Federated Learning over Multi-Hop Wireless Networks with In-Network Aggregation

Xianhao Chen, Guangyu Zhu, Yiqin Deng, and Yuguang Fang, *Fellow, IEEE*

***Abstract*—Communication limitation at the edge is widely recognized as a major bottleneck for federated learning (FL). Multi-hop wireless networking provides a cost-effective solution to enhance service coverage and spectrum efficiency at the edge, which could facilitate large-scale and efficient machine learning (ML) model aggregation. However, FL over multi-hop wireless networks has rarely been investigated. In this paper, we optimize FL over wireless mesh networks by taking into account the heterogeneity in communication and computing resources at mesh routers and clients. We present a framework that each intermediate router performs *in-network* model aggregation before sending the data to the next hop, so as to reduce the outgoing data traffic and hence aggregate more models under limited communication resources. To accelerate model training, we formulate our optimization problem by jointly considering model aggregation, routing, and spectrum allocation. Although the problem is a non-convex mixed-integer nonlinear programming, we transform it into a mixed-integer linear programming (MILP), and develop a coarse-grained fixing procedure to solve it efficiently. Simulation results demonstrate the effectiveness of the solution approach, and the superiority of the in-network aggregation scheme over the counterpart without in-network aggregation.**

***Index Terms*—Federated learning, multi-hop wireless network, wireless mesh network, edge computing, in-network aggregation.**

I. Introduction

Machine learning (ML) lays the foundation for tremendous innovative applications, such as smart home, e-health, and autonomous driving. The success of ML rests on the unprecedented amount of data that can be used for model training. However, due to privacy considerations, it may be impractical to gather privacy-sensitive data, such as personal photos and videos, at a central location for training. To overcome this hurdle, federated learning (FL) has been proposed to train a global ML model across decentralized clients through model exchange without accessing clients' raw data [1]. By enabling privacy-preserving model training, FL is regarded as a key enabler for deploying ML at scale.

5G/6G cellular networks have attracted intensive interests in the last few years [2]–[4]. To unleash the potential of ML, 5G/6G and beyond are envisioned to support intelligence/computing services [5]–[7], including model inference and training, at the network edge. In FL, model training relies on frequent model exchange between parameter servers and a massive number of clients. Given the fact that a deep neural network generally contains millions of parameters (e.g., the popular model $VGG16$ contains 138 million parameters, which is 528MB in 32-bit float), the model transmissions would pose a great challenge to the wireless edge [8]. Therefore, to accelerate FL process, resource optimization for wireless networks plays a crucial role. The existing works in this field mostly focus on single-hop wireless networks, where clients upload their models to a base station co-located with an edge server for global model aggregation. However, given the limited transmit power, clients far from a base station may fail to upload models or suffer from low uplink data rate due to the severe path loss. To offer seamless services in a large region, wireless mesh network is a cost-effective and fast deployment choice. Compared with a single cell site, wireless mesh network can bring significant advantages like larger coverage, higher spectrum efficiency, and more adaptive/flexible network reconfigurability, which is deemed an important component of 5G/6G [9], [10]. In emerging heterogeneous networks (HetNets), multi-tier network entities, including macro/small base stations and relay stations, can naturally form a mesh of multi-hop wireless network [11]–[14]. In such a case, a macro base station can serve as the gateway base station (GBS), while the remaining network entities can be wirelessly interconnected to extend the coverage of the macro base station.

Implementing FL in wireless mesh networks opens some unique research opportunities. A key observation about FL is that, a central server is only interested in the aggregated model, and hence it may be unnecessary to transmit every local model to the central site if they can be aggregated beforehand. By leveraging in-network edge computing resource, a multi-hop network can perform model aggregation at each intermediate router before sending data to the next hop. This process, called "in-network aggregation" in this paper, can significantly reduce the outgoing data traffic from routers, and hence enable more efficient model aggregation over multi-hop networks under limited communication resources.

Fully exploiting the potential of in-network aggregation requires a holistic system design for multi-hop wireless networks. First of all, judicious spectrum allocation is required to mitigate network-wide wireless interference. Second, model routing must be optimized by considering model aggregation, which is completely different from traditional flow optimization problem with flow conservation (e.g., ingoing flow equals outgoing flow at intermediate routers). At last, one should

Xianhao Chen, Guangyu Zhu, and Yuguang Fang are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA. (e-mail: xianhaochen@ufl.edu, gzhu@ufl.edu, fang@ece.ufl.edu).

Yiqin Deng is with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: dengyiqin@csu.edu.cn).

This work was supported in part by US National Science Foundation under grants CNS-2106589 and IIS-1722791.

determine when to aggregate the models at each router by considering the communication and computing capabilities. These three issues are tightly intertwined, making the problem highly challenging. To tackle these challenges, in this paper, we jointly optimize spectrum allocation, routing, and in-network aggregation to accelerate FL training over wireless mesh networks. To our best knowledge, FL over wireless mesh networks has seldom been studied. The closest work to ours is [15], where multi-agent reinforcement learning is employed to accelerate the FL convergence over wireless mesh networks by learning a routing strategy to minimize the average end-to-end model transmission delay. However, in-network aggregation has not been considered in their work. Besides, they devise a routing scheme operating on contention-based wireless technology (e.g., 802.11 protocols), whereas we aim to design a centralized joint routing and spectrum allocation strategy by taking full advantage of the global knowledge of GBS.

The major contributions of this paper are summarized as follows.

- We propose an FL framework over wireless mesh networks with in-network aggregation. Specifically, our protocol sets a deadline for model download, update, and upload in each training round, and aims to maximize the aggregated models before the deadline. This objective is based on the observation that selecting more clients in each round can improve the training performance in FL [1], [16]. To fully reap the benefits of the multi-hop architecture, we jointly optimize routing, spectrum allocation, and in-network aggregation. As far as we know, this is the first FL framework over wireless mesh networks with in-network aggregation.
- We formulate our problem as a non-convex mixed-integer nonlinear programming, where the non-convexity arises from the in-network aggregation constraints. We transform the problem into a mixed-integer linear programming (MILP), and develop a coarse-grained fixing algorithm to obtain sub-optimal solution efficiently.
- Simulation results demonstrate the effectiveness of the proposed coarse-grained fixing algorithm by using the optimal solution to the formulated MILP as the benchmark, and show that in-network aggregation can significantly improve the training performance comparing to the counterpart without in-network aggregation.

The remainder of this paper is organized as follows. Section II introduces the related work. Section III presents the system architecture and federated learning training process. We formulate the optimization problem in Section IV, and offer the solution approach in Section V. In Section VI, we extend the formulation to consider traditional multi-hop relaying without in-network aggregation. Section VII conducts performance evaluation. We conclude the paper in the last section.

II. Related Work

A number of recent works investigate FL implementations over wireless networks. To reduce the training time to achieve a desired learning accuracy, client selection and radio resource management is the key. In [17], Chen et al. propose a joint learning and communication resource allocation scheme to minimize the FL loss by considering the packet errors introduced by wireless channels. In [18], Ren et al. develop a joint batchsize selection and communication resource allocation strategy for FL to optimize a novel criterion, called learning efficiency. In [19], Yang et al. address the joint communication and computing resource allocation problem for FL, with the objective to minimize the total energy consumption under latency constraint. In [16], Nishio et al. devise a client selection policy to aggregate as many model updates as possible within predefined deadline subject to the heterogeneous clients' computing and communication constraints. Unfortunately, all aforementioned works focus on FL implementation just for a single base station.

In FL, multi-level or multi-hop aggregation plays a pivotal role in increasing coverage and reducing the data traffic from a large set of clients [15], [20]–[24]. In [20], Liu et al. propose a two-level cloud-edge-client federated learning architecture and analyze its convergence property. Under similar two-level scenarios, the resource allocation problems are studied in [21] and [22]. In [23], Chen et al. propose a collaborative FL mechanism where local devices transmit models to nearby devices via device-to-device communications, and directly perform model aggregation on local devices. However, the above schemes focus on special or layered network structure, which cannot be applied to general mesh networks. Besides, routing has not been addressed in their works. In [15], Pinyoanuntapong et al. employ multi-agent reinforcement learning to learn the routing strategy to minimize the average end-to-end model transmission latency over wireless mesh networks while failing to consider in-network aggregation and spectrum allocation.

The design of data aggregation scheme can be traced back to the studies on wireless sensor networks (WSNs). Analogous to FL, a sink node in a WSN may require an aggregated form of sensed data, e.g., temperature, where in-network computation can also greatly reduce the communication overhead. Similar to our work, the problem on maximizing the aggregated information within given deadline in WSNs has been addressed in several works [25], [26]. Unfortunately, the existing data aggregation schemes for WSNs cannot be applied to FL. First, FL has the crucial processes of *global model download* and *local model update*, which do not exist in the data aggregation problem for WSNs. Due to the heterogeneous channel conditions and computing resources of clients, the global model download and local model update times can be vastly different across clients, which greatly affect the optimal client/router selection, routing, and resource allocation strategy. Second, the existing schematic design on WSNs commonly ignores the in-network aggregation delay. This assumption may be reasonable for WSNs, since the operations on sensor information, e.g., temperature, is typically simple. Nevertheless, given the fact that state-of-the-art deep neural networks (DNNs) generally contain millions of parameters, in-network aggregation delay is generally non-negligible in FL, which cannot be simply neglected.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2022.3168538, IEEE Transactions on Wireless Communications

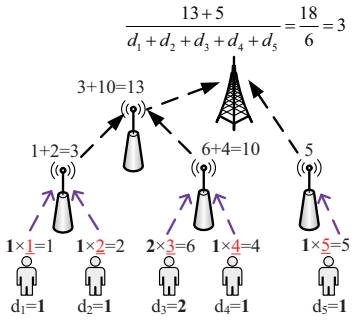


Fig. 1: Illustration of in-network aggregation over a toy mesh network. Each red underlined number represents a specific parameter of a client's model, and each bold number is the size of a client's dataset, i.e., $d_c = |\mathcal{D}_c|$. Although an ML model is composed of many parameters, we just take one parameter, i.e., the red underlined number, for the ease of exposition. Each PoC^2 sums up its received models before sending to the next hop. GBS sums up all the received models and divides the aggregated model by the total number of data samples.

