CF4FL: A Communication Framework for Federated Learning in Transportation Systems

Pedram Kheirkhah Sangdeh*, Chengzhang Li†, Hossein Pirayesh*, Shichen Zhang*, Huacheng Zeng*, and Y. Thomas Hou†
*Department of Computer Science and Engineering, Michigan State University
†Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA

***Abstract*—Federated Learning (FL) is a promising technique to enhance the safety and efficiency of intelligent transportation systems. While FL has been extensively studied, the communication and networking challenges related to the operations of FL in dynamic yet dense vehicular networks remain under-explored. Limited storage and communication capacities of individual vehicles throttle the timely training of an FL model in distributed vehicular networks. In this paper, we present a communication framework for FL (CF4FL) in transportation systems. CF4FL aims to accelerate the convergence of FL training process through the innovation of two complementary networking components: (i) a deadline-driven vehicle scheduler (DDVS), and (ii) a concurrent vehicle polling scheme (CVPS). DDVS identifies a subset of vehicles for local model training in each iteration of FL, with the aim of minimizing data loss while respecting the deadline constraints derived from vehicles' storage, computation, and energy budgets. CVPS takes advantage of multiple antennas on an edge server to enable concurrent local model transmissions in dynamic vehicular networks, thereby reducing the airtime overhead of each FL iteration. We have evaluated CF4FL through a blend of experimentation and simulation. Trace-driven simulation shows that, compared to existing scheduling and transmission schemes, CF4FL reduces the convergence time of FL training by 39%.**

***Index Terms*—Federated learning, machine learning, autonomous driving, intelligent transportation, scheduling, MIMO**

I. Introduction

Knowledge about vehicles, drivers, environments, and their mutual interactions is critical for intelligent transportation systems (ITS). Machine learning (ML) techniques have been extensively studied to extract useful knowledge from massive data collected by vehicles so as to enhance the safety and efficiency of ITS. Conventional ML techniques are propelled by a central server with unconditional access to data collected by vehicles and infrastructure. However, with the advancement of autonomous vehicles, the amount of data from the sensors of vehicles (e.g., LIDARs, radars, cameras, and inertial sensors) can easily reach to gigabit per second, making it impractical to transfer raw data to a server, let alone the privacy issue around sharing raw data.

Federated Learning (FL) has been introduced as a privacy-preserving and communication-efficient alternative, where individual clients (rather than a central server) carry out the model training process [1]. While FL is a promising training paradigm for vehicular networks, the limited communication capacity of these networks along with the heterogeneous sensing, storage, and processing capabilities of individual vehicles, bring up an important question – how to optimize the design and operation of wireless vehicular networks to facilitate FL.

Different strategies have been proposed for FL to address its communication cost, such as decreasing the communication frequency [2], reporting local models using a sparse representation [3]–[6], and quantization of model parameters [7]–[9]. The main idea of these strategies is to reduce the communication overhead of FL by tuning learning parameters and structure, which will likely cause FL performance degradation. Recently, pioneering work [10]–[16] has been conducted to address FL's communication overhead problem from a networking perspective by efficient resource allocation and scheduling schemes. To the best of our knowledge, existing works mainly employ cross-layer optimization techniques to enhance learning efficiency. They assume that global channel state information (CSI) is available at the server. They also assume that CSI remains valid for the time period of an FL iteration (a.k.a. global iteration). Given the small channel coherence time caused by high mobility of vehicles, these two assumptions may not be valid in practical vehicular networks.

In this paper, we present a Communication Framework for FL (CF4FL) for ITS, with the aim of accelerating the FL training process. We consider a vehicular network that comprises a server (for model aggregation and dissemination) and many distributed vehicles (for data collection and local model training). Each vehicle *continuously* collects data samples from its surrounding environment using on-board sensors such as camera, radar, and lidar; and it uses its collected data samples for local model training when scheduled by the server. To embrace topology dynamicity and hardware heterogeneity of vehicular networks, a deadline is defined for each vehicle as the maximum number of global iterations during which the vehicle can keep/store its collected data samples. Once the deadline is reached, the newly collected data samples will be partially or entirely lost due to the limited storage or other limiting factors. CF4FL considers the case where each vehicle has a specific deadline for its data collection. CF4FL mainly comprises two complementary components: (i) deadline-driven vehicle scheduler (DDVS), and (ii) concurrent vehicle polling scheme (CVPS).

DDVS is an online scheduler equipped with two scheduling schemes: *a general but complex scheduler* and *a lightweight heuristic scheduler*. DDVS selects a subset of vehicles in each global iteration. The selected vehicles will perform local model training (using their collected data samples) and send their resultant local models to the server in the current FL iteration, while the vehicles that are not selected will continue to collect data samples. Given the deadline of data collection at vehicles, DDVS must meticulously and systematically select the vehicles in each FL iteration to maximize the amount of data samples for local model training and therefore minimize the data loss at vehicles. CVPS, on the other hand, focuses on enhancing the communication capacity between vehicles and server to reduce the duration of a global iteration. CVPS allows the server to concurrently poll multiple vehicles in a global iteration. The key challenge is the time misalignment of multiple concurrent packets caused by the signal propagation delay, packet processing delay, and clock imperfections. CVPS addresses this challenge by a novel spatial signal detection algorithm, which decodes asynchronous data packets from multiple vehicles. Particularly, CVPS needs neither inter-vehicle synchronization nor instantaneous CSI for asynchronous concurrent vehicle transmissions.

We have evaluated CF4FL through a blend of experimentation and simulation. We implemented CVPS on a software-defined radio (SDR) vehicular testbed where the server has four antennas and each vehicle has one antenna, and evaluated its performance in three typical scenarios: parking lots, local streets, and highways. Our experimental results shows four vehicles can send their local models to the server simultaneously with 98% success rate. The experimental results are utilized to conduct trace-driven simulation for the performance evaluation of CF4FL. Our results show that, DDVS reduces data loss by 76%, 54%, and 59% compared to Random, Round-Robbin, Earliest-Deadline-First schedulers, respectively. Overall, CF4FL reduces the training convergence time of FL by 39%.

II. Related Work

In the literature, there are two research lines involving both FL and networking: *FL for networking* and *networking for FL*. This work belongs to the latter category.

FL in Wireless Vehicular Networks. While many works studied FL applications for transportation systems [17]–[19], few investigated the unique challenges of FL in vehicular networks. [20] considered the heterogeneity of local data samples and designed an approach to selectively collect and aggregate local models for fast convergence. [21] proposed a privacy-preserving aggregation for FL in navigation systems. In [22]–[24], blockchain-based FL frameworks were proposed to protect the privacy of vehicles when sharing local models. [25] proposed a new clustered architecture for FL in vehicular networks which leverages vehicle-to-vehicle communications to conserve communication resources. [26] proposed a greedy algorithm to accelerate FL by assigning resources to the vehicles with high-quality data samples. [10] and [27] are the most relevant works to this paper. [10] proposed an algorithm for vehicle selection and wireless resource allocation in cellular systems based on dataset content. This work took into account both limited bandwidth and packet error rate in its resource allocation strategy to maximize learning efficiency. The proposed resource allocation is reliant on the exact realization of links capacity and the availability of CSI. While CF4FL pursues a similar objective, it differs from [10] in the problem settings, including the lack of instantaneous CSI and concurrent vehicle polling. [27] accelerates FL in vehicular networks by selecting vehicles with massive local datasets and dropping those with few data samples. It neither considers vehicle-specific deadlines nor focuses on minimizing data loss.

Resource Allocation and Scheduling for FL. Resource allocation and participant scheduling in each global iteration are important for FL convergence. Several research efforts have been made to study participant scheduling and resource allocation toward different objectives, such as minimizing FL loss [10]–[12], minimizing latency [13], improving energy efficiency [14]–[16], and enhancing learning efficiency [28]. However, these works are limited to stationary or semi-stationary networks where instantaneous CSI with relatively large coherence time can be estimated at the server in each global iteration. In practice, such an luxury is barely available, especially in vehicular networks. CF4FL differs from these efforts in terms of requirements, objective, and network settings.

Communication Airtime Overhead of FL. The limited communication capacity is a realistic barrier for the FL deployment in wireless networks, which throttles the learning process and slows down the learning convergence. Pioneering works have been done to resolve the communication problem for FL using different approaches, such as decreasing the communication frequency (i.e., the number of global iterations) [2], reporting local models using their sparsified representations [3]–[6], and quantization of model parameters [7]–[9]. Apparently, CF4FL is orthogonal to these efforts as it does not optimize FL but innovates the networking design to improve the convergence speed of FL training.

III. Federated Learning in Vehicular Networks

The deployment of FL in vehicular networks is a complex task due to the unique features of vehicular networks and the stringent requirements of data collection. CF4FL assumes that vehicles can label their collected data for local training. It also assumes that vehicles have sufficient computational power for local training [29]. In what follows, we first describe the system model and then formulate the problem. Finally, we point out the challenges in the design of an efficient solution.

A. System Model

Practical realization of ITS requires to collect an immense amount of data in vehicular environments, such as information of other vehicles, the condition of road surface, the probability of accidents, and the existence of local objects. The collected data by different vehicles is a valuable source of information to train ML models for the applications such as pothole detection, collision avoidance, object/pedestrian identification, and curb avoidance. For vehicles, sharing their raw data samples with a central server for training a unified model may not be a good idea due to the concerns about data privacy, the limited network bandwidth, and the huge size of data samples. FL alleviates these issues and offers a decentralized framework to train a *global model* through the *local model* training at individual vehicles using their privately-owned data.

We consider a vehicular network that comprises a central server and many vehicles as shown in Fig. 1 , where each

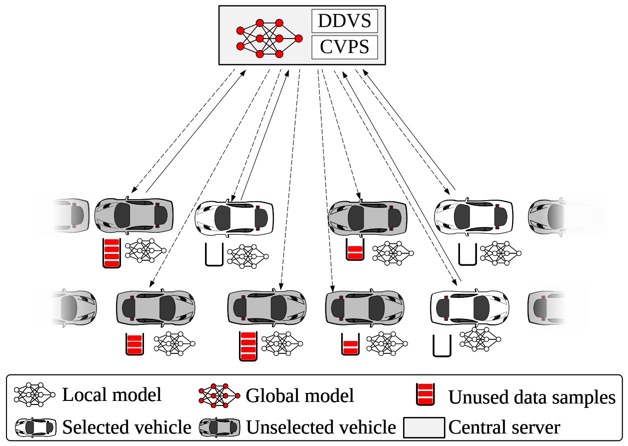


Fig. 1: FL in a vehicular network.

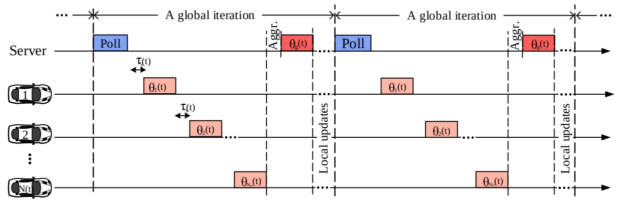


Fig. 2: Sequential polling for the iterative training of FL in vehicular networks.

vehicle is equipped with a single antenna and the server is equipped with multiple antennas. The server is responsible for vehicle scheduling, receiving local models from scheduled vehicles, aggregating models, and broadcasting the aggregated model to all vehicles. Each vehicle performs *continuous* data collection from its surrounding environment using its on-board sensors. If it is scheduled in the current iteration, it first uses its collected data for local model training and then reports the updated local model to the server; otherwise (not scheduled), it continues to collect data until its deadline is reached. To describe FL training, let us consider a network at global iteration t. Denote $\mathcal{N}(t)$ as the set of vehicles associated to a server, with $|\mathcal{N}(t)| = N(t)$. Denote M as the number of antennas at the server. Denote $\mathcal{I}_i(t)$ as the dataset at vehicle i in global iteration t. Assume that the data collection and transmissions at vehicles are done in parallel. Also, assume that a unique frequency band (e.g., a channel in 802.11p or a resource block in C-V2X) is assigned to the FL task under consideration, and the frequency band is used via time division multiple access (TDMA) for communications between vehicles and the server.

B. Problem Formulation

In the conventional training process of FL as shown in Fig. 1 and Fig. 2, the server selects a subset of vehicles for local model training. Denote $\mathcal{S}(t)$ as the set of selected vehicles in global iteration t. Each selected vehicle, say i, trains its local model to minimize a loss function. Denote $\theta_i(t)$ as the local model parameters. Then, the loss function can be written as: $L\left(\theta_i(t), \mathcal{I}_i(t)|\theta_i(t-1)\right)$, where $\theta_i(t-1)$ is the initial parameters of local model. In the rest of this paper, we drop the condition of $\theta_i(t-1)$ for notation simplicity. Vehicle i sends $\theta_i(t)$ to the server and discards $\mathcal{I}_i(t)$ in its buffer to collect future data. An unselected vehicle, on the other hand, piles up the collected data samples during the current iteration on top of what it already has in its buffer, until it has been selected. Piling up data samples can adversely affect the entire training process, especially in a dense vehicular network where some vehicles would not be selected for many global iterations. These vehicles face several issues. *First*, the size of unused data samples (i.e., data samples collected since the last participation in FL) may exceed the storage limit. *Second*, vehicle stragglers (i.e., computation-slow vehicles) drastically increase the time duration of a global iteration. The local processing time can contribute to a time limit (i.e., deadline) for each vehicle or equivalently a virtual cap on an allowable amount of data that the vehicle can collect. We denote this cap as F_i (in bits) for vehicle i. When vehicle i is not selected for a long time and its unused data samples exceed F_i bits, the data samples will be lost from that point on.

At the end of a global iteration, the server receives local models from the selected vehicles and aggregates them to obtain a new global model. The contribution of each local model to the global one is proportional to the amount of data that is used in training that local model [12]. Simply put, the aggregation is a weighted average of the polled local models. Denote $\theta_g(t)$ as the parameters of the aggregated global model. Then, we have

$$\theta_g(t) = \frac{\sum_{i\in\mathcal{S}(t)} |\mathcal{I}_i(t)| \cdot \theta_i(t)}{\sum_{i\in\mathcal{S}(t)} |\mathcal{I}_i(t)|}. \tag{1}$$

Similarly, the global loss is evaluated as a weighted average on loss of local models, i.e.,

$$L(\theta_g(t)) = \frac{\sum_{i\in\mathcal{S}(t)} |\mathcal{I}_i(t)| \cdot L\left(\theta_i(t), \mathcal{I}_i(t)\right)}{\sum_{i\in\mathcal{S}(t)} |\mathcal{I}_i(t)|}. \tag{2}$$

As shown in Fig. 2, the server then broadcasts the global model to all the vehicles, including those who were not selected for local model training in the current global iteration. At all vehicles, their local models are replaced with the global one to initialize the next iteration of local model training. The network continues global iterations until the global model converges, e.g., $|L(\theta_g(t)) - L(\theta_g(t-1))| \le \epsilon$, where ϵ is a predefined threshold. We define the learning efficiency of global iteration t as $|L(\theta_g(t)) - L(\theta_g(t-1))| \cdot \frac{1}{\Delta t}$, where Δt is the time duration of global iteration t.

Now, the question to ask is how to increase learning efficiency in each global iteration while avoiding data loss due to the deadline and storage constraints. To answer this question, we first formulate the learning efficiency in its general form for conventional FL setting. As shown in Fig. 2, Δt is dominated by the uplink and downlink data transmissions as well as local processing time. Then, it can be calculated as:

$$\Delta t = \sum_{i\in\mathcal{S}(t)} \left(\tau_i(t) + \frac{Z(\theta_i(t))}{C_{ui}(t)}\right) + \frac{Z(\theta_g(t))}{\min\limits_{i\in\mathcal{N}(t)} C_{di}(t)}, \tag{3}$$

where $\tau_i(t)$, $C_{ui}(t)$, and $C_{di}(t)$ are vehicle i's local processing time, uplink data rate, and downlink data rate, respectively. $Z(\cdot)$ returns the size of its input in bits. Here, we have $Z(\theta_i(t)) = Z(\theta_g(t)), \forall i \in \mathcal{S}(t)$. The uplink transmissions also take place sequentially in a TDMA manner. In practice, the server typically limits the maximum time duration of a global iteration, i.e., $\Delta t \leq T$, where T is a pre-defined constant.

In each global iteration, the server selects a subset of vehicles that maximize learning efficiency without losing collected data at vehicles. Hence, the problem can be formulated as:

$$\underset{\mathcal{S}(t)}{\text{maximize}} \quad \frac{1}{\Delta t} \cdot \big(L(\theta_g(t)) - L(\theta_g(t-1))\big) \tag{4a}$$

$$\text{s.t.} \quad \Delta t \leq T, \tag{4b}$$

$$|\mathcal{I}_i(t+1)| \leq F_i, \quad \forall i \in \mathcal{N}(t)/\mathcal{S}(t), \tag{4c}$$

where (4b) is the maximum allowable time for a global iteration, and (4c) attempts to avoid data loss for the vehicles that are not selected for local model training in the current global iteration.

C. Challenges

There are two challenges to solve the problem in (4).

Challenge 1. Solving (4) requires perfect global CSI knowledge for the server to select vehicles. However, obtaining fresh CSI for the server is extremely difficult, if not impossible. This is because the channel coherence time in vehicular networks is too short for channel acquisition. For example, consider a relatively small neural network model with $8,778$ parameters, which we later use for digit classification in Section VII. If each parameter is represented by 32 bits, it takes at least 10.1 ms to transmit the entire model in the uplink using the most aggressive modulation and coding scheme (MCS) of IEEE 802.11p, 64QAM and 3/4 coding rate. However, 10.1 ms is very likely beyond the channel coherence time of many vehicular networks. To address this challenge, we propose a deadline-driven vehicle scheduler, which allows the server to poll vehicles in the absence of CSI.

Challenge 2. Another challenge is to reduce the airtime consumption of a global iteration. A natural approach is uplink MU-MIMO transmission, which allows the server to communicate with multiple vehicles at the same time. However, existing uplink MU-MIMO schemes require the packets from vehicles to be aligned in time. This is extremely hard in vehicular networks due to the high mobility (e.g., 60 mph) and the dynamic network topology. Pursuing network-wide timing synchronization, even if possible in practice, inevitably entails a large amount of airtime overhead. To combat this challenge, we propose an asynchronous uplink MU-MIMO scheme, which allows the server to receive packets from multiple vehicles at the same time.

IV. CF4FL: Overview

CF4FL is a heuristic vehicular communication framework to accelerate FL in general. To maximize the learning efficiency in (4), CF4FL focuses on two tasks. *First*, CF4FL endeavors to maximize the numerator of the objective function in (4a) (i.e., learning accuracy). CF4FL pursues the the same objective as the schedulers proposed in [10], [12], [13], [30], in which the analysis shows that learning efficiency will be improved by using (consuming) more data samples for training local models. CF4FL assumes that data at all vehicles are independent and identically distributed (iid) and bear the same quality. It also assumes a direct relation between the data consumption of local training and the convergence of global model. CF4FL considers the vehicle-specific deadlines and avoids data sample loss at vehicles. Leveraging the maximum number of local data samples for training the global model, CF4FL improves the objective function in (4a). *Second*, given the set of selected vehicles (i.e., $\mathcal{S}(t)$), concurrent polling minimizes the denominator of the objective function in (4a) (i.e., Δt). CF4FL strives to solve (4) and meet constraints (4b) and (4c), provided that the original problem has a feasible solution. These two tasks will be carried out by DDVS and CVPS, respectively, as shown in Fig. 3. In what follows, we highlight the key components of CF4FL.

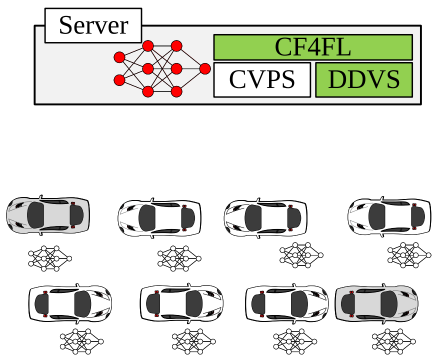
Server. As the central controller, the server is responsible for three tasks: i) passing appropriate information (i.e., deadlines, which are translation of F_i in time domain) to DDVS; ii) aggregating local models; and iii) broadcasting the global model at the end of each global iteration. The calculation of the deadlines is detailed in Section V-A.

DDVS. At the beginning of each global iteration, DDVS receives the deadlines from server and designs a scheduler to poll at most M vehicles by CVPS, where M is the number of antennas at the server. It is the available spatial degrees of freedom (DoF) for polling. If the scheduling problem is feasible, it guarantees that a vehicle is polled before reaching its deadline. Unfortunately, the scheduling problem is not always feasible. As such, DDVS first determines the feasibility of the scheduling problem. If infeasible, DDVS removes some vehicles with the shortest deadlines to make the scheduling problem feasible. This process is illustrated in Fig. 3(a). Once the scheduling feasibility (termed schedulability) is secured, DDVS finds a scheduler to poll the remaining vehicles within a finite number of iterations.

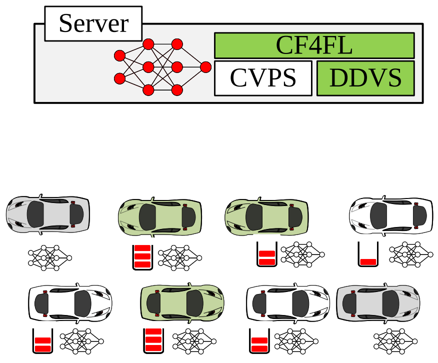
CVPS. Upon receiving a poll frame from DDVS, the selected vehicles prepare their local models and send them to the server. CVPS leverages multiple antennas at the server to decode the uplink data packets. As the vehicles are asynchronous in nature, CVPS first compensates the time and frequency offsets of the collided frames. Then, it constructs a spatial detection filter to recover the data packets, from which local models are extracted by the server.

V. Deadline-Driven Vehicle Scheduler (DDVS)

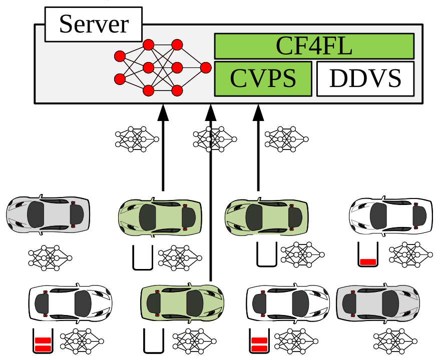
DDVS is responsible for examining the feasibility of (4) and designing a scheduler based on the deadlines specified by the server. We first propose a general scheduler to find a cyclic scheduler that guarantees zero data loss if the *network deadline* (deadlines of all vehicles in the network) is schedulable. The general scheduler comes with a high computational complexity as it needs to find a cycle on a large graph called *steady state graph*, making it hard to implement for large vehicular networks. We therefore propose a lightweight scheduler to handle vehicle selection problem in large vehicular networks.



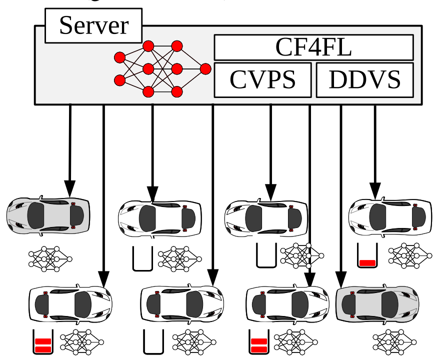
(a) Step 1: If necessary, DDVS removes some of the vehicles (gray colored) and considers a set of vehicles which are schedulable (white colored)



(b) Step 2: Without knowing CSI, DDVS designs a scheduler merely based on vehicles deadlines and buffer status and selects a subset vehicles (green colored)



(c) Step 3: CVPS recovers local models which are concurrently sent by selected vehicles



(d) Step 4: Server aggregates the local models and sends back the global model to all vehicles

Fig. 3: An overview of CF4FL and its underlying components: DDVS schedules the vehicles without requiring global CSI and CVPS recovers concurrent, but asynchronous, packets transmitted by vehicles.

A. Network Deadline and State

DDVS determines vehicles' polling order based on their deadline and state, which we describe below.

Deadline. For a vehicle, say vehicle i, we denote its deadline as d_i. It indeed translates F_i into time domain based on three parameters: the worst-case duration of a global iteration, the sensing rate of vehicle i, and processing delay of vehicle i, which are denoted by t_w, b_i, and τ_i, respectively. t_w is conservatively defined with respect to the case where the lowest MCS in 802.11p is used for all transmissions in a global iteration. b_i and τ_i root in hardware capabilities of vehicle i. We also assume the vehicles persistently collect data in time domain, and the number of their collected data samples linearly increases over time. Then, the deadline for vehicle i is defined as $d_i = \lfloor F_i/(t_w.b_i) + \tau_i/t_w \rfloor$, which is reflected in (4c).

Zero data loss is guaranteed for vehicle i if it collects data no more than d_i subsequent global iterations before being polled. With respect to the individual deadline of vehicles, we further define network deadline as $\vec{d}(t) \triangleq \big(d_1, d_2, \cdots, d_{N(t)}\big)$ for global iteration t. The individual and network deadlines are calculated at the server. Vehicle i, reports F_i and b_i once to the server as a part of its association process. On the other hand, the server is aware of M, $Z(\theta_g)$, and MCS; therefore, it easily obtains t_w which is fixed during the whole training cycle. Then, the server calculates d_i.

State. To indicate the number of global iterations elapsed from the last time a vehicle has been polled, we define a counter, i.e., buffer state. For vehicle i, the buffer state is denoted by $p_i(t)$ and can be written as:

$$p_i\left(t\right) = \begin{cases} 1, & \text{if } i \in \mathcal{S}(t); \\ p_i(t-1)+1, & \text{if } i \notin \mathcal{S}(t). \end{cases} \qquad (5)$$

We further define *network state* as $\vec{p}(t) \triangleq \big(p_1(t), p_2(t), \cdots, p_{N(t)}(t)\big)$ for global iteration t.

B. Schedulability of Network Deadline and General Scheduler

The objective of DDVS is to design a scheduler to honor the constraint of $\vec{p}(t) \preccurlyeq \vec{d}(t)$ for $\forall t > 0$. To design such a scheduler, we need to answer two fundamental questions: i) Is the network deadline schedulable (i.e., the existence of a scheduler that satisfies the constraints in (4))? ii) If a network deadline is schedulable, how to find a scheduler for it? To answer these two questions, we have the following remarks. First, not every network deadline is schedulable. If the network deadline is not scheduable, DDVS first removes some of vehicles to secure the scheduability. Second, for a schedulable network deadline, DDVS designs a cyclic scheduler, which turns out to be optimal but with high computational complexity. Subsequently, a low-complexity heuristic is designed for large-scale vehicular networks.

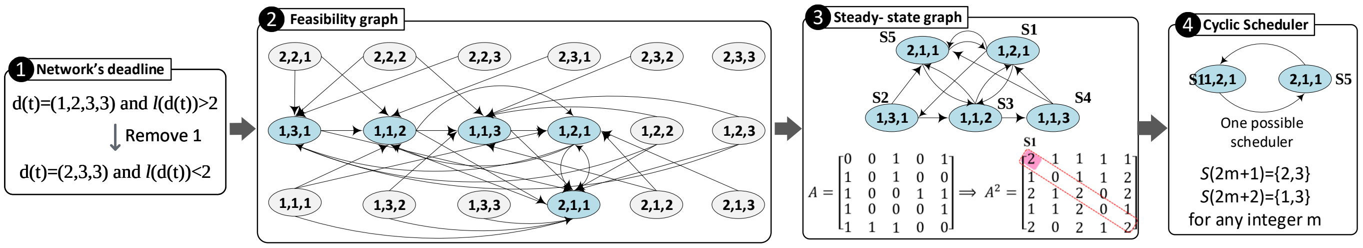


Fig. 4: An example of checking the necessary condition of schedulibility, constructing feasibility and steady-state graphs, finding the shortest cycle, and obtaining a cyclic scheduler.

The two tasks, examining the schedulibility of a network deadline and designing a scheduler, are tightly intertwined. Apparently, the network deadline $\vec{d}(t)$ is schedulable if there exists a scheduler $\mathcal{S}$ (with $\mathcal{S}(t)$ being the set of selected vehicles at global iteration t) such that $\vec{p}(t) \preccurlyeq \vec{d}(t)$ for $\forall t > 0$. Such a scheduler should select vehicle i at least once per d_i global iterations. In addition, as we will show, CVPS will allow the server to poll M vehicles in a global iteration. Define $l(\vec{d}(t)) \triangleq \sum_{i=1}^{N(t)} \frac{1}{d_i}$ as the *network load*. Then, we have the following necessary condition for the schedulibility of $\vec{d}(t)$.

Lemma 1. *If $\vec{d}(t)$ is schedulable, then $l(\vec{d}(t)) \leq M$.*

The proof is given in Appendix A. Lemma 1 implies that the network load supported by the server should not exceed M. DDVS does not set a limit on the number of vehicles for scheduling. Instead, it sets a limit on the network load for schedulability. DDVS can work for a small-size network with as less as M vehicles or a large-scale network with many vehicles. DDVS first determines if the network deadline meets the necessary condition. If not, DDVS removes the vehicles with the smallest deadlines one by one until the condition is met. This treatment follows two reasons. First, a vehicle with the shortest deadline is the bottleneck of scheduling as it has the highest contribution to the network load. Second, a vehicle with the shortest deadline has the lowest contribution in improving global model per poll.

We use a small example shown in Fig. 4 to illustrate this process. In this example, four vehicles with network deadline $\vec{d}(t) = (1, 2, 3, 3)$ are associated to a server with two antennas, i.e., $M = 2$. Referring to step 1 in the example, the initial network load is $l(\vec{d}(t)) = 2.16$. As the network load does not meet the necessary condition, DDVS removes first vehicle with $d_1 = 1$ and updates the network deadline to $\vec{d}(t) = (2, 3, 3)$. Then, we have $l(\vec{d}(t)) < 2$, which meets the necessary condition of schedulability.

Feasibility Graph. Once $\vec{d}(t)$ meets the necessary condition of schedulability, DDVS examines the schedulability of $\vec{d}(t)$. To do so, DDVS constructs a feasible scheduling space including feasible network states and possible transitions between the network states. The feasible scheduling space is constructed using a directed graph called *feasibility graph*. The feasibility graph G is constructed as follows.

$$G = (V, E), \tag{6a}$$

$$V = \{\vec{p}(t) : \ \vec{p}(t) \preccurlyeq \vec{d}(t)\}, \tag{6b}$$

$$E = \{\vec{p}(t-1) \rightarrow \vec{p}(t) : \ \exists\, \mathcal{S}(t) \text{ and } \vec{p}(t) \in V\}. \tag{6c}$$

Referring to Fig. 4, the constructed feasibility graph for a given network deadline $\vec{d}(t) = (2, 3, 3)$ is shown in Step 2. To further clarify state and transitions in this graph, let us focus on an example where the network state is $\vec{p}(t) = (1, 3, 1)$ and $\mathcal{S}(t) = \{1, 2\}$. Since vehicles 1 and 2 are selected, their buffer will be cleared and the network state transits to $\vec{p}(t+1) = (1, 1, 2)$ according to (5). Since $\vec{p}(t) \preccurlyeq \vec{d}(t)$, $\vec{p}(t+1) \preccurlyeq \vec{d}(t+1)$, and $|\mathcal{S}(t)| = M$, both states and corresponding transition belong to the feasibility graph. As an another example, let us consider the case that, for the same initial state, i.e., $\vec{p}(t) = (1, 3, 1)$, first and third vehicles are selected for polling. Then, $\vec{p}(t+1) = (1, 4, 1)$, which is not a feasible state. Therefore, $\vec{p}(t+1)$ and the transition from $\vec{p}(t)$ are not in the feasibility graph.

Per (6c), an edge in the feasibility graph corresponds to a scheduling decision, and a cycle in the graph corresponds to a cycle of decisions that can be followed for infinite time. Therefore, a unique correspondence exists between a cycle on feasibility graph G and a cyclic scheduler $\mathcal{S}$. A cycle with length c in the feasibility graph is equivalent to a *cyclic scheduler* having $\mathcal{S}(t + c) = \mathcal{S}(t)$ for $t > 0$. It is easy to see that, under such a cyclic scheduler, we also have $\vec{p}(t + c) = \vec{p}(t)$ for $t > c$. This correspondence is exploited to determine the schedulibility of a network deadline.

Lemma 2. *$\vec{d}(t)$ is schedulable if and only if there is a cycle on the feasibility graph G in* (6)*, and the repetition of the cycle represents a feasible cyclic scheduler for* (4)*.*

The proof is given in Appendix B. Lemma 2 implies that finding a scheduler for $\vec{d}(t)$ is equivalent to finding a cycle on graph G. The complexity of such a search is $\mathcal{O}(|V| + |E|)$ [31], which is likely to be intractable in practice.[1]

Pruning feasibility graph. To reduce the computational complexity, we narrow down the search space by pruning graph G while maintaining its cycles. This is done by removing all the network states and their connected edges that cannot be a part of any cycle in G. We call the pruned graph *steady state graph* $G_s = (V_s, E_s)$. If $\vec{p}(t) \in V$ is also a vertex in V_s, it has the following features: i) no more than M elements of $\vec{p}(t)$ have hit their deadline; ii) no more than M elements of $\vec{p}(t)$

[1]The number of states in a feasibility graph is $|V| = \prod_{i=1}^{N(t)} d_i$, and the number of outgoing edges from one state can reach up to $C_M^{N(t)}$, which is the number of M-combinations from $N(t)$ vehicles. For a network with tens of vehicles and a few antennas at the server, $|V| + |E|$ can easily reach to millions, making the search for a cycle on G intractable.

that have not hit the deadline carry the same values; and iii) one or more elements of $\vec{p}(t)$ equal to 1. The latter feature can be relaxed to the case where exactly M elements are equal to 1, if the vehicular network is very dense and DDVS intends to use all available spatial degrees of freedom to poll M vehicles in each global iteration. It is obvious that all cycles in G do exist in G_s and vice versa. Obtaining the steady state graph is illustrated in Step 3 of the example shown in Fig. 4, where the feasibility graph is pruned as described above. The steady state graph is much smaller than the feasibility graph. For the example in Fig. 4, it has only 5 states (marked as S1 to S5).

Shortest Cycle in Steady State Graph G_s. The shortest cycle in G_s is critical as it keeps $p_i(t)$ for all $i \in \mathcal{N}(t)$ at small values. Hence, if a vehicle suddenly leaves the network or stops participating in FL, then a small number of data samples will be lost. To find a cycle with the shortest length, we sort vertices in V_s and derive an adjacency matrix $\mathbf{A}$ for G_s such that $\mathbf{A}(i,j) = 1$ if there exists an edge from vertex i to vertex j, and $\mathbf{A}(i,j) = 0$ otherwise. If $N(t) > M$, which is the case for a typical vehicular network, all the diagonal entries of $\mathbf{A}$ are zero, i.e, $\text{diag}(\mathbf{A}) = \mathbf{0}$ where $\mathbf{0}$ denotes an all-zero vector with length $|V_s|$. For such an adjacency matrix, the shortest cycles has length n if n is the smallest integer number for which $\text{diag}(\mathbf{A}^n) \neq \mathbf{0}$. A state corresponding to the position of a non-zero element on diameter of $\mathbf{A}^n$ is located on the shortest cycle(s). Then, we can leverage Floyd–Warshall algorithm [32] to find the shortest cycle for that state. If $N(t) \leq M$, even with the most pressing deadlines, i.e., $d_i = 1\ \forall i \in \mathcal{N}(t)$, the network load meets the necessary condition of schedulibility. All vehicles will be scheduled to serve in every global iterations. In the steady state graph, this scheduler is a self-loop that starts and ends at the same state in every global iteration.

Referring to the example in Fig. 4, the adjacency matrix is calculated. As the first element on the diameter of $\mathbf{A}^2$ is non-zero, the length of the shortest cycle is 2 and it passes through S1. DDVS finds such a cycle on the steady state graph. It chooses one of the two existing cycles with such conditions. In this example, the cycle between S1 and S5 is selected. The cycle corresponds to a cyclic scheduler which selects vehicles 2 and 3 on odd global iterations and selects vehicle 1 and 3 on even ones.

A General Scheduler. Alg. 1 presents an algorithm to check the schedulability of $\vec{d}(t)$ and construct a cyclic scheduler. It comprises four main steps: preparing network deadline, constructing and pruning feasibility graph, constructing steady state graph, and finding the shortest cycle.

Computational Complexity of General Scheduler. The computational complexity of finding the shortest cycles on the steady state graph $G_s = (V_s, E_s)$ using Floyed-Warshall algorithm is $\mathcal{O}(|V_s|^3)$. The number of vertices in G_s can be approximated by: $|V_s| \approx \sum_{j=1}^{C_M^{N(t)}} \prod_{k \in \mathcal{C}_{Mj}} (d_k) - \sum_{i=M+1}^{N(t)} C_i^{N(t)} - \sum_{i=M+1}^{N(t)} \sum_{j=1}^{C_i^{N(t)}} \min_{k \in \mathcal{C}_{ij}} (d_k - 1)$ where where $\mathcal{C}_{ij}$ is the jth realization of i-combinations from $N(t)$ vehicles. In the worst case, the computational complexity of the general scheduler is $\mathcal{O}(d_{max}^{3(N(t)-M)})$, where $d_{max} = \max_{i \in \mathcal{N}(t)} \{d_i\}$. It can be seen that the computational complexity grows polynomially w.r.t. deadlines and exponentially w.r.t. the number of vehicles. Due to its high complexity, this scheme is intractable in dense vehicular networks.

Algorithm 1 A deadline-driven cyclic scheduler

```
1:  Input. The network deadline $\vec{d}(t)$
2:  if $\vec{d}(t) = \vec{d}(t-1)$ then
3:      Keep current scheduler
4:  else
5:      $\mathcal{Q} = \mathcal{N}(t)$
6:      $\mathcal{S} = \emptyset$
7:      while $\sum_{i \in \mathcal{Q}} 1/d_i > M$ or $\mathcal{S} = \emptyset$ do
8:          $\mathcal{Q} \longleftarrow \mathcal{Q} \backslash \{i\}$ such that $d_i = \min\{\vec{d}(t)\}$
9:          Update $\vec{d}(t)$
10:         if $\sum_{i \in \mathcal{Q}} 1/d_i < M$ then
11:             Construct feasibility graph $G$
12:             Obtain $G_s$ from $G$ with adjacency matrix $\mathbf{A}$
13:             if $\exists n \in \mathbb{N}$ such that $\text{diag}(\mathbf{A}^n) \neq \vec{\mathbf{0}}$ then
14:                 Find smallest $n$
15:                 Find cyclic scheduler $\mathcal{S}$ with length $n$ using
                    Floyd–Warshall algorithm [32]
16: Return $\mathcal{S}$
```

C. A Lightweight Scheduler

While Alg. 1 is capable of constructing a cyclic scheduler for a given network deadline, it is of high computational complexity and thus only suited for small networks. For large-scale networks, we propose a heuristic called Extended Polynomial Scheduler (EPS), which is of a low computational complexity.

Main Idea. EPS was inspired by the transformation of "Fictitious Polynomial Mapping" in [33]. The main idea behind EPS is to map a network deadline $\vec{d}(t)$ to a *fictitious polynomial deadline* (FPD) $\vec{\tilde{d}}(t)$ that satisfies $\vec{\tilde{d}}(t) \preccurlyeq \vec{d}(t)$. Based on FPD, we propose EPS for the polynomial deadline $\vec{\tilde{d}}(t)$. Given $\vec{\tilde{d}}(t) \preccurlyeq \vec{d}(t)$, the proposed scheduler by EPS will also meet the original deadline $\vec{d}(t)$.

Transformation. EPS is designed based on a special structure of FPD. A vector $\vec{\tilde{d}}(t) = (\tilde{d}_1, \tilde{d}_2, \cdots, \tilde{d}_{N(t)})$ is FPD if $\tilde{d}_i = b \cdot 2^{m_i}$ for $\forall i \in \mathcal{N}(t), b \in \mathbb{N}$, and $m_i \in \mathbb{Z}$ [33]. Now, a question is that for a given $\vec{d}(t)$, how can we find an FPD $\vec{\tilde{d}}(t)$ such that $d_i \geq \tilde{d}_i$ for all i's and $l(\vec{\tilde{d}}(t)) \leq M$? To find such an FPD, it is sufficient to check $N(t)$ different realizations of FPD, i.e., $d_i = \tilde{d}_i$ for $i \in \{1, 2, \cdots, N(t)\}$. Specifically, for each $i \in \{1, 2, \cdots, N(t)\}$, we construct $\vec{\tilde{d}}(t) = \big(d_i 2^{\lfloor \log_2(d_1/d_i) \rfloor}, d_i 2^{\lfloor \log_2(d_2/d_i) \rfloor}, \cdots, d_i 2^{\lfloor \log_2(d_{N(t)}/d_i) \rfloor}\big)$. We can find a mapping for $\vec{d}(t)$ if and only if we have $l(\vec{\tilde{d}}(t)) \leq M$ for one of these realizations. If such an FPD is not found from all the realizations, we remove vehicles with the shortest deadlines one by one until an FPD is found. It is worth noting that for a single network deadline, multiple FPDs with suitable load may exist. In such a case, we pick the FPD with the lowest load.

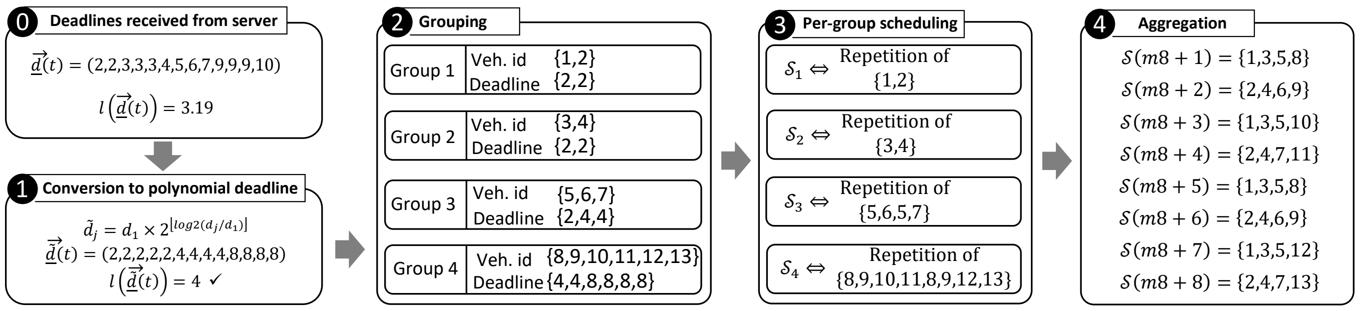

0 Deadlines received from server
$\vec{d}(t) = (2,2,3,3,3,4,5,6,7,9,9,9,10)$
$l(\vec{d}(t)) = 3.19$
1 Conversion to polynomial deadline
$\tilde{d}_j = d_1 \times 2^{\lfloor log2(d_j/d_1) \rfloor}$
$\vec{\tilde{d}}(t) = (2,2,2,2,2,4,4,4,4,8,8,8,8)$
$l(\vec{\tilde{d}}(t)) = 4$ ✓
2 Grouping
Group 1 Veh. id {1,2} Deadline {2,2}
Group 2 Veh. id {3,4} Deadline {2,2}
Group 3 Veh. id {5,6,7} Deadline {2,4,4}
Group 4 Veh. id {8,9,10,11,12,13} Deadline {4,4,8,8,8,8}
3 Per-group scheduling
$\mathcal{S}_1 \Leftrightarrow$ Repetition of {1,2}
$\mathcal{S}_2 \Leftrightarrow$ Repetition of {3,4}
$\mathcal{S}_3 \Leftrightarrow$ Repetition of {5,6,5,7}
$\mathcal{S}_4 \Leftrightarrow$ Repetition of {8,9,10,11,8,9,12,13}
4 Aggregation
$\mathcal{S}(m8+1) = \{1,3,5,8\}$
$\mathcal{S}(m8+2) = \{2,4,6,9\}$
$\mathcal{S}(m8+3) = \{1,3,5,10\}$
$\mathcal{S}(m8+4) = \{2,4,7,11\}$
$\mathcal{S}(m8+5) = \{1,3,5,8\}$
$\mathcal{S}(m8+6) = \{2,4,6,9\}$
$\mathcal{S}(m8+7) = \{1,3,5,12\}$
$\mathcal{S}(m8+8) = \{2,4,7,13\}$

Fig. 5: An example of using EPS for scheduling. The network includes 13 vehicles with network deadline $\vec{d}(t) = (2,2,3,3,3,4,5,6,7,9,9,9,10)$ and $l(\vec{d}(t)) = 3.19$. Final scheduler is the output of step 4 and it is a cyclic scheduler with cycle length 8.

We illustrate the mapping procedure through the example as shown in Fig 5. In this example, the network deadline is $\vec{d}(t) = (2,2,3,3,3,4,5,6,7,9,9,9,10)$, yielding $l(\vec{d}(t)) = 3.19$. We pick $d_1 = 2$ for mapping. Then, we have $\vec{\tilde{d}}(t) = (2 \times 2^{\lfloor \log_2(2/2) \rfloor}, 2 \times 2^{\lfloor \log_2(2/2) \rfloor}, \cdots, 2 \times 2^{\lfloor \log_2(10/2) \rfloor}) = (2,2,2,2,2,4,4,4,4,8,8,8,8)$, yielding $l(\vec{\tilde{d}}(t)) = 4$. Since the network load meets $l(\vec{\tilde{d}}(t)) \leq M$ condition in Lemma 1, we use the polynomial deadline $\vec{\tilde{d}}(t)$ for scheduling.

Grouping. Once an FPD with a proper load is found, we then construct a feasible scheduler. For the special case where $M = 1$, the Fictitious Scheduler Construction (FSC) algorithm in [33] can provide a feasible scheduler when $\vec{d}(t)$ can be mapped to $\vec{\tilde{d}}(t)$ whose load is no more than one, i.e., $l(\vec{\tilde{d}}(t)) \leq 1$. Therefore, for the general case with $M > 1$, if we can divide $\mathcal{N}(t)$ into M separate groups and each group can be mapped to an FPD with load no greater than 1, then a feasible scheduler can be constructed. We now present a procedure to divide $\mathcal{N}(t)$ into such M separate groups. Recall that $\vec{d}(t)$ can be mapped to an FPD $\vec{\tilde{d}}(t)$ with $l(\vec{\tilde{d}}(t)) \leq M$. Without loss of generality, we assume $\tilde{d}_1 \leq \tilde{d}_2 \leq \cdots \leq \tilde{d}_{N(t)}$. Then, we pick the first k elements with $\sum_{i=1}^{k-1} 1/\tilde{d}_i < 1$ and $\sum_{i=1}^{k} 1/\tilde{d}_i \geq 1$ as one group. Again, referring to the example shown in Fig. 5, the FPD is divided into four groups, each of which holds a load no less than 1. The deadlines of the four groups are $\{2,2\}$, $\{2,2\}$, $\{2,4,4\}$, and $\{4,4,8,8,8,8\}$.

Construction of feasible scheduler. To design the feasible scheduler, we apply FSC on each group. The schedulers designed for all groups will be aggregated toward a final scheduler, which makes a decision for polling a subset of vehicles in each global iteration. For the final scheduler, we have the following lemma:

Lemma 3. *For any $\vec{d}(t)$ that can be mapped to an FPD $\vec{\tilde{d}}(t)$ with $\vec{\tilde{d}}(t) \preccurlyeq \vec{d}(t)$ and $l(\vec{\tilde{d}}(t)) \leq M$, EPS can find a feasible scheduler.*

The proof is given in Appendix C. In our example shown in Fig. 5, a cyclic scheduler is designed for each group using FSC in [33]. As an instance, for the first group which includes vehicles 1 and 2, the scheduler polls vehicle 1 on every even global iteration and polls vehicle 2 on every odd global iteration.

Aggregation. Once a cyclic scheduler is constructed for each group. In each global iteration, we poll the vehicles specified by each group. Referring to the example in Fig. 5, at $t = 3$, the vehicle selected in the third global iteration are $\mathcal{S}(3) = \{1,3,5,10\}$. For the general case, if the number of groups are less than M, more than one vehicle specified by a scheduler will be selected. For example, if there are two groups and $M = 4$, on each global iteration, two subsequent vehicles will be selected by the scheduler of each group.

Computational Complexity of EPS. The computational complexity of EPS can be attributed to finding FPD and FSC. FSC is called only once when an appropriate FPD is found. Finding an appropriate FPD may go through an iterative removal of vehicles with the shortest deadline to relax the network load. EPS finds FPD over the entire set of vehicles. Based on the computational complexity analysis in [33], the computational complexity of EPS is $\mathcal{O}(N^3(t)) + \mathcal{O}(Md_{max} \log d_{max})$. In a dense vehicular network, the computational complexity of EPS grows polynomially w.r.t. $N(t)$, which is much lower compared to the general scheduler.

When to Invoke EPS. