FedAir: Towards Multi-hop Federated Learning Over-the-Air

Pinyarash Pinyoanuntapong*, Prabhu Janakaraj*, Pu Wang, Minwoo Lee and Chen Chen†,
Department of Computer Science, † Department of Electrical and Computer Engineering
University of North Carolina Charlotte
Charlotte, USA
{ppinyoan, pjanakar, pu.wang, minwoo.lee, chen.chen}@uncc.edu,

***Abstract*—Federated learning (FL) has emerged as a key technology for enabling next-generation AI at scale. The classical FL systems use single-hop cellular links to deliver the local models from mobile workers to edge routers that then reach the remote cloud servers via high-speed Internet core for global model averaging. Due to the cost-efficiency, wireless multi-hop networks have been widely exploited to build communication backbones. Therefore, enabling FL over wireless multi-hop networks can make it accessible in a low-cost manner to everyone (e.g., under-developed areas and disaster sites). Wireless multi-hop FL, however, suffers from profound communication constraints including noisy and interference-rich wireless links, which results in slow and nomadic FL model updates. To address this, we suggest novel machine learning-enabled wireless multi-hop FL framework, namely FedAir, that can greatly mitigate the adverse impact of wireless communications on FL performance metrics such as model convergence time. This will allow us to fast prototype, deploy, and evaluate FL algorithms over ML-enabled, programmable wireless router (ML-router). The experiments on the deployed testbed validate and show that wireless multi-hop FL framework can greatly accelerate the runtime convergence speed of the de-facto FL algorithm, FedAvg.**

***Index Terms*—Federated Learning, Multi-hop wireless edge, Wireless Networking**

I. Introduction:

A. Why Multi-hop Federated Learning Over-the-Air?

Distributed machine learning, specially federated learning (FL), has been envisioned as a key technology for enabling next-generation AI at-scale. FL significantly reduces privacy risks and communication costs, which are critical in modern AI systems. In FL, the workers, i.e., edge devices, collaboratively learn a shared global model while keeping their data locally to prevent privacy leakage. The workers only need to send their local model updates to the server, which aggregates these updates to continuously improve the shared global model. FL can greatly reduce the required number of communication rounds for model convergence by increasing computation parallelization, where more edge devices are involved as the workers, and by increasing local computation, where the worker performs multiple iterations of model updates before sending the updated model to the server. Through FL, edge devices can still learn much more accurate models with small local datasets. As a result, FL has demonstrated its success for a variety of applications, such as on-device item ranking, content suggestions for on-device keyboards, and next word prediction [1].

Recently, FL systems over edge computing networks have received increasing attention. With single-hop wireless connections, edge devices can quickly reach the FL servers co-located with cellular base stations [2], [3], [4]. Different from cellular systems with high deployment and operational costs, wireless multi-hop networks, consisting of a mesh of interconnected wireless routers, have been widely exploited to build cost-efficient communication backbones, including *wireless community mesh networks* [5] (e.g., NYC mesh [6]), *high-speed urban networks* (e.g., Facebook Terragraph network [7], *global wireless Internet infrastructures* (e.g., SpaceX Starlink satellite constellation [8] and Google Loon balloon network [9]), *battlefield networks* (e.g., rajant kinetic battlefield mesh networks [10]), and *public safety/disaster recuse networks* [11]. Enabling FL over wireless multi-hop networks not only can augment AI experiences for urban mobile users, but also can democratize AI and make it accessible in a low-cost manner to everyone, including the large population of people in low-income communities, under-developed regions and disaster areas.