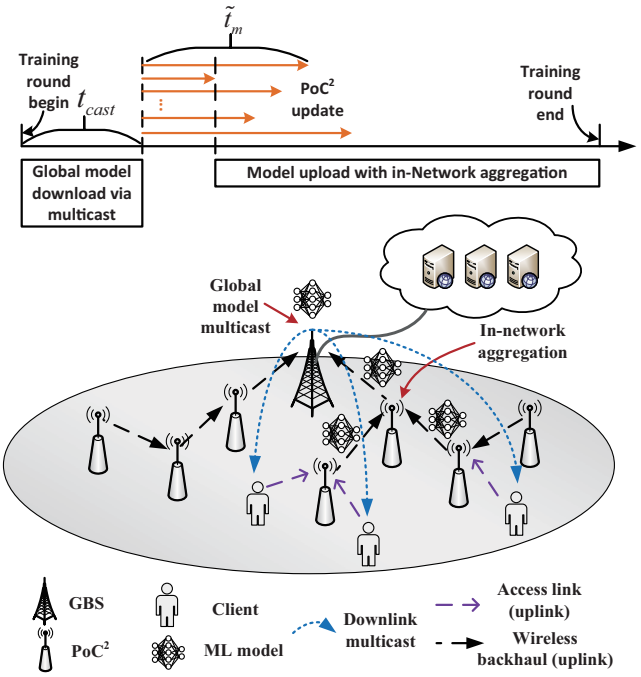


Fig. 2: Federated learning over multi-hop wireless networks. t_{cast} is the model multicasting time, and $\tilde{t}_m$ is the PoC^2 m's update time (explained in Section III-C).

III. System Model

A. Federated Learning and In-network Aggregation

Assume that we have a set $\mathcal{C}$ of clients with local datasets $\mathcal{D}_c$ for $c \in \mathcal{C}$ in the considered region. An FL server aims to train a global model using the data available at these clients. Considering supervised ML, a training data sample j consists of input sample x_j and label y_j. Let $l_j(\boldsymbol{\theta}) \triangleq l(\boldsymbol{\theta}, x_j, y_j)$ represent the loss function on data sample j based on model $\boldsymbol{\theta}$. For client $c \in \mathcal{C}$, the loss function on its dataset $\mathcal{D}_c$ is given by

$$L_c(\boldsymbol{\theta}) \triangleq \frac{1}{|\mathcal{D}_c|} \sum_{j \in \mathcal{D}_c} l_j(\boldsymbol{\theta}). \tag{1}$$

The global loss function over all the local datasets is defined as

$$L(\boldsymbol{\theta}) \triangleq \frac{1}{|\mathcal{D}|} \sum_{c \in \mathcal{C}} |\mathcal{D}_c| L_c(\boldsymbol{\theta}), \tag{2}$$

where $\mathcal{D} \triangleq \cup_{c \in \mathcal{C}} \mathcal{D}_c$.

To minimize the global loss function $L(\boldsymbol{\theta})$, the standard federated learning algorithm $FedAvg$ proceeds iteratively as follows [1]. In each training round, each client downloads the global model from the central server. Each client updates the model based on its local dataset for several epochs, typically relying on gradient-descent algorithms, and then uploads the model to the central server. Upon receiving all the desired model updates, the central server sets the global model to the weighted average of these models:

$$\boldsymbol{\theta} = \frac{1}{|\mathcal{D}|} \sum_{c \in \mathcal{C}} |\mathcal{D}_c| \boldsymbol{\theta}_c, \tag{3}$$

where $\boldsymbol{\theta}_c$ represents the local model updated by client c. In practice, it may not be feasible to select all the clients in the every training round because of the limited communication and computing resource as well as clients' activities. Defining $\mathcal{S}$ as the set of selected clients in the considered round, the global model can still be updated according to (3) by aggregating only the models from the selected clients, i.e., replacing $\mathcal{C}$ with $\mathcal{S}$ and letting $\mathcal{D} \triangleq \cup_{c \in \mathcal{S}} \mathcal{D}_c$ in (3).

Since the central server is interested in the model average rather than each individual model update, a multi-hop network can aggregate the models in each intermediate router to reduce the data traffic. Specifically, client c can scale up the model update to $|\mathcal{D}_c|\boldsymbol{\theta}_c$, and upload to the associated router. Each router aggregates the received models, from clients and other routers, into one model by summing up the model parameters, then forwards to the next hop. Finally, the central server aggregates the models from routers, and scales the aggregated model by $\frac{1}{|\cup_{c \in \mathcal{S}} \mathcal{D}_c|} = \frac{1}{\sum_{c \in \mathcal{S}} |\mathcal{D}_c|}$ to recover (3). While an ML model is composed of many parameters, we can take one parameter as an example for ease of exposition. Suppose that the central server desires the weighted average $\frac{d_1*1+d_2*2+d_3*3+d_4*4+d_5*5}{d_1+d_2+d_3+d_4+d_5} = \frac{18}{6} = 3$ from five clients holding parameter 1, 2, 3, 4, and 5, respectively, where $d_1 = 1$, $d_2 = 1$, $d_3 = 2$, $d_4 = 1$, and $d_5 = 1$ are the sizes $|\mathcal{D}_c|$ of clients' local datasets. As illustrated in Fig. 1, the central server can exactly recover the desired result 3 via in-network aggregation over the toy mesh network.

B. Network Architecture

Let us consider a static wireless mesh network with one GBS g and a set $\mathcal{R}$ of wireless routers, as shown in Fig. 2. These routers, also called points of communication and computing (PoC^2s), are set-top like devices endowed with customized communication (e.g., software-defined radio), computing, and

TABLE I: Frequently used nomenclature and notations

Notation	Description
PoC^2	Point of communication and computing
$\mathcal{D}_c$	The dataset of client c
$\mathcal{R}$	The set of PoC^2s
$\mathcal{S}_m$	The clients selected by PoC^2 m
g	The gateway base station (GBS)
$\mathcal{B}_m$	The set of backhaul channels around node m
W	The bandwidth for each backhaul channel
W_{cast}	The bandwidth for the downlink multicast channel
Γ_m	The transmission set of PoC^2 m
$\mathcal{I}_m$	The interference set of PoC^2 m
c_m/c_g	The time taken by PoC^2 m/GBS to sum up two models
$e_{m,n}$	The capacity of a channel over link (m,n)
$\tilde{t}_m$	The time for PoC^2 m to collect and aggregate set $\mathcal{S}_m$ of clients' models
D	The length of a training round
Δ	The length of a control interval
K	Number of control intervals in a training round
a_m, $r_{m,n}$, $s^{b,k}_{m,n}$, $f^{b,k}_{m,n}$, t_m, t^{cast}	Decision variables (explained in Section IV)
v^k_m, h^k_n, $y_{m,n}$, b^k_m, d^k_n, $u_{m,n}$	Auxiliary variables (to linearize **P1**)

storage capabilities, or CCS capabilities for short. GBS is typically a macro base station endowed with significantly CCS capabilities for global model aggregation. GBS also acts as a central controller to coordinate PoC^2s for model delivery and aggregation via multi-hop wireless links.

At the beginning of each training round, GBS multicasts a global model to a set of clients for model training. After local training at client side, PoC^2s aggregate models from their associated clients and then deliver the models to GBS via multi-hop relaying. Each PoC^2 aggregates models from its associated clients and other PoC^2s before sending to the next hop, which is the in-network aggregation process. The clients, PoC^2s, and GBS are wirelessly interconnected. However, we assume that the backhaul links (among PoC^2s and GBS) and access links (from clients to PoC^2s) use different set of channels, and hence do not interfere with each other. Moreover, we assume that each client is pre-associated with one PoC^2. In this way, we can concentrate on the multi-hop wireless backhauling in this study, while excluding clients and access links from our network topology. We want to first add some explanations for the nodes (PoC^2s and GBS), links, and channels in the wireless backhaul part as follows. Let $\mathcal{B}_m$ represent the set of channels available around node $m \in \{g\} \cup \mathcal{R}$, and (m,n) denote the uplink wireless link from node $m \in \mathcal{R}$ to node $n \in \{g\} \cup \mathcal{R}$. The channels in $\mathcal{B}_m \cap \mathcal{B}_n$ can be utilized to support model transmission between node m and node n. Since the downlink multicasting from GBS to clients is carried out before model upload, it does not interfere with $\mathcal{B}_m$ around each PoC^2 for model upload. Thus, we do not need to specify the channel used by GBS for downlink multicasting, but rather simply assume that the multicast bandwidth is W_{cast}.

By adopting the deterministic power propagation model, we characterize the channel gain for link (m,n) on channel b as $\gamma_{m,n} = \eta d^{-\alpha}_{m,n}$, where η is an antenna related constant, $d_{m,n}$ is the distance between node m and node n, and α is the path loss exponent [27][1]. Each channel is assumed to be additive white Gaussian noise (AWGN) channel with fixed noise power. Since all transmitted data is assumed to be the models of the same size for the analysis, we can treat a model as the basic unit of the transmitted data in our systems. Therefore, we normalize channel capacity and other parameters related to data rate/amount by model size M for notational convenience. The normalized capacity of a channel over link (m,n) is given by

$$e_{m,n} = \frac{W}{M} \log\big(1 + \frac{\gamma_{m,n} p}{W N_0}\big), \tag{4}$$

where W is the bandwidth of each channel, p is PoC^2's transmit power on each channel, N_0 is the noise power spectral density, and M is the ML model size. We denote by Γ_m and $\mathcal{I}_m$ the *transmission set* and *interference set* of PoC^2 m, respectively, where Γ_m and $\mathcal{I}_m$ contain the nodes within a transmission range R and a potentially larger interference range R' of PoC^2 m. Following the well-known protocol interference model [28], we assume that link (m,n) exists only if node n is in Γ_m, and the interference from node m to node n exists only if node n is in $\mathcal{I}_m$. For readers' convenience, the frequently used notations are summarized in Table I.

