = \{\widetilde{x}_t, \bar{y}_t, \bar{z}_t, \bar{\boldsymbol{w}}_t\}$ as the solution at t produced by our online approach proposed in this paper, and $\widetilde{\boldsymbol{I}}_t = \{\widetilde{x}_t, \widetilde{y}_t, \bar{z}_t, \bar{\boldsymbol{w}}_t\}$ as the solution at t produced by our online approach before rounding $\widetilde{y}_t$ into $\bar{y}_t$. We define $\bar{\boldsymbol{I}}$ and $\widetilde{\boldsymbol{I}}$ as $\{\bar{\boldsymbol{I}}_t, \forall t\}$ and $\{\widetilde{\boldsymbol{I}}_t, \forall t\}$. We define $\boldsymbol{I}^*$ as the offline optimal solution to $\mathbb{P}$, i.e., $\mathcal{P}(\boldsymbol{I}^*) = \mathcal{P}^*$, and $\boldsymbol{I}^*_t$ as the part of $\boldsymbol{I}^*$ at t.

We use t'_w to represent the time slot when the w-th switching operation occurs. We denote $w \in \mathcal{W} \triangleq \{1, 2, ..., w'\}$, where w' refers to the last switching operation. From Algorithm 1, we know that the cumulative non-switching cost from t'_w to $t'_{w+1} - 1$ is at least η times the switching cost at t'_w:

$$C_S^{t'_w}(\widetilde{\boldsymbol{I}}_{t'_w}) \cdot \eta \leq \sum\nolimits_{t=t'_w}^{t'_{w+1}-1} C^t_{\neg S}(\widetilde{\boldsymbol{I}}_t). \tag{6}$$

Meanwhile, when a switching operation occurs at t, we have

$$\frac{C^t_S(\widetilde{\boldsymbol{I}}_t)}{C^t_{\neg S}(\widetilde{\boldsymbol{I}}_t)} \leq \frac{M_t b}{\kappa^L(\chi) \sum_{i \in \mathcal{N}} |\mathcal{D}_i| a_t} \triangleq \kappa_t. \tag{7}$$

First, we derive the following chain of inequalities:

$$\begin{aligned}
&\sum\nolimits_{t=1}^{|\mathcal{T}|} C^t_S(\widetilde{\boldsymbol{I}}_t) = \sum\nolimits_{w \in \mathcal{W} \setminus \{w'\}} C_S^{t'_w}(\widetilde{\boldsymbol{I}}_{t'_w}) + C_S^{t'_{w'}}(\widetilde{\boldsymbol{I}}_{t'_{w'}}) \\
&\overset{\text{(a)}}{\leq} \sum\nolimits_{w \in \mathcal{W} \setminus \{w'\}} \{\tfrac{1}{\eta} \sum\nolimits_{t=t'_w}^{t'_{w+1}-1} C^t_{\neg S}(\widetilde{\boldsymbol{I}}_t)\} + \kappa_{t'_{w'}} C_{\neg S}^{t'_{w'}}(\widetilde{\boldsymbol{I}}_{t'_{w'}}) \\
&\overset{\text{(b)}}{\leq} \sum\nolimits_{t=1}^{t'_1-1} C^t_{\neg S}(\widetilde{\boldsymbol{I}}_t) + \tfrac{1}{\eta} \sum\nolimits_{t=t'_1}^{t'_{w'}-1} C^t_{\neg S}(\widetilde{\boldsymbol{I}}_t) + \sum\nolimits_{t=t'_{w'}}^{|\mathcal{T}|} \kappa_t C^t_{\neg S}(\widetilde{\boldsymbol{I}}_t) \\
&\leq \max\{\tfrac{1}{\eta}, \max_t\{\kappa_t\}\} \cdot \sum\nolimits_{t=1}^{|\mathcal{T}|} C^t_{\neg S}(\widetilde{\boldsymbol{I}}_t) \triangleq \xi \cdot \sum\nolimits_{t=1}^{|\mathcal{T}|} C^t_{\neg S}(\widetilde{\boldsymbol{I}}_t),
\end{aligned}$$

where Inequality (a) holds due to Inequalities (6) and (7); and Inequality (b) holds as we add non-negative values to the sum.

Next, if we define $\theta = \max_t \frac{\max_{\boldsymbol{I}_t} C^t_{\neg S}(\boldsymbol{I}_t)}{\min_{\boldsymbol{I}_t} C^t_{\neg S}(\boldsymbol{I}_t)}$, then we have $C^t_{\neg S}(\widetilde{\boldsymbol{I}}_t) \leq \theta \cdot C^t_{\neg S}(\boldsymbol{I}^*_t)$, $\forall t$. Based on this, we have

$$\begin{aligned}
&E[\mathcal{P}(\bar{\boldsymbol{I}})] \overset{\text{(c)}}{=} \mathcal{P}(\widetilde{\boldsymbol{I}}) = \sum\nolimits_t (C^t_S(\widetilde{\boldsymbol{I}}_t) + C^t_{\neg S}(\widetilde{\boldsymbol{I}}_t)) \leq (1+\xi) \sum\nolimits_t C^t_{\neg S}(\widetilde{\boldsymbol{I}}_t) \\
&\leq \theta(1+\xi) \sum\nolimits_t C^t_{\neg S}(\boldsymbol{I}^*_t) \leq \theta(1+\xi)\mathcal{P}^*,
\end{aligned}$$

where Equality (c) holds due to our rounding step.