B. Challenges in Multi-hop Federated Learning:

Despite the impressive features of federated learning and wireless mesh network, there are incumbent challenges that could inherently affect the model accuracy:

- Slow convergence speed of single-layer FL architecture: The FL algorithms generally adopt a single-layer server-client architecture, where a central server collects and aggregates the model updates of all workers. The wireless routing paths towards the central server can be easily saturated in such flat FL architecture.
- Prolonged per-round training time and potential divergence of synchronous federated computing: the de-facto FL algorithm, FedAvg [1] and many its variants operate in a synchronized manner where the server has to wait and collect a minimum number of local model updates before performing model aggregation and moving to the

* These authors contributed equally to this work
This work is supported by NSF 1763182

next round. The long and random multi-hop delay dramatically increases the number of stragglers (slow devices), therefore prolonging the training time per-round.

- Difficulties of model-based optimization for multi-hop FL system: so far, there are limited research efforts on optimizing wireless FL systems. Existing efforts all focus on single-hop FL over cellular edging computing system [2], [12], [4]. With such assumption, the impact of wireless communication control parameters (e.g., transmission power) on the FL related metrics (e.g., model update delay and loss reduction) can be formulated in an explicit closed-form mathematical model, which greatly eases the FL system optimization. Such model-based optimization is not feasible in multi-hop FL, where the FL performance metrics (e.g., FL convergence time) cannot be explicitly formulated as a closed-form function of the networking control parameters, such as transmission power and packet forwarding decision at each router.

C. Our Contributions:

Our objective in this work is to develop a platform for multi-hop wireless FL that can guarantee high accuracy and faster convergence by taming communication latency.

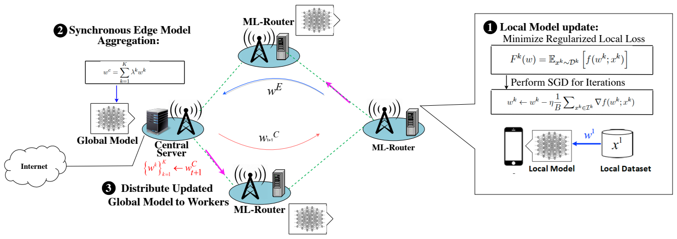
- This is the first work in the literature to reveal, formulate, and experiment on the inherent interplay between multi-hop wireless networking and federated learning.
- We propose and prototype the FedAir, which is the first federated learning system that is optimized by multi-agent reinforcement learning algorithms.
- We demonstrate in the physical testbed that multi-agent reinforcement routing algorithms have the great potential to significantly improve the convergence of federated learning, compared to the widely-adopted standardized IEEE 802.11s [13] protocol.

II. Federated Learning over Multi-hop networks

A. Federated Learning as the Local SGD

Wireless multi-hop FL system consists of central server as aggregator with multi-hop link wireless to edge servers, termed as workers. Fig. 1 shows the architecture of federated learning over multi-hop wireless network. Federated learning methods are designed to handle distributed training of neural networks over multiple devices, where the devices have their local training data and aim to find a common model that yields the minimum training loss. Such a scenario can be modeled as the following distributed parallel non-convex optimization $\min_w F(w) = \sum_{k=1}^{N} \lambda^k F^k(w)$, $F^k(w) = {}_{x^k \sim \mathcal{D}^k}\left[f(w^k; x^k)\right]$ where $F(w)$ is the global loss, $F^k(w)$ is the local loss of device k, N is the number of devices, $\lambda^k = \frac{n^k}{n}$ and $\sum_{k=1}^{N} \lambda^k = 1$, where n^k is the number of training samples on device k and $n = \sum_k n^k$ is the total number of training samples in network. The local loss $F^k(w)$ is a non-convex function over data distribution $\mathcal{D}^k$, which is possibly different for different device k.