C. Federated Learning Training Process

We next present our FL training process, which consists of multiple training round. Each training round involves four stages, i.e., coordination, model download, PoC^2 update, and model upload.

In the coordination stage, GBS collects the channel state information (CSI) for wireless backhaul links (m,n), downlink channels to clients (for downlink multicasting), and the available computing resource at each PoC^2 (for measuring in-network aggregation delay). Meanwhile, each PoC^2 collects the information about its associated clients, including their CSI and available computing resource. Based on the information, each PoC^2 m computes the set $\mathcal{S}_m$ of selected clients, and the needed time $\tilde{t}_m$ for collecting and aggregating these clients' models (called PoC^2 *update time*), and reports $\mathcal{S}_m$ and $\tilde{t}_m$ to GBS. Finally, GBS makes the centralized coordination (e.g., PoC^2 selection, routing, and link scheduling) and informs the decisions to PoC^2s for execution. Due to heterogeneous computing and communication resources at PoC^2s and clients, $\tilde{t}_m$ could be vastly different across PoC^2s, as depicted in Fig. 2.

[1]For simplicity, we assume the same gain model for all channels.

A key process above is that a PoC^2 needs to compute $\mathcal{S}_m$ and $\tilde{t}_m$. By assuming that proximate PoC^2s are assigned with different access bands and thus do not interfere with each other when collecting models from clients, each PoC^2 can implement some existing single-cell FL policies to obtain $\mathcal{S}_m$ and $\tilde{t}_m$. For instance, the FedCS protocol proposed in the well-cited paper [16], allowing a PoC^2 to aggregate as many models as possible before a predefined deadline, can be adopted here to obtain $\mathcal{S}_m$ and $\tilde{t}_m$.

In the model download stage, GBS multicasts the global model to the *selected clients*. Note that our policy will only select PoC^2s, each of which has the autonomy of selecting clients to update and upload models based on its local situations. Therefore, the selected clients are those selected by the *selected* PoC^2*s*. In other words, any client associated with the unselected PoC^2s would not participate or receive the global model in the current round. In addition, we have made the implicit assumption that clients are in the downlink coverage of GBS. Given the fact that a GBS typically has powerful communication capabilities (e.g., high transmit power), we believe that this assumption is reasonable in many cases.

In the PoC^2 update stage, the set $\mathcal{S}_m$ of selected clients perform local model update and upload the models to PoC^2 m. Then, each PoC^2 aggregates the received clients' models. As discussed before, some existing single-cell FL protocols can be used at this stage.

In the model upload stage, the PoC^2s collaboratively deliver their aggregated models to GBS. Upon receiving the models from other PoC^2, a PoC^2 performs model aggregation before sending to the next hop. Note that the unselected PoC^2s could also facilitate the multi-hop model delivery, even though they do not directly collect models from clients.

IV. Problem Formulation

Since the aforementioned four stages in one training round will be repeated until the global model converges to a desired accuracy, we focus on one training round for analysis. In each round, being consistent with [16], [29], our design goal is to *aggregate as many model updates as possible at GBS given a predefined deadline* D *and resource constraints*. This objective is motivated by the key observation that increasing the number of participating clients can accelerate the training process in FL [1], [16], [29]–[31].

We start by introducing the network resource constraints. Since ML models are gradually moved from source nodes to GBS, the link traffic demands are not static over time, implying that a static resource allocation strategy may not be able to fully exploit the network resources. Therefore, we assume that a training round consist of K control intervals, and different resource allocations can be made across the control intervals to accommodate the time-varying traffic. We define resource tuple $((m,n),b,k)$ to indicate that over link (m,n), transmitter m operates on channel b during the k-th control interval $[(k-1)\Delta, k\Delta]$, where Δ is the length of a control interval. For clarity, the decision variables and definitions are summarized below.

- $a_m \in \{0,1\}$ is the PoC^2 selection variable, where $a_m = 1$ if PoC^2 m is selected, and 0 otherwise. As explained before, a selected PoC^2 will collect its clients' models, whereas an unselected PoC^2 will not.
- $r_{m,n} \in \{0,1\}$ is the routing decision variable, where $r_{m,n} = 1$ if PoC^2 m forwards the data to node n (PoC^2 or GBS), and 0 otherwise.
- $s^{b,k}_{m,n} \in \{0,1\}$ denotes the resource allocation strategy, where $s^{b,k}_{m,n} = 1$ if resource tuple $((m,n),b,k)$ is active, and 0 otherwise.
- $f^{b,k}_{m,n} \geq 0$ represents the amount of data flow (normalized by model size) delivered via resource tuple $\big((m,n),b,k\big)$.
- $t_m \in [0,D]$ indicates the aggregation time that node m (PoC^2 or GBS) stops receiving models from other PoC^2s and begins in-network aggregation.
- $t^{\text{cast}} \in [0,D]$ represents the time for downlink model multicasting.

We will use boldfaced $\boldsymbol{a}$, $\boldsymbol{r}$, $\boldsymbol{s}$, $\boldsymbol{f}$, and $\boldsymbol{t}$ to denote the collections of a_m, $r_{m,n}$, $s^{b,k}_{m,n}$, $f^{b,k}_{m,n}$, and t_m, respectively.

A. Routing Constraints

For simplicity, we study a single-path routing scheme where a node can have at most one outgoing link [32]. In other words, a PoC^2 cannot partition its aggregated model into multiple data pieces for sending to multiple receivers[2]. We have

$$\sum_{n\in\Gamma_m} r_{m,n} \leq 1, \forall m \in \mathcal{R}. \tag{5}$$

Besides, PoC^2 m must transmit if receiving the models from other PoC^2s or selected by GBS ($a_m = 1$), leading to

$$\sum_{n\in\Gamma_m} r_{m,n} \geq r_{m',m}, \quad \forall m \in \mathcal{R}, m' \in \{m' | m \in \Gamma_{m'}\}, \tag{6}$$

$$\sum_{n\in\Gamma_m} r_{m,n} \geq a_m, \quad \forall m \in \mathcal{R}. \tag{7}$$

For the sake of simplicity, we omit the constraint on storage by assuming that each PoC^2 has enough storage space to cache the received models for aggregation. Since PoC^2 m aggregates the received models (from clients and/or other PoC^2s) into one model before transmitting to the next hop, the outgoing link (m,n) must deliver exactly one model over the total K control intervals if routing decision $r_{m,n} = 1$ is made, yielding

$$\sum_{b\in\mathcal{B}_m\cap\mathcal{B}_n} \sum_{k=1}^{K} f^{b,k}_{m,n} = r_{m,n}, \quad \forall m \in \mathcal{R}, n \in \Gamma_m, \tag{8}$$

where the flow amount $f^{b,k}_{m,n}$ is normalized by model size. $f^{b,k}_{m,n}$ is constrained by spectrum allocation and aggregation time decisions for node m and n, which will be introduced in the next subsection. (8) also ensures that there is no "routing loop", as the flow amount $f^{b,k}_{m,n}$ (and therefore $r_{m,n}$) can be positive only if aggregation time $t_n > t_m$, as can be seen later. Since there is no loop in a route and only GBS g is not required to transmit after receiving models, all the models from selected PoC^2s can be eventually transferred to GBS.

[2]While multi-path routing may take advantage of load balancing to improve throughput, it would introduce more complexity when taking in-network aggregation into account. Hence, we will not investigate it in this paper.

B. Aggregation Time and Flow Constraints

At the beginning, GBS multicasts a global model to a set of clients. To enable multicasting, GBS should employ the modulation and coding scheme in accordance with the client experiencing the worst propagation condition. Let e_m^{DL} be the normalized downlink channel capacity (normalized by model size M) from GBS to the weakest client (with the lowest downlink signal-to-noise ratio (SNR)) in $\mathcal{S}_m$, i.e.,

$$e_m^{\text{DL}} = \frac{W_{\text{cast}} \log\left(1 + \frac{\min_{c\in\mathcal{S}_m}\{\gamma_{g,c}\}P}{W_{\text{cast}}N_0}\right)}{M}, \tag{9}$$

where $\gamma_{g,c}$ is the channel gain from GBS to client c, P is GBS's downlink transmit power, and W_{cast} is the multicasting bandwidth. Since multicasting is carried out before model upload, GBS can re-utilize the backhaul channels to conduct the downlink transmissions. In this case, we have $W_{\text{cast}} = W|\mathcal{B}_g|$, recalling that $\mathcal{B}_g$ is the set of channels available for GBS, and W is the channel bandwidth. Furthermore, the multicasting time t^{cast} is determined by the weakest client among all the clients selected by different PoC^2s, yielding

$$t^{\text{cast}} \geq \frac{1}{e_m^{\text{DL}}} a_m, \quad m \in \mathcal{R}. \tag{10}$$

Thus, t^{cast} depends on PoC^2 selection variables a_m.

After receiving the global model, local update is performed by each PoC^2. PoC^2 m, if selected by the system, should receive the models from $|\mathcal{S}_m|$ clients, and aggregate these local models into one model, which demands $\tilde{t}_m$ (i.e. PoC^2 update time) in total. Consequently, the aggregation time t_m for PoC^2 m must satisfy

$$t_m \geq t^{\text{cast}} + \tilde{t}_m a_m, \quad m \in \mathcal{R}. \tag{11}$$

We use in-network aggregation delay $\sigma_m(\boldsymbol{a}, \boldsymbol{r})$ to denote the backhaul-level model aggregation delay (not including the time PoC^2 m aggregates the local models from $\mathcal{S}_m$, which has already been considered in $\tilde{t}_m$), i.e.,

$$\sigma_m(\boldsymbol{a}, \boldsymbol{r}) = \left(\sum_{\{m'|m\in\Gamma^I_{m'}\}} r_{m',m} + a_m - 1 \right)^+ c_m, \quad \forall m \in \mathcal{R}, \tag{12}$$

where c_m represents the computation delay for summing up two models, and x^+ denotes the positive part of x, i.e., $\max\{x, 0\}$. The number of models received by PoC^2 m is equal to $\sum_{\{m'|m\in\Gamma_{m'}\}} r_{m',m} + a_m$, i.e., the number of routing decisions made towards it, plus one if the PoC^2 is selected (i.e., plus the aggregated local model). The number of summations is equal to the number of received models minus one. The positive part operator x^+ ensures that $\sigma_m(\boldsymbol{a}, \boldsymbol{r}) = 0$ if node m receives no model.

Since a PoC^2 aggregates all received models into one model, there is no benefit for it to transmit before completing the aggregation. Therefore, PoC^2 m is allowed to transmit only after completing in-network aggregation, namely, after $t_m + \sigma_m(\boldsymbol{a}, \boldsymbol{r})$. Moreover, link (m, n) cannot carry data flow after t_n, i.e., the time that receiver n stops receiving models. These conditions impose the following constraints on how much data traffic can be carried over each resource tuple:

$$f_{m,n}^{b,k} \leq s_{m,n}^{b,k} e_{m,n} \Delta, \tag{13}$$

$$f_{m,n}^{b,k} \leq e_{m,n}\Big(\Delta k - \big(t_m + \sigma_m(\boldsymbol{a}, \boldsymbol{r})\big)\Big)^+, \tag{14}$$

$$f_{m,n}^{b,k} \leq e_{m,n}\big(t_n - \Delta(k-1)\big)^+, \tag{15}$$

$$f_{m,n}^{b,k} \leq e_{m,n}\Big(t_n - \big(t_m + \sigma_m(\boldsymbol{a}, \boldsymbol{r})\big)\Big)^+, \tag{16}$$

$$\forall m \in \mathcal{R}, n \in \Gamma_m, b \in \mathcal{B}_m \cap \mathcal{B}_n, k \in \{1, ..., K\}.$$

As indicated in (13), the data flow $f_{m,n}^{b,k}$ is upper bounded by $e_{m,n}\Delta$ if the resource tuple $s_{m,n}^{b,k}$ is allocated, i.e., $s_{m,n}^{b,k} = 1$, and 0 otherwise, where $e_{m,n}$ is the link capacity given in (4) and Δ is the length of the k-th control interval $[(k-1)\Delta, k\Delta]$. Moreover, $f_{m,n}^{b,k}$ is further constrained by time $t_m + \sigma_m(\boldsymbol{a}, \boldsymbol{r})$ and t_n, since the transmission can only be conducted between these two time instants. Overall, (13)-(16) together ensure that the data transmissions over resource tuple $s_{m,n}^{b,k}$ can only be conducted during the intersection of the k-th control interval $[(k-1)\Delta, k\Delta]$ and the time interval $[t_m + \sigma_m(\boldsymbol{a}, \boldsymbol{r}), t_n]$.

During the model upload stage, GBS does not transmit any models. Therefore, the optimal t_g (for stopping receiving models) can be set to the deadline D minus the time that it should take to perform global aggregation:

$$t_g = D - \bar{c} - \Big(\sum_{\{m'|g\in\Gamma_{m'}\}} r_{m',g} - 1\Big) c_g, \tag{17}$$

where $\bar{c}$ is the time taken by GBS to scale the aggregated model by $\frac{1}{\sum_{c\in\mathcal{S}}|\mathcal{D}_c|}$, and c_g is the time for GBS to sum up two models. The last term indicates that the number of summation operations will be equal to the number of received models minus one, if GBS receives at least one model (which is the non-trivial case).

C. Resource Allocation Constraints

The aforementioned routing and in-network aggregation rely on appropriate network resource allocation. In our formulation, the resource allocation strategy is characterized by the time-frequency resource tuples $s_{m,n}^{b,k}$. Due to the conflict relationship, not all the resource tuples can be chosen together. A node is not allowed to either receive from or transmit to multiple nodes on the same channel in an interval, which is expressed as

$$\sum_{\{n\in\Gamma_m|b\in\mathcal{B}_n\}} s_{m,n}^{b,k} \leq 1, \tag{18}$$

$$\sum_{\{m'|m\in\Gamma_{m'}, b\in\mathcal{B}_{m'}\}} s_{m',m}^{b,k} \leq 1,$$

$$\forall m \in \mathcal{R} \cup \{g\}, b \in \mathcal{B}_m, k \in \{1, ..., K\}, \tag{19}$$

To avoid self-interference, a PoC^2 cannot use a single channel for transmission and reception simultaneously, yielding

$$s_{m',m}^{b,k} + \sum_{\{n\in\Gamma_m|b\in\mathcal{B}_n\}} s_{m,n}^{b,k} \leq 1, \forall m \in \mathcal{R},$$

$$b \in \mathcal{B}_m, m' = \{m'|m \in \Gamma_{m'}, b \in \mathcal{B}_{m'}\}, k \in \{1, ..., K\}. \tag{20}$$

A feasible scheduling should also ensure that there is no interference (in the sense of the protocol interference model) among the nodes. Specifically, if PoC^2 m is transmitting to node n, other PoC^2s that have node n within their interference ranges cannot use the same channel and time resource. We therefore have

$$s_{m,n}^{b,k} + \sum_{\{n' \in \Gamma_{m'} | b \in \mathcal{B}_{n'}\}} s_{m',n'}^{b,k} \leq 1, \quad \forall m \in \mathcal{R}, n \in \Gamma_m, b \in \mathcal{B}_m \cap \mathcal{B}_n, m' \in \Upsilon_n^b, k \in \{1, ..., K\}, \tag{21}$$

where $\Upsilon_n^b = \{m' \in \mathcal{R} | n \in \mathcal{I}_{m'}, b \in \mathcal{B}_{m'}, m' \neq m\}$ denotes the set of PoC^2s having receiver n within their interference ranges and with channel b available.

In summary, given the constraints above, our optimization problem, which jointly optimizes resource allocation $s_{m,n}^{b,k}$, routing decision $r_{m,n}$, PoC^2 selection a_m, flow allocation $f_{m,n}^{b,k}$, multicasting time t^{cast}, and node aggregation time t_m, is formulated as follows

$$\begin{aligned} \textbf{P1:} \max_{\boldsymbol{a},\boldsymbol{r},\boldsymbol{s},\boldsymbol{f},\boldsymbol{t},t^{\text{cast}}} & \sum_{m \in \mathcal{R}} w_m a_m, \\ s.t. \quad & (5) - (21), \\ & s_{m,n}^{b,k} \in \{0,1\}, f_{m,n}^{b,k} \geq 0, \quad \forall m \in \mathcal{R}, \\ & n \in \Gamma_m, b \in \mathcal{B}_m \cap \mathcal{B}_n, k \in \{1, ..., K\}, \quad (22) \\ & r_{m,n} \in \{0,1\}, \ \forall m \in \mathcal{R}, n \in \Gamma_m, \quad t_m \in [0, D], \ \forall m \in \mathcal{R}, \\ & t^{\text{cast}} \in [0, D], \quad (23) \end{aligned}$$

where $w_m = \frac{\sum_{c \in \mathcal{S}_m} |\mathcal{D}_c|}{\sum_{c \in \mathcal{S}} |\mathcal{D}_c|}$ is the weighting factor for PoC^2 m, which is proportional to the number of data samples associated with PoC^2 m. Due to the non-convex constraints (14)-(16) on aggregation time, Problem **P1** is a non-convex mixed-integer nonlinear programming, for which we have the following result.

***Theorem** 1:* Problem **P1** is NP-hard.

Proof: See Appendix A. ∎

Briefly speaking, it can be shown that the maximum independent set problem can be reduced to Problem **P1**. It has been proved that the maximum independent set problem is not only NP-hard but also extremely hard to approximate [33][3]. This fact hinders us from not only finding the exact solution, but also a constant-factor approximation to Problem **P1** in polynomial time. Therefore, we will develop a heuristic solution approach to Problem **P1**.

V. Solution Approach

In this section, we first transform the mixed-integer nonlinear programming into a mixed-integer linear programming (MILP). Then, inspired by [34], [35], we develop a heuristic algorithm, called coarse-grained fixing procedure, to effectively find the solution.

[3]Reference [33] studies the maximum clique problem, which is approximation-equivalent to the maximum independent set problem. These two problems are complementary because a clique in a graph is an independent set of the complement graph.

A. Linearization

The main challenge in solving Problem **P1** stems from the non-convex constraints (14)-(16) with positive part operator "+" and a nonlinear function $\sigma_m(\boldsymbol{a}, \boldsymbol{r})$ (defined in (12)). To linearize these constraints, we first need to linearize $\sigma_m(\boldsymbol{a}, \boldsymbol{r})$, for which we can directly remove the positive part operator "+" as follows

$$\sigma_m(\boldsymbol{a}, \boldsymbol{r}) = \left(\sum_{\{m' | m \in \Gamma_{m'}\}} r_{m',m} + a_m - 1 \right) c_m, \quad \forall m \in \mathcal{R}. \tag{24}$$

Although $\sigma_m(\boldsymbol{a}, \boldsymbol{r})$ can be $-c_m$ based on the linear constraint (24), which is not meaningful in practice, it would not impact the solution. This is because a PoC^2 with $\sigma_m(\boldsymbol{a}, \boldsymbol{r}) = -c_m$ would never receive or forward any model (since $r_{m',m}$ and a_m are equal to 0), and hence have no impact on the system.

In order to linearize (14)-(16), we also need to remove the operator "+" in (14)-(16). We define several auxiliary variables, including continuous variables $v_m^k \in [0, D]$, $h_n^k \in [0, D]$, and $y_{m,n} \in [0, D]$, and binary variables b_m^k, d_n^k, and $u_{m,n}$. Then, (14)-(16) can be equivalently expressed as

$$f_{m,n}^{b,k} \leq e_{m,n} v_m^k, \ f_{m,n}^{b,k} \leq e_{m,n} h_n^k, \ f_{m,n}^{b,k} \leq e_{m,n} y_{m,n}, \quad \forall m \in \mathcal{R}, n \in \Gamma_m, b \in \mathcal{B}_m \cap \mathcal{B}_n, k \in \{1, ..., K\}. \tag{25}$$

with

$$\begin{aligned} & v_m^k \geq k\Delta - (t_m + \sigma_m(\boldsymbol{a}, \boldsymbol{r})), \\ & v_m^k \leq k\Delta - (t_m + \sigma_m(\boldsymbol{a}, \boldsymbol{r})) + D b_m^k, \ v_m^k \leq D(1 - b_m^k), \\ & v_m^k \in [0, D], \ b_m^k \in \{0,1\}, \ \forall m \in \mathcal{R}, k \in \{1, ..., K\}, \quad (26) \\ & h_n^k \geq t_n - \Delta(k-1), \ h_n^k \leq t_n - \Delta(k-1) + D d_n^k, \\ & h_n^k \leq D(1 - d_n^k), \ h_n^k \in [0, D], \\ & d_n^k \in \{0,1\}, \quad \forall n \in \mathcal{R} \cup \{g\}, k \in \{1, ..., K\}, \quad (27) \\ & y_{m,n} \geq t_n - (t_m + \sigma_m(\boldsymbol{a}, \boldsymbol{r})), \\ & y_{m,n} \leq t_n - (t_m + \sigma_m(\boldsymbol{a}, \boldsymbol{r})) + D u_{m,n}, \\ & y_{m,n} \leq D(1 - u_{m,n}), \quad y_{m,n} \in [0, D], \ u_{m,n} \in \{0,1\}, \\ & \forall m \in \mathcal{R}, n \in \Gamma_m. \quad (28) \end{aligned}$$

The linearization is achieved at the cost of adding three binary variables b_m^k, d_n^k, and $u_{m,n}$. Enforced by (26), (27), and (28), the auxiliary variable v_m^k, h_n^k, and $y_{m,n}$ are equal to the term $()^+$ in (14)-(16), respectively. Consequently, Problem **P1** can be reformulated by replacing (12) and (14)-(16) with the linear constraints above, i.e.,

$$\begin{aligned} \textbf{P2:} \max_{\boldsymbol{a},\boldsymbol{r},\boldsymbol{s},\boldsymbol{f},\boldsymbol{t},t^{\text{cast}},\boldsymbol{v},\boldsymbol{h},\boldsymbol{y},\boldsymbol{b},\boldsymbol{d},\boldsymbol{u}} & \sum_{m \in \mathcal{R}} w_m a_m, \\ s.t. \quad & (5) - (11), (13), (17) - (28), \end{aligned}$$

where $\boldsymbol{v}$, $\boldsymbol{h}$, $\boldsymbol{y}$, $\boldsymbol{b}$, $\boldsymbol{d}$, and $\boldsymbol{u}$ denote the collections of the auxiliary variables v_m^k, h_n^k, $y_{m,n}$, b_m^k, d_n^k, and $u_{m,n}$. Since **P2** is a reformulation of **P1**, it is not hard to see that the proof for NP-hardness in Theorem 1 can also be applied to **P2**. Fortunately, **P2** is an MILP, implying that many existing algorithms, such as branch and bound algorithm, or off-the-shelf

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2022.3168538, IEEE Transactions on Wireless Communications

Algorithm 1 Coarse-grained Fixing Procedure

Input: The parameters for Problem **P2** and threshold θ.
Output: The solution $\boldsymbol{x}$ to Problem **P2**.

1: For Problem **P2**, eliminate the nodes in set Γ_m farther from GBS than node m is if Γ_m is not empty after this elimination.
2: Relax $s_{m,n}^{b,k}$ to $[0,1]$ in **P2**.
3: **while** not all $s_{m,n}^{b,k}$ are fixed **do**
4: Solve the relaxed **P2** with fixed $s_{m,n}^{b,k}$ (if any), and obtain the solution for $s_{m,n}^{b,k}$.
5: **if** there exists $s_{m,n}^{b,k} > \theta$ **then**
6: Fix all $s_{m,n}^{b,k} > \theta$ to 1.
7: **else**
8: Fix the largest $s_{m,n}^{b,k}$ to 1.
9: **end if**
10: Based on the fixed $s_{m,n}^{b,k}$, fix some other $s_{m,n}^{b,k}$ to 0 according to (18)-(21).
11: **end while**
12: Solve Problem **P2** with fixed $s_{m,n}^{b,k}$ to obtain the solution $\boldsymbol{x}$.

Algorithm 2 Enhanced Coarse-grained Fixing Procedure

Input: The parameters for Problem **P2**, scale factors $\{\beta_1, \beta_2, ..., \beta_N\}$, and threshold θ.
Output: The solution $\boldsymbol{x}$ to Problem **P2**.

1: For Problem **P2**, eliminate the nodes in set Γ_m farther from GBS than node m is if Γ_m is not empty after this elimination.
2: Initialize $i = 0$.
3: **for** $i < N$ **do**
4: $i = i + 1$.
5: Change link capacity $e_{m,n}$ to $\beta_i e_{m,n}$.
6: Execute line 1 - 11 in Algorithm 1 to fix $s_{m,n}^{b,k}$.
7: Solve Problem **P2** with the original link capacity $e_{m,n}$ and the fixed $s_{m,n}^{b,k}$ to obtain the solution $\boldsymbol{z}_i$ and the objective value V_i.
8: **end for**
9: $\boldsymbol{x} = \boldsymbol{z}_{\arg\max_{1 \le i \le N}\{V_i\}}$.

softwares (e.g., MATLAB) can solve it exactly in practice. However, these approaches may not be applicable to a large-scale problem, because the time complexity exponentially increases with the number of the integer variables, particularly the four-dimension resource allocation variables $s_{m,n}^{b,k}$. In view of this, we further devise a heuristic for more efficient solution finding.

B. Coarse-grained Fixing Procedure

1) Basic Coarse-grained Fixing Procedure: The major obstacle in solving Problem **P2** lies in the integer variables $s_{m,n}^{b,k}$, the number of which increases with the network size and resource dimensions. Our idea is to first develop a heuristic algorithm to fix $s_{m,n}^{b,k}$, and then solve the problem with fixed $s_{m,n}^{b,k}$. Specifically, we will fix $s_{m,n}^{b,k}$ through an iterative process. At each round, we relax *all* the integer variables $s_{m,n}^{b,k}$, a_m, $r_{m,n}$, b_m^k, d_n^k, and $u_{m,n}$ to $[0,1]$, converting Problem **P2** into a linear programming (LP), which can be easily solved by LP solvers. Given the solution $s_{m,n}^{b,k}$ to the relaxed problem, we fix all the $s_{m,n}^{b,k} > \theta$ to 1, where the threshold θ can be chosen from $[0.5, 1]$. The threshold range ensures that (18)-(21) will not be violated upon fixing $s_{m,n}^{b,k}$. If all $s_{m,n}^{b,k}$ are less or equal to θ, we set the largest $s_{m,n}^{b,k}$ to 1. Besides, we fix some other $s_{m,n}^{b,k}$ to 0 based on the conflict relationships characterized by (18)-(21). With fixed $s_{m,n}^{b,k}$, we repeatedly solve the resulting LP with decreasing number of variables until all $s_{m,n}^{b,k}$ are fixed to either 0 or 1. Since at least one variable of $s_{m,n}^{b,k}$ is fixed at each round, the iteration number of solving LP is limited by the number of $s_{m,n}^{b,k}$, implying that the iterative process can converge. However, the actual iteration number is much smaller than this, since multiple variables $s_{m,n}^{b,k}$ can be fixed during one iteration. Finally, with the $s_{m,n}^{b,k}$ in place, we solve the resulting optimization problem to obtain the solution. Note that, although variables $s_{m,n}^{b,k}$ are fixed, at the last step, we still need to solve an MILP with binary variables a_m, $r_{m,n}$, b_m^k, d_n^k, and $u_{m,n}$. However, the resulting MILP can be efficiently solved by off-the-shelf optimization software in practice (as validated in Section VII-B), because the number of integer variables is not large under a practical problem setting. The algorithm is summarized in Algorithm 1.

In addition, to reduce the complexity of the solution approach and forward models to GBS quickly, we eliminate some receivers in transmission set Γ_m of each node m to narrow down the decision space. Specifically, we eliminate the nodes in Γ_m which are farther from GBS than node m is if Γ_m does not become empty after this elimination (corresponding to Line 1 in Algorithm 1). The rationale is that forwarding data far away is typically not the optimal solution given that our objective is to deliver models to GBS.

2) Enhanced Coarse-grained Fixing Procedure: Although the approach above can yield a solution efficiently, it relies on a greedy fixing rule, which is not optimal. The sub-optimality comes from the fact that the set of $s_{m,n}^{b,k}$, which are fixed to 1, may not be optimal. Enlightened by [34], we can obtain different sets of fixed $s_{m,n}^{b,k}$, and choose the best among them so as to enhance the solution. Specifically, we change the link capacity $e_{m,n}$ of each channel by a scale factor $\beta \in (0,1]$ and fix $s_{m,n}^{b,k}$ accordingly. When $e_{m,n}$ reduces to $\beta e_{m,n}$, the channel gain becomes lower. To support model delivery, the system may need to select more critical resource tuples, thereby producing a different set of fixed $s_{m,n}^{b,k}$. Then, with the fixed $s_{m,n}^{b,k}$ in place, we solve the resulting problem with the *original* link capacity $e_{m,n}$ to obtain the solution. This process can be done multiple times by using different scale factors. Assuming that we perform N trials with scale factors $\{\beta_1, \beta_2, ..., \beta_N\}$, the solution is set to the best among these N trials. Essentially, this process potentially improves the solution at the cost of running the coarse-grained fixing procedure multiple times. We call this procedure as "enhanced coarse-grained fixing procedure", and outline it in Algorithm 2.

VI. Extension: Traditional Multi-hop Relaying

For the benchmarking purpose, in this section, we adapt our problem formulation to consider the traditional multi-hop relaying scheme without in-network aggregation. In such a case, we assume that each PoC^2 still aggregates its local models from set $\mathcal{S}_m$ of clients, while directly relaying other PoC^2s' models to the next hop upon receiving. Let $\overline{f}_{m,n}^{b,k}$ denote the flow amount delivered via resource tuple $\big((m,n),b,k\big)$ accounting for the local aggregated model at PoC^2 m, which constitutes part of $f_{m,n}^{b,k}$. We have the flow conservation constraints

$$\sum_{n\in\Gamma_m}\sum_{b\in\mathcal{B}_m\cap\mathcal{B}_n}\sum_{k=1}^{k_0}(f_{m,n}^{b,k}-\overline{f}_{m,n}^{b,k})\leq \sum_{\{m'|m\in\Gamma_{m'}\}}\sum_{b\in\mathcal{B}_m\cap\mathcal{B}_{m'}}\sum_{k=1}^{k_0}f_{m',m}^{b,k},\ \forall m\in\mathcal{R},\ k_0\in\{1,...,K-1\},\tag{29}$$

$$\sum_{n\in\Gamma_m}\sum_{b\in\mathcal{B}_m\cap\mathcal{B}_n}\sum_{k=1}^{K}(f_{m,n}^{b,k}-\overline{f}_{m,n}^{b,k})= \sum_{\{m'|m\in\Gamma_{m'}\}}\sum_{b\in\mathcal{B}_m\cap\mathcal{B}_{m'}}\sum_{k=1}^{K}f_{m',m}^{b,k},\ \forall m\in\mathcal{R}.\tag{30}$$

Constraint (29) means that during the first k_0 time slots, for PoC^2 m, the outgoing data flow amount should be no more than the incoming data amount (from other PoC^2) plus its local aggregated model. PoC^2 m can temporarily store some incoming flow and transmit in the upcoming control intervals, as indicated in (29). Constraint (30) guarantees the flow balance over the whole training round that spans K intervals. For simplicity, we do not consider the buffer size limitation. Furthermore, the flow amount variables $f_{m,n}^{b,k}$ and $\overline{f}_{m,n}^{b,k}$ should satisfy

$$\sum_{n\in\Gamma_m}\sum_{b\in\mathcal{B}_m\cap\mathcal{B}_n}\sum_{k=1}^{K}\overline{f}_{m,n}^{b,k}=a_m,\ \forall m\in\mathcal{R},\tag{31}$$

$$f_{m,n}^{b,k}\leq r_{m,n}e_{m,n}\Delta,\tag{32}$$

$$\overline{f}_{m,n}^{b,k}\leq f_{m,n}^{b,k},\ \overline{f}_{m,n}^{b,k}\leq e_{m,n}\Big(k\Delta-\big(\tilde{t}_m a_m+t^{\text{cast}}\big)\Big)^+,\ \forall m\in\mathcal{R},n\in\Gamma_m,b\in\mathcal{B}_m\cap\mathcal{B}_n,k\in\{1,...,K\},\tag{33}$$

where (31) ensures that the local aggregated model is transmitted out if this PoC^2 is selected, i.e. $a_m=1$. (32) indicates that the data traffic can be transmitted over link (m,n) only if routing decision $r_{m,n}=1$ is made. In (33), $\overline{f}_{m,n}^{b,k}$ is upper bounded by $f_{m,n}^{b,k}$ because it is a portion of $f_{m,n}^{b,k}$. (33) also ensures that resource tuple $((m,n),b,k)$ can carry $\overline{f}_{m,n}^{b,k}$ only after PoC^2 m aggregates the models from set $\mathcal{S}_m$ of clients, which is analogous to (14). In Problem **P2**, we replace (8) and (25)-(28) with (29)-(33) for $m\in\mathcal{R}$, yielding a new problem. While (33) is non-convex, we can linearize it in the same way as we linearize (14), which hence is omitted. Finally, we can employ the enhanced coarse-grained fixing procedure (in Section V-B) to solve the MILP.

TABLE II: Key experimental parameters

Parameters	Values
Number of PoC^2s $\lvert\mathcal{R}\rvert$	16
Number of channels $\lvert\mathcal{B}_m\rvert$	4 or 6
PoC^2's transmit power p (dBm)	28
GBS's downlink transmit power P (dBm)	46
Transmission range (meters) R	350
Interference range (meters) R'	500
Channel bandwidth W (MHz)	10
Downlink multicast bandwidth (MHz)	50
Client's one-epoch update time (seconds)	[1, 2]
Epoch number	5
Delay for PoC^2 m to aggregate two models c_m (seconds)	[0.05, 0.2]
The deadline for each training round D (seconds)	60
The length of control interval Δ (seconds)	15
Number of control intervals K	4
Path loss exponent α	4
Noise power spectral density N_0 (W/Hz)	10^{-16}
Antenna-related parameter η	2.5

VII. Performance Evaluation

In this section, we evaluate the effectiveness of our solution approach and the learning performance of the proposed aggregation scheme.

A. Simulation Setup

We consider a $1000\times 1000m^2$ wireless mesh network, consisting of one GBS at the center and 16 PoC^2s deployed nearby. We assume each PoC^2 provides access services to 10 clients, among which only 3 clients are active at each training round due to sporadic traffic. The transmission and interference ranges for PoC^2s are 350m and 500m, respectively. We set deadline $D=1$ minute for a training round, which is composed of $K=4$ control intervals, each lasting 15 seconds.

We adopt two widely used datasets, CIFAR-10 and Fashion MNIST, to evaluate the learning performance. CIFAR contains color images of 10 classes of objects, such as dog, cat, and automobile, whereas Fashion MNIST contains grayscale images of fashion products, such as T-shirt, bag, and dress. These two datasets are good representations of object recognition tasks for many applications, including autonomous driving, smart cameras, e-commerce, etc. CIFAR-10 has 50000 training images and 10000 test images, and Fashion MNIST has 60000 training images and 10000 test images. The training data samples are evenly distributed to 160 clients. We consider both IID settings and non-IID settings. In the IID setting, the data samples are shuffled and evenly partitioned into the clients. In the non-IID setting, each client's data is randomly drawn from 2 classes of samples.

For both datasets, we implement a convolutional neural network (CNN), which consists of convolutional layers with kernel size 3×3 and channel number $(32,32,64,64,128,128)$. Each convolutional layer is activated by ReLU, and every two

of them is followed by 2×2 max pooling. These convolutional layers are followed by three fully connected layers (382 and 192 units with ReLU activation and the last one with 10 units with softmax activation) [16]. The model contains 1.14 millions parameters, and is 4.375 Megabytes in 32-bit float. Each client trains with mini-batch size 32, learning rate 0.01, and momentum 0.5. Notice that higher test accuracy can often be achieved by larger models. Nevertheless, since our goal in this section is to demonstrate the effectiveness of our in-network aggregation method rather than achieving the best accuracy, such a small-sized model is sufficient for our need. Besides, we set epoch number to 5 , i.e., each client performing 5 local updates before uploading. We assume that each client's one-epoch update time is uniformly distributed within [1, 2] seconds, and each client's upload data rate is uniformly distributed within [10, 20] Mbps. The PoC^2 update time $\tilde{t}_m$ is set to the maximum value among the summations of local update and upload times for the active clients with PoC^2 m, plus the model aggregation time at PoC^2 m. The aggregation delay at PoC^2s for one-time model summation, i.e., c_m, is drawn from [0.05, 0.1] seconds. Since a macro base station is typically endowed with significantly powerful computing resource, the delay c_g and $\bar{c}$ in (17) at GBS are ignored. Given that our laptop takes about 1 second to perform one-epoch training on a client's dataset, and 0.08 seconds to sum up two models, the above parameters are realistic. The following simulation results are averaged from 5 trials with different network topologies. For readers' convenience, we summarize the important experimental parameters in Table II.

We will compare the following strategies.

- **In-Network Aggregation Scheme:** each PoC^2 aggregates all the received models before sending to the next hop, corresponding to the solution to Problem **P1**.
- **Traditional Multi-hop Relaying Scheme:** each PoC^2 only aggregates the models from local clients. For the models from other PoC^2s, it directly relays them to the next hop without aggregation. This scheme corresponds to the solution to the extended problem in Section VI.
- **Direct2GBS Scheme:** After aggregating the models from local clients, each PoC^2 directly uploads its aggregated model to GBS via one-hop communications. This is a special case of Problem **P1** where the transmission set of each PoC^2 only contains GBS. To fully reap the benefits of single-hop communications, we do not consider the limits of transmission range and interference range as in the multi-hop modeling, implying that all PoC^2s are allowed to directly communicate with GBS (even though the received SNR can be very low due to long distance).

B. Algorithm Evaluation

Table III compares the proposed coarse-grained fixing procedure with the optimal benchmark. To optimally solve Problem **P2** with a tolerable time, we consider a small-scale problem with 10 PoC^2s and only one channel of 50MHz shared by PoC^2s. For the enhanced coarse-grained fixing procedure, we set the scale factor $\{\beta_1 = 1, \beta_2 = 0.8, \beta_3 = 0.5\}$ and threshold $\theta = 1$. Since the weighting factors w_m are the same in our parameter settings, the objective of **P2** reduces to maximizing the number of aggregated models. As shown in Table III, the enhanced coarse-grained fixing procedure achieves a reasonable solution to Problem **P2** compared with the optimal benchmark (with about 17.6% performance degradation in average). Besides, due to the multiple trials, the enhanced procedure outperforms the coarse-grained fixing procedure. In fact, the enhanced procedure can be further improved by adding more trials with different scaling factor β_i at the cost of increasing computation overhead.

TABLE III: The number of aggregated models achieved by different strategies versus the number of control intervals K.

Strategy \ K	3	4	5	6	7
Optimal	9.0	15.0	21.0	27.0	30.0
Enhanced coarse-grained fixing	8.4	12.0	16.8	21.6	25.2
Basic coarse-grained fixing	4.8	8.4	13.8	16.8	21.0

TABLE IV: The running time (in seconds) for different strategies versus the number of PoC^2s $|\mathcal{R}|$.

Strategy \ $\|\mathcal{R}\|$	4	8	12	16	20
Enhanced coarse-grained fixing	0.58	1.82	5.40	11.35	30.09
Basic coarse-grained fixing	0.19	0.61	1.80	3.78	10.03

Note that we have chosen the number of aggregated models as the performance metric. By now, we just assume that more aggregated models generally lead to better training performance, as pointed out in [1], [16], [29]–[31]. This relationship will be studied and confirmed later.

Table IV examines the running time of our proposed schemes versus the number of PoC^2s, which is conducted via MATLAB on a laptop with 2.60GHz Intel(R) Core(TM) i7-9750H CPU and 16 GB RAM. The running time for both enhanced and basic procedures increases with the the number of PoC^2s. Depending on the network scale, the running time taken by the basic coarse-grained fixing procedures ranges from 0.19 second to 10.03 seconds, and the time taken by the enhanced procedure is two times longer than the basis one because it solves the problem three times with different scale factors β_i. Note that the number of PoC^2s (e.g., small base stations) within one macro cell is typically not large in practice. Besides, for a stationary wireless backhaul network where channel gains vary slowly, GBS can collect the channel state information in advance and solve the optimization problem before one training round starts. These imply that our proposed schemes have reasonable running time for practical implementations.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2022.3168538, IEEE Transactions on Wireless Communications

TABLE V: The training performance for CIFAR-10 and Fashion MNIST with $B = 4$ channels under IID settings.

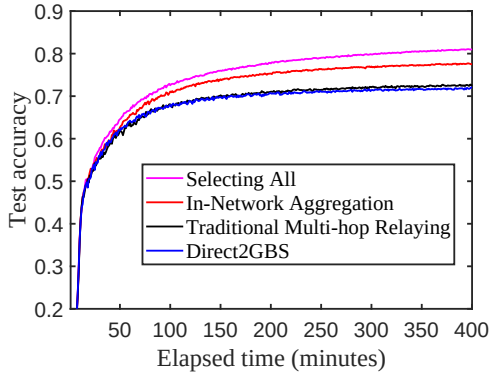
Strategy	The average number of aggregated models per round	Test accuracy at the final deadline		Time (in minutes) taken to achieve a target accuracy	
		CIFAR-10	Fashion MNIST	CIFAR-10 @72%	Fashion MNIST@90%
In-network aggregation	25.2	77.51%	91.66%	114	115
Traditional Multi-hop relaying	15.0	72.42%	90.75%	281	146
Direct2GBS	11.4	72.09%	90.47%	399	162

C. Training Performance

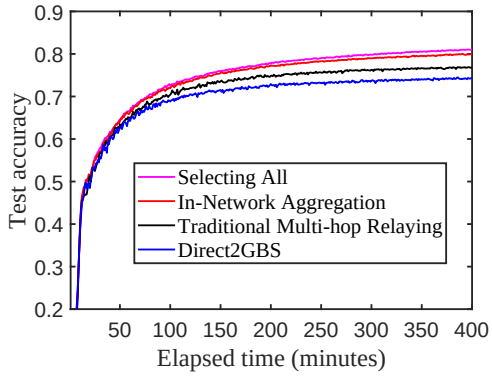
Table V shows the test accuracy for the CIFAR-10 and Fashion MNIST at the final deadline (400 minutes) with $B = 4$ channels under IID settings. As shown in the table, the average number of aggregated models at GBS per round for the proposed in-network aggregation scheme is 25.2, which is 68% more than the traditional multi-hop relaying scheme and 121% more than the Direct2GBS scheme. The Direct2GBS produces the worst performance because the direct model transmissions to GBS suffer from severe path loss resulting from long communication distance. Benefiting from both multi-hop relaying and in-network aggregation, the proposed scheme aggregates the largest number of models and therefore achieves the best test accuracy. For instance, the proposed scheme achieves 77.51% at the final deadline under the CIFAR-10 dataset, which is 5.09% more than the multi-hop relaying scheme and 5.42% more than the Direct2GBS scheme.

Table V also provides the time required for achieving a target accuracy, where CIFAR-10@72% and Fashion M-NIST@91% indicate the time required for achieving 72% and 91% test accuracy under CIFAR-10 and Fashion MNIST datasets, respectively. It is shown that the proposed in-network aggregation scheme significantly reduces the time taken to achieve a certain accuracy level. For example, the proposed scheme only takes 114 minutes to achieve 72% test accuracy under the CIFAR-10 dataset, whereas other two benchmarks take 281 minutes and 399 minutes to achieve the same accuracy level, respectively.

Fig. 3-4 show the test accuracy curves under CIFAR-10 with $B = 4$ and $B = 6$, respectively. In the figures, "Selecting All" assumes that all the active clients' models are aggregated at GBS in each round, which can be regarded as the performance "upper bound". For these figures, it is found that the in-network aggregation scheme outperforms other two benchmark strategies, either under IID or non-IID setting. Moreover, it is not surprising to observe that $B = 6$ leads to higher test accuracy than $B = 4$ for the three schemes under both IID and non-IID settings, because more models can be delivered to GBS under more spectrum resource. Furthermore, it is shown that the in-network aggregation scheme brings more significant improvement than other two benchmarks under the case of $B = 4$ than the case of $B = 6$. For example, the gap between the in-network aggregation and the traditional multi-hop relaying scheme at the final deadline is 5.09% with $B = 4$ while being 3.34% with $B = 6$ under IID settings, implying that the proposed scheme is more beneficial under limited communication resource due to the reduction in data traffic.



(a) IID settings with $B = 4$ channels.

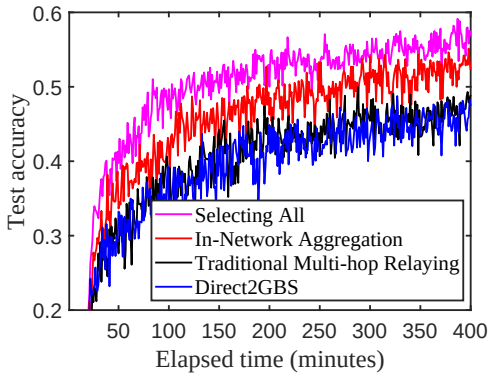


(b) IID settings with $B = 6$ channels.

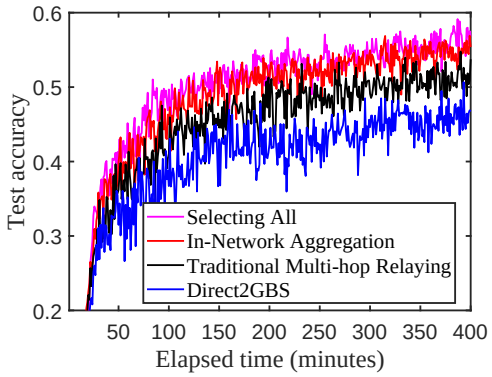
Fig. 3: Test accuracy for the CIFAR-10 dataset under IID settings.

VIII. Conclusion

In this paper, we have proposed an FL framework for multi-hop wireless networks comprising of a mesh of interconnected PoC^2, where each intermediate PoC^2 performs in-network model aggregation before forwarding the data to the next hop. By considering the heterogeneous communication and computing resources in the wireless mesh network, we have designed a joint routing and spectrum allocation scheme to accelerate the training process. While the formulated problem is



(a) Non-IID settings with $B = 4$ channels.



(b) Non-IID settings with $B = 6$ channels.

Fig. 4: Test accuracy for the CIFAR-10 dataset under non-IID settings.

a mixed-integer nonlinear programming, we have transformed it into a mixed-integer linear programming, and developed a coarse-grained fixing procedure to find the suboptimal solution efficiently. We have also generalized our problem formulation to the case where some PoC^2s are not equipped with edge computing capabilities and unable to perform model aggregation. The simulation results have demonstrated the evident advantage of the in-network aggregation scheme over the benchmarks, due to the fact that it can aggregate more models under the limited communication resources.

Appendix A
Proof of Theorem1

Proof: It is known that finding a maximum independent set in a graph is NP-hard. To prove the theorem, our basic idea is to consider a special case of Problem **P1** such that solving Problem **P1** is equivalent to finding a maximum independent set of a graph. Specifically, we construct a two-tier wireless network where a number of tier-1 PoC^2s directly connect with GBS, and a number of tier-2 PoC^2s connect with the tier-1 PoC^2s. There is no link between the same tier of PoC^2s. Every PoC^2 has one model to transmit. We assume the number of time interval $K = 2$, aggregation delay $c_m = 0$, $\tilde{t}_m^{\text{cast}} = 0$, and weighting factor $w_m = 1$ for all PoC^2s. We set PoC^2 update time $\tilde{t}_m = 0$ for the tier-2 PoC^2s, and $\tilde{t}_m = \Delta$ for the tier-1 PoC^2s, implying that tier-1 PoC^2s should wait until the second interval to conduct transmissions. We also set $e_{m,n}^1 = \frac{1}{\Delta}$ for every link (m, n) such that any resource block can deliver one model within an interval. All PoC^2s share a common channel, and each tier-1 PoC^2 has an additional dedicated channel to communicate with GBS. GBS receives data from tier-1 PoC^2s only through these dedicated channels.

We want to make sure that, under the optimal strategy, every tier-1 PoC^2 will transmit during the second interval, such that any model transmitted from a tier-2 PoC^2 to any tier-1 PoC^2 in the first interval will also be successfully transferred to GBS. This is exactly the case in the constructed network, since each tier-1 PoC^2 has a dedicated channel. We define a set of *effective* resource tuples, including the resource tuples from tier-2 to tier-1 PoC^2s in the first interval, and the resource tuples from tier-1 PoC^2s to GBS in the second interval. It is not hard to see that the optimal objective (the maximum number of aggregated models) of Problem **P1** is exactly equal to the number of effective resource tuples that are allocated. We construct a conflict graph G, where the vertices are the effective resource tuples, and the edge between two vertices exists if there is a conflict between them according to (18)-(21). In this way, the optimal objective of Problem **P1** is achieved if and only if a maximum independent set of the conflict graph G is found. Therefore, Problem **P1** is NP-hard. ■

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[2] H. Guo, J. Li, J. Liu, N. Tian, and N. Kato, "A survey on space-air-ground-sea integrated network security in 6G," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 53 – 87, First Quarter 2021.

[3] H. Guo, X. Zhou, J. Liu, and Y. Zhang, "Vehicular intelligence in 6G: Networking, communications, and computing," *Vehicular Communications*, p. 100399, 2021.

[4] H. Ding and K. G. Shin, "Context-aware beam tracking for 5G mmwave V2I communications," *IEEE Trans. Mobile Comput.*, 2021.

[5] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE Commun. Mag*, vol. 57, no. 8, pp. 84–90, 2019.

[6] H. Guo and J. Liu, "UAV-enhanced intelligent offloading for internet of things at the edge," *IEEE Trans. Ind. Informat*, vol. 16, no. 4, pp. 2737–2746, 2019.

[7] J. Liu, H. Guo, J. Xiong, N. Kato, J. Zhang, and Y. Zhang, "Smart and resilient EV charging in SDN-enhanced vehicular edge computing networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 1, pp. 217–228, 2019.

[8] 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 Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, Third Quarter 2020.

[9] Samsung Research. 6G the next hyper-connected experience for all. Accessed on: August 1, 2021. [Online]. Available: https://cdn.codeground.org/nsr/downloads/researchareas/6G%20Vision.pdf

[10] H. Ding, Y. Fang, X. Huang, M. Pan, P. Li, and S. Glisic, "Cognitive capacity harvesting networks: Architectural evolution toward future cognitive radio networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1902–1923, Third Quarter 2017.

[11] Y. Zhu, Y. Niu, J. Li, D. O. Wu, Y. Li, and D. Jin, "QoS-aware scheduling for small cell millimeter wave mesh backhaul," in *2016 IEEE International Conference on Communications (ICC)*. Kuala Lumpur, Malaysia: IEEE, July 2016, pp. 1–6.

[12] M. E. Rasekh, D. Guo, and U. Madhow, "Joint routing and resource allocation for millimeter wave picocellular backhaul," *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 783–794, Feb. 2019.

[13] M. Cao, X. Wang, S.-J. Kim, and M. Madihian, "Multi-hop wireless backhaul networks: A cross-layer design paradigm," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 4, pp. 738–748, May 2007.

[14] H. Ding, C. Zhang, X. Li, J. Liu, M. Pan, Y. Fang, and S. Chen, "Session-based cooperation in cognitive radio networks: A network-level approach," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 685–698, April 2018.

[15] P. Pinyoanuntapong, P. Janakaraj, P. Wang, M. Lee, and C. Chen, "FedAir: Towards multi-hop federated learning over-the-air," in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. Atlanta, GA, USA: IEEE, May 2020, pp. 1–5.

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

[17] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, July 2021.

[18] J. Ren, G. Yu, and G. Ding, "Accelerating dnn training in wireless federated edge learning systems," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 219–232, January 2021.

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

[20] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. Dublin, Ireland: IEEE, June 2020, pp. 1–6.

[21] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "Hfel: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6535–6548, Oct. 2020.

[22] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Barcelona, Spain: IEEE, 2020, pp. 8866–8870.

[23] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Wireless communications for collaborative federated learning," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 48–54, Dec. 2020.

[24] S. Hosseinalipour, S. S. Azam, C. G. Brinton, N. Michelusi, V. Aggarwal, D. J. Love, and H. Dai, "Multi-stage hybrid federated learning over large-scale d2d-enabled fog networks," *IEEE/ACM Trans. Netw.*, early access, 2022.

[25] S. Hariharan and N. B. Shroff, "Maximizing aggregated information in sensor networks under deadline constraints," *IEEE Trans. Autom. Control*, vol. 56, no. 10, pp. 2369–2380, Oct 2011.

[26] B. Alinia, M. H. Hajiesmaili, A. Khonsari, and N. Crespi, "Maximum-quality tree construction for deadline-constrained aggregation in WSNs," *IEEE Sensors J.*, vol. 17, no. 12, pp. 3930–3943, June 2017.

[27] H. Yue, M. Pan, Y. Fang, and S. Glisic, "Spectrum and energy efficient relay station placement in cognitive radio networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 5, pp. 883–893, May 2013.

[28] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," *Wireless networks*, vol. 11, no. 4, pp. 471–487, 2005.

[29] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, Mar. 2020.

[30] S. U. Stich, "Local SGD converges fast and communicates little," in *ICLR 2019-International Conference on Learning Representations*, no. CONF, 2019.

[31] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-IID data," in *International Conference on Learning Representations*, 2019.

[32] J. Luo, C. Rosenberg, and A. Girard, "Engineering wireless mesh networks: Joint scheduling, routing, power control, and rate adaptation," *IEEE/ACM Trans. Netw.*, vol. 18, no. 5, pp. 1387–1400, Oct. 2010.

[33] J. Hastad, "Clique is hard to approximate within $n^{1-\epsilon}$," in *Proceedings of 37th Conference on Foundations of Computer Science*. IEEE, 1996, pp. 627–636.

[34] H. Ding, Y. Guo, X. Li, and Y. Fang, "Beef up the edge: Spectrum-aware placement of edge computing services for the internet of things," *IEEE Trans. Mobile Comput.*, vol. 18, no. 12, pp. 2783–2795, Dec. 2019.

[35] M. Pan, C. Zhang, P. Li, and Y. Fang, "Spectrum harvesting and sharing in multi-hop CRNs under uncertain spectrum supply," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 2, pp. 369–378, Feb. 2012.

Xianhao Chen received the B.Eng. degree from Southwest Jiaotong University, Chengdu, China, in 2017. Since 2018, he has been working towards the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA. His research interests include wireless networking, mobile crowdsourcing, and machine learning.

Guangyu Zhu received the B.Eng. degree from Xidian University, Xi'an, China, in 2019. Since 2019, he has been pursuing the Ph.D degree with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA. His research interests include machine learning, wireless networks, and edge computing.

Yiqin Deng received the B.S. degree in project management from Hunan Institute of Engineering, Xiangtan, China, in 2014, and the M.S. degree in software engineering from Central South University, Changsha, China, in 2017, where she is currently pursuing the Ph.D. degree in computer science and technology. She was a Visiting Researcher with the University of Florida, Gainesville, FL, USA, from 2019 to 2021. Her research interests are primarily focused on edge/fog computing, Internet of Vehicle, and resource management.

Yuguang Fang (Fellow, IEEE) received the M.S. degree from Qufu Normal University, Shandong, China, in 1987, the Ph.D. degree from Case Western Reserve University, Cleveland, OH, USA, in 1994, and the Ph.D. degree from Boston University, Boston, MA, USA, in 1997. He joined the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA, in 2000, as an Assistant Professor, then received early promotion to Associate Professor with tenure in 2003 and a Full Professor in 2005, and has been a Distinguished Professor since 2019.

Prof. Fang received the U.S. NSF Career Award in 2001, the U.S. ONR Young Investigator Award in 2002, the IEEE Vehicular Technology Outstanding Service Award in 2018, the IEEE Communications Society AHSN Technical Achievement Award in 2019, the IEEE Communications Society CISTC Technical Recognition Award in 2015, the IEEE Communications Society WTC Recognition Award in 2014, the Best Paper Award from IEEE ICNP in 2006, and the UF Doctoral Dissertation Advisor/Mentoring Award from 2010 to 2011. He holds the University of Florida Foundation Preeminence Term Professorship from 2019 to 2022, the University of Florida Research Foundation Professorship from 2017 to 2020 and from 2006 to 2009, and the University of Florida Term Professorship from 2017 to 2019 and from 2019 to 2021. He was the Editor-in-Chief of IEEE Transactions on Vehicular Technology from 2013 to 2017 and IEEE Wireless Communications from 2009 to 2012, and serves/served on several editorial boards of premier journals. He also served as the Technical Program Co-Chair for IEEE INFOCOM2014. He is a fellow of AAAS.