References

[1] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[2] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM*, 2019.

[3] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935–1949, 2021.

[4] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[5] Y. Jin, L. Jiao, M. Ji, Z. Qian, S. Zhang, N. Chen, and S. Lu, "Scheduling in-band network telemetry with convergence-preserving federated learning," *IEEE/ACM Transactions on Networking*, vol. 31, no. 5, pp. 2313–2328, 2023.

[6] A. Sapio, M. Canini, C.-Y. Ho, J. Nelson, P. Kalnis, C. Kim, A. Krishnamurthy, M. Moshref, D. Ports, and P. Richtárik, "Scaling distributed machine learning with in-network aggregation," in *USENIX NSDI*, 2021.

[7] C. Lao, Y. Le, K. Mahajan, Y. Chen, W. Wu, A. Akella, and M. Swift, "ATP: In-network aggregation for multi-tenant learning," in *USENIX NSDI*, 2021.

[8] Y. Qiu, G. Zhao, H. Xu, H. Huang, and C. Qiao, "PARING: Joint task placement and routing for distributed training with in-network aggregation," *IEEE/ACM Transactions on Networking*, 2024.

[9] J. Liu, Y. Zhai, G. Zhao, H. Xu, J. Fang, Z. Zeng, and Y. Zhu, "InArt: In-network aggregation with route selection for accelerating distributed training," in *ACM Web Conference*, 2024.

[10] R. Segal, C. Avin, and G. Scalosub, "Constrained in-network computing with low congestion in datacenter networks," in *IEEE INFOCOM*, 2022.

[11] L. Zeno, D. R. Ports, J. Nelson, D. Kim, S. Landau-Feibish, I. Keidar, A. Rinberg, A. Rashelbach, I. De-Paula, and M. Silberstein, "SwiSh: Distributed shared state abstractions for programmable switches," in *USENIX NSDI*, 2022.

[12] X. Su, Y. Zhou, L. Cui, and S. Guo, "Expediting in-network federated learning by voting-based consensus model compression," in *IEEE INFOCOM*, 2024.

[13] X. Chen, G. Zhu, Y. Deng, and Y. Fang, "Federated learning over multi-hop wireless networks with in-network aggregation," *IEEE Transactions on Wireless Communications*, vol. 21, no. 6, pp. 4622–4634, 2022.

[14] T. Q. Dinh, D. N. Nguyen, D. T. Hoang, T. V. Pham, and E. Dutkiewicz, "In-network computation for large-scale federated learning over wireless edge networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 5918–5932, 2022.

[15] S. Luo, P. Fan, H. Xing, L. Luo, and H. Yu, "Eliminating communication bottlenecks in cross-device federated learning with in-network processing at the edge," in *IEEE ICC*, 2022.

[16] A. Sacco, A. Angi, G. Marchetto, and F. Esposito, "P4FL: An architecture for federating learning with in-network processing," *IEEE Access*, vol. 11, pp. 103 650–103 658, 2023.

[17] X. Pan, Y. An, S. Liang, B. Mao, M. Zhang, Q. Li, M. Jung, and J. Zhang, "Flagger: Cooperative acceleration for large-scale cross-silo federated learning aggregation," in *ACM/IEEE ISCA*, 2024.

[18] J. Bao, G. Zhao, H. Xu, H. Wang, and P. Yang, "InGo: In-network aggregation routing with batch size adjustment for distributed training," in *IEEE/ACM IWQoS*, 2024.

[19] T. Chen, Q. Ling, and G. B. Giannakis, "An online convex optimization approach to proactive network resource allocation," *IEEE Transactions on Signal Processing*, vol. 65, no. 24, pp. 6350–6364, 2017.

[20] B. H. Korte, J. Vygen, B. Korte, and J. Vygen, *Combinatorial optimization*. Springer, 2011.

[21] Z. Hu, B. Li, and J. Luo, "Flutter: Scheduling tasks closer to data across geo-distributed datacenters," in *IEEE INFOCOM*, 2016.

[22] A. Krause and D. Golovin, "Submodular function maximization." *Tractability*, vol. 3, pp. 71–104, 2014.

[23] "NetworkX," https://networkx.org/, 2024.

[24] "Bandwidth from edges to aliyun," https://www.aliyun.com/, 2024.

[25] "TensorFlow federated," https://www.tensorflow.org/federated, 2024.

[26] "MNIST database," http://yann.lecun.com/exdb/mnist/, 2024.

[27] "CIFAR-10 and CIFAR-100 datasets," https://www.cs.toronto.edu/ kriz/cifar.html, 2024.

[28] R. Segal, C. Avin, and G. Scalosub, "SOAR: Minimizing network utilization with bounded in-network computing," in *ACM CoNEXT*, 2021.

[29] X. Yuan, L. Pu, L. Jiao, X. Wang, M. Yang, and J. Xu, "When computing power network meets distributed machine learning: An efficient federated split learning framework," in *IEEE/ACM IWQoS*, 2023.

[30] L. He, S. Wang, Y. Xu, P. Kuang, J. Cao, Y. Liu, X. Li, and S. Peng, "Enabling application-aware traffic engineering in ipv6 networks," *IEEE Network*, vol. 36, no. 2, pp. 42–49, 2022.

[31] "Welcome to CVXPY 1.5," https://www.cvxpy.org, 2024.

[32] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0—survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.