To solve above optimization problem, FL methods are following a common stochastic optimization technique, called



❷ Synchronous Edge Model Aggregation:
$w^c = \sum_{k=1}^{K} \lambda^k w^k$
Global Model
Internet
Central Server
$\{w^k\}_{k=1}^{K} \leftarrow w_{t+1}^C$
❸ Distribute Updated Global Model to Workers
ML-Router
ML-Router
ML-Router
w^E
w_{t+1}^C
❶ Local Model update:
Minimize Regularized Local Loss
$F^k(w) = \mathbb{E}_{x^k \sim \mathcal{D}^k}\left[f(w^k; x^k)\right]$
Perform SGD for Iterations
$w^k \leftarrow w^k - \eta \frac{1}{B} \sum_{x^k \in \mathcal{I}^k} \nabla f(w^k; x^k)$
w^1
x^1
Local Model
Local Dataset

Fig. 1. Federated Learning over Multi-hop wireless network

local stochastic gradient descent (SGD), which alternates between local SGD iteration and global model averaging for multiple (server-worker communication) rounds, where the worker is the device that participates in the collaborative model training. During each round, the worker tries to reduce its local loss $F^k(w)$ by performing H mini-batch SGD iterations with each iteration following local SGD update: $w^k \leftarrow w^k - \eta \frac{1}{B} \sum_{x^k \in \mathcal{I}^k} \nabla f(w^k; x^k)$, where $\mathcal{I}^k$ is a subset (mini-batch) of the training samples on worker k and $B = |\mathcal{I}^k|$ is the size of the mini-batch. After finishing H local SGD iterations, the workers send their local models $\{w^k\}_{k \le K}$ to the central server, which averages them and updates the global model accordingly $w^c = \sum_{k=1}^{K} \lambda^k w^k$, where K is the number of devices selected to be the workers. The new global model is sent to the workers and the above procedure is repeated.

B. Runtime Convergence Time of Multi-hop FL

Local SGD methods (e.g., de-factor FL algorithm FedAvg) is generally implemented in a synchronous manner, where the SGD update sequences on the workers are synchronized (by model averaging). In other words, the server needs to wait for the model updates from all workers and then it can perform model aggregation, after which the workers can resume their local SGD updates for the next round. As a result, if the actual training runtime (wallclock time) t is used instead of iteration index T, the convergence of local SGD could be as worst as $\mathcal{O}(\sqrt{\tau_{max}}/\sqrt{Kt})$ (where each worker only performs one local iteration) [14]. τ_{max} is the delay of the slowest worker (straggler) to deliver its local model to the server, which could be very small in high-speed data center networks and wireless single-hop networks (e.g., WiFi or cellular). In wireless multi-hop networks, τ_{max} will become a more dominant factor affecting the true runtime convergence due to the large, random and heterogeneous E2E communication delays experienced by the workers. As a result, the theoretically fast convergence of local SGD can be practically slowed down in wireless multi-hop networks. Moreover, the linear convergence speedup by increasing the number of workers K could be also accompanied with the (linearly) increased delay τ_{max} due to escalated network congestion characterized by Little's theory, which leads to convergence slowdown. Thus, the true runtime convergence of synchronous local SGD is still unknown in wireless networks.

C. Convergence Optimization of Multi-hop FL via MA-MDP

Our overall objective is to minimize the runtime convergence time to achieve a required FL accuracy. Towards this goal, the optimal strategy is to minimize the worker-server delay of the **straggler**, which experiences the maximum delay among all workers. However, in the highly dynamic wireless environments, the role of straggler can be randomly switched among different workers as time proceeds. In this paper, we sought a sub-optimal solution, where we minimize the average end-to-end (E2E) delay between all workers and the server. However, even for such sub-optimal solution, we cannot apply the classic model-based optimization because the server-worker E2E delay cannot be explicitly formulated as a closed-form function of the routing/forwarding decisions. As a result, the model-free optimization strategy based on multi-agent reinforcement learning is much more desirable, where each wireless router exploits its instantaneous local experiences to collaboratively learn the delay-optimal routing paths between the workers and the server.

In particular, this problem can be formulated as the multi-agent Markov decision processes (MA-MDP), which can be solved by multi-agent reinforcement learning algorithms. Given the local observation o_i, which is the source IP and destination IP of the incoming FL packet, each router i selects an action a, i.e., the next-hop router, to forward this packet, according to a local forwarding policy π_i. After this packet is forwarded, the router i receives a reward r_i, which is the negative one-hop delay between router i and the selected next-hop router. The return $G_i = \sum_{k=i}^{T} r_k$ is the total reward from intermediate state s_i to final state s_T, where s_i and s_T are the states when a FL packet arrives at the relay router i and destination router T, respectively. Let s_1 be the initial state when a FL packet enters the network from its source router. The source/destination router is the router that a worker or the server is attached to. The goal is to find the optimal policy π_i for router i so that the expected return $J(\boldsymbol{\pi})$ from the initial state (i.e.,E2E server-worker delay) is optimal, where $J(\boldsymbol{\pi}) = E[G_1|\boldsymbol{\pi}] = E[\sum_{i=1}^{T} r_i|\boldsymbol{\pi}]$ where $\boldsymbol{\pi} = \pi_1, ..., \pi_N$.

III. FedAir Framework Design and Prototyping

In this section, we will introduce our ML-enabled wireless edge device design. Our system design is based on the integrated wireless edge hardware platform that can execute federated learning tasks, in addition to the functioning as a wireless mesh router. Fig. 2 shows our proposed wireless edge router design. The framework is composed of an edge ML engine and programmable wireless router as detail follows.

Edge ML Engine for Federated Learning: The motivation of the system design is to facilitate an edge ML engine for federated wireless edge node that is highly modular and configurable. At first, the service provider should be able to deploy any model on our edge nodes and train seemlessly without requiring additional system tuning. To facilitate our design objective, we decouple our edge ML engine into 4 layers (1) user data (2) local model trainer (3) local model exchange (4) global model exchange. We leverage two popular opensource frameworks Tensorflow[15] and Flask[16] to build our Edge ML system. User data module owns the data for federated model training and pre-processing handlers. Local

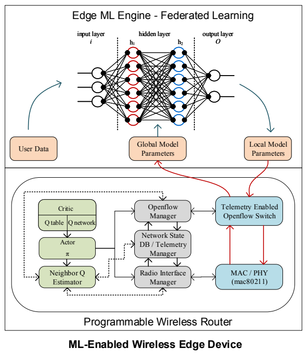


Fig. 2. Architecture of our ML-Enabled Programmable Router (ML-router)

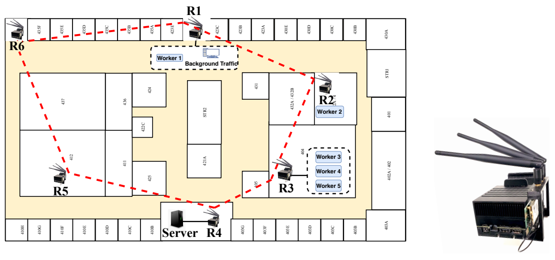
model trainer built on top of tensorflow provides external API-service accessible for deploying new models, initiating training and querying the model status. Local and global model exchange modules are built using light weight Flask web framework to handle the exchange of model updates between the central server and the wireless router.

Programmable Wireless Router: The commercial wireless routers are not suitable for implementing our proposed multi-agent reinforcement routing algorithms due to lack of programmable packet forwarding architecture and telemetry features for wireless multi-hop networking. The three key components of our router includes (1) telemetry-enabled OpenFlow switch with mac80211 interface, (2) programmable flow manager, and (3) reinforcement learning (RL) routing module.

1) Telemetry-enabled OpenFlow Switch with Mac80211: Datapath and wireless radio support are enabled in our wireless router by Linux mac80211 wireless kernel module and Ofsoftswitch13 [17] software switch, designed based on the specifications of SDN [18] OpenFlow protocol version 1.3 [19] for programmable forwarding. Moreover, we modified Ofsoftswitch13 to incorporate in-band network telemetry that allows us to use normal data traffic packets to exchange network measurement data for RL training. In this work, the timestamps that record when each packet left a router are embedded in each packet header to measure the per-hop delay (i.e., reward).

2) Flow Manager: The core function of this module is to enable programmable packet forwarding and radio interface control in wireless multi-hop network. OpenFlow manager receives the human-understandable actions generated by the RL routing module and convert them into the flow rules that can be executed by the OpenFlow switch Ofsoftswitch13. Our radio interface manager module enables dynamic power and channel control.

3) Reinforcement Learning Routing Module: We followed our proposed multi-agent actor-critic (MA-AC) TE framework [20] to solve the above MA-MDP problem, which each router individually runs local actor and critic. **Local Critic for Policy Evaluation:** The task of critic is to criticize how agent behave from the selected policy. This can be evaluated by the action-value $q_i^{\pi}(s, a)$, which is an E2E performance metric. The action-value $q_i^{\pi}(s, a)$ of router i can be written as the sum of 1-hop reward of router i and the action-value of



(a) (b)

Fig. 3. (a) Testbed deployed in a campus building (b) Wireless Router built on the top of Nvidia Jetson Xavier running WINOS

the next-hop router $i+1$. $q_i^{\pi_i}(s,a) = E\left[r_i + q_{i+1}^{\pi_{i+1}}(s',a')\right]$. The estimation of $q_i^{\pi_i}(s,a)$ used exponential weighted average $q_i^{\pi_i}(s,a)$, denoted by $Q_i^{\pi_i}(s,a)$, can be updated based on 1-hop experience tuples (s,a,r_i,s',a') and the estimate of $q_{i+1}^{\pi_{i+1}}(s',a')$ of next-hop router, denoted by $Q_{i+1}^{\pi_{i+1}}(s',a')$. **Local Actor for Policy Improvement:** The task of actor aims to improve the local policy, which can maximize the cumulative sum of reward $J(\boldsymbol{\pi})$. In this work, two near-greedy policies can be used for encouraging exploration. (1) $\epsilon-$greedy policy, where with probability $1-\epsilon$, select the best action and with probability ϵ, other actions are selected, and (2) softmax-greedy policy, where each action a is selected with a probability $P(a)$ according to the exponential Boltzmann distribution, $P(a) = \frac{\exp(Q_i^{\pi_i}(s,a)/\tau)}{\sum_{b\in\mathcal{A}_i}\exp(Q_i^{\pi_i}(s,b)/\tau)}$.

IV. Experimental Evaluation

We investigate and study how RL-based networking can efficiently improve the convergence speed performance of FL algorithms in a physical testbed with our proposed system design. The performance of our solution is compared with IEEE 802.11s, which is the most widely-adopted and standardized wireless multi-hop routing protocol.

Testbed Setup: As shown in Fig. 3, a software-defined wireless mesh network testbed was deployed on the 4th floor of Woodward Hall at UNCC. This testbed consists of six Nvidia Jetson Xavier nodes connected with WLE900VX wireless interface card and WINOS system on top of Ubuntu 18.04 Linux operating system running on each Nvidia Jetson node. Each mesh router was configured to operate in Mesh point (MP) mode, with fixed 5 Ghz channel, 40 Mhz channel width in 802.11ac operating mode, and 30 dBm transmission power. The Nvidia Jetson not only serves as a wireless mesh node to form a wireless multi-hop backbone, but also host local workers to train the federated learning model. We deployed five local workers on three routers (R1, R2, and R3), where each local worker has its own IP address. The server is connected to R4 to run the global model updates as shown in Fig. 3. Worker 1 and a background traffic client were attached to R1. Worker 2 was deployed on R2. We deployed the other workers on R3.

We study the FL convergence time under different network congestion conditions by varying the *background traffic intensity* from none to 1 Mbps and 2 Mbps respectively from a client at R1 and with *different model complexities*. The background traffic is generated by the client at R1 and sent

TABLE I
FL Hyperarameters

Parameter	MNIST CNN	MNIST LSTM
Number of global rounds	20	20
Number of local iterator	1	1
Batch size	32	32
learning rate	0.01	0.01
Model size	5.8 Mbytes	0.8 Mbytes

to server following a fixed routing path of (R2 $\rightarrow$ R3 $\rightarrow$ R4). To emphasize the impact of networking, we use the exactly same parameters shown in Table I for FL across different experiments.

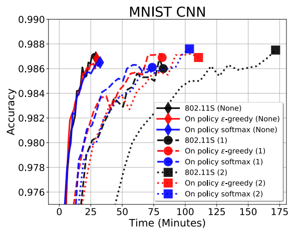
Models and Dataset: To evaluate the performance of Federate learning, we used FedAvg, which trains a deep learning model on the MNIST [21] digit recognition task. The training experiments are performed over two separate models whose weights get updated using federated learning.

- MNIST CNN: we used a CNN with two convolution layers, with 32 and 64 filters respectively. Each convolutional layer was followed by a 2x2 max pooling layer. The convolutions were followed by a fully connected layer with 128 units with Relu activation. A final fully connected layer with softmax activation was used as the final output layer. The model has a size of 5.8 Mbytes.
- MNIST LSTM: we used two stacked LSTM layers each with 128 hidden units. We added a dropout layer after each LSTM layer. The LSTMs were followed by a fully connected layer with 32 units and Relu activation, and a fully connected layer with softmax activation was used as the model output layer. The model has a size of 0.8 Mbytes.

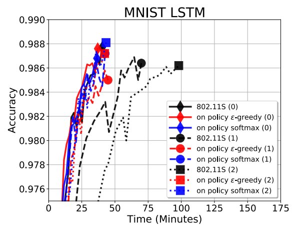
Experiment Results: Fig. 4 and Fig. 5 depict the FL performance in accuracy and convergence time when varying the network traffic load and model complexity. When there is no injected background traffic (solid line with diamond marker), FL traffic can fully utilize all the network resources without any interference from other application traffic. As the straggler (worker 1) can freely send the FL packets to the right path straight away, which is faster than the left one, we can observe that all algorithms achieved a similar performance in terms of accuracy with (98.7%).

In the case of 1 Mbps background traffic load, both on-policy softmax and $\epsilon-$greedy RL-based routing algorithms performed slightly better than the baseline (802.11s) in terms of the total convergence time. Both RL-based routing algorithms converged to select the left path and learned to avoid sending the FL traffic of worker 1 at R1 to the right path, which is congested by a continuous background traffic flow and FL traffic from the workers (2-5). However, the benefit of selecting the left path is not that evident in this case because the end-to-end throughput of the left path was less than the right path and the model size of MNIST CNN (5.8 Mbytes) is relatively large,

The performance gain significantly increased when the background traffic increased to 2 Mbps (dotted-line with

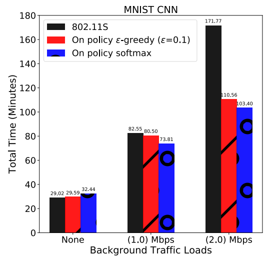


(a) MNIST CNN

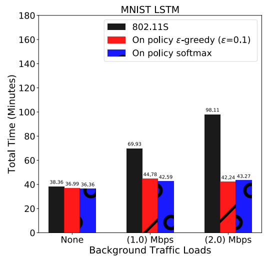


(b) MNIST LSTM

Fig. 4. Comparison results after 20 epochs of accuracy over time of 802.11s routing (black), On-policy $\epsilon-$greedy (red), and On-policy softmax (blue) RL-based routing, respectively, by varying the load of background traffic from None, (1) Mbps, and (2) Mbps (solid, dashed, dotted)-lines.



(a) MNIST CNN



(b) MNIST LSTM

Fig. 5. Total convergence time comparison of 802.11s routing, On-policy $\epsilon-$greedy, and On-policy softmax.

square marker). Since 802.11s is a layer 2 unicast routing protocol, only the destination MAC address is used to route the packet without taking into account the traffic source information. As a result, 802.11s was not able to distinguish the background traffic flow from FL traffic from worker 1. Therefore, it forwarded both background and FL traffic flows to the same link/path, which increased the communication time between straggler (worker 1) and the server. As shown in Fig. 5(a), 802.11s took almost up to 3 hours (171 Minutes) to finish the 20 rounds of training whereas both RL-based routing algorithms took less than 2 hours (110 Minutes) because they learned to optimally distribute different flows among different routing paths.

By varying the model complexity, we investigate how computational and communication overhead affect the FL performance. For both CNN and LSTM, when there is no communication overhead, CNN enjoys the fast local computation. Thus, we observe about 30 minutes of convergence time for CNN and a bit longer 37 minutes for LSTM. However, when the background traffic is injected, the simplicity of the model benefits as it lowers the communication overhead. We can observe that the overall time is greatly reduced with LSTM (even with 802.11s).

Our experiments showed that RL-based networking can efficiently improve the convergence performance of FL algorithms especially when model is complex and the network traffic load is high. Interestingly, the results also show that the choice of model complexity can be another factor to affect the performance of FL.

V. Conclusion

FL over wireless multi-hop networks is challenging due to dynamic network performance resulting in non-optimal routing paths and high communication delay. To maximize the FL accuracy with minimum convergence time, we proposed MARL methods as model-free optimization approaches, where the distributed routers exploit their instantaneous local experiences to collaboratively tune networking parameters on-the-fly. To analyze the convergence of FL system with MARL routing solution, we developed a modular wireless edge system for federated learning with programmable network control. Our experimental results show that the RL-routing algorithms have a great potential to accelerate the convergence of FL in the wireless multi-hop networks, compared with the widely-adopted standardized IEEE 802.11s protocol. To the best of our knowledge, this is the first work to prototype, optimize and demonstrate the wireless multi-hop FL system.

References

[1] H. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2016.

[2] M. Chen, Z. Yang, W. Saad, C. Yin, and S. C. H. Poor, "A joint learning and communications framework for federated learning over wireless networks," 2019, available: https://arxiv.org/abs/1909.07972.

[3] M. Amiri and D. Gunduz, "Over-the-air machine learning at the wireless edge," in *Proc. IEEE SPAWC*, 2019.

[4] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning based on over-the-air computation," in *Proc. IEEE ICC*, 2019.

[5] "wireless community networks," available: http://bit.ly/2Hugn4c.

[6] "New york city (nyc) mesh network," available: https://www.nycmesh.net/.

[7] "Facebook terragraph network," available: https://terragraph.com/.

[8] "Spacex satellite constellation wireless internet," available: http://bit.ly/2uRWDEG.

[9] "Google balloon powered global wireless internet," available: https://loon.com/technology/.

[10] "rajant kinetic mesh networks for battlefield communication," available: https://rajant.com/markets/federal-military-civilian/.

[11] M. Portmann and A. Pirzada, "Wireless mesh networks for public safety and crisis management applications," *IEEE Internet Computing*, vol. 12, no. 1, pp. 18–25, 2008.

[12] M. Amiri and D. Gunduz, "Federated learning over wireless fading channels." Available: https://arxiv.org/abs/1907.09769, 2019.

[13] M. Bahr, "Proposed routing for ieee 802.11 s wlan mesh networks," in *Proceedings of the 2nd annual international workshop on Wireless internet*. ACM, 2006, p. 5.

[14] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proc. AAAI 2019*, 2019.

[15] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[16] *Flask is a lightweight WSGI web application framework.* [Online]. Available: https://www.palletsprojects.com/p/flask/

[17] "Ofsoftswitch13," available: https://github.com/CPqD/ofsoftswitch13.

[18] N. McKeown, "Software-defined networking," *INFOCOM keynote talk*, vol. 17, no. 2, pp. 30–32, 2009.

[19] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[20] P. Pinyoanuntapong, M. Lee, and P. Wang, "Delay-optimal traffic engineering through multi-agent reinforcement learning," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019.

[21] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, Nov 2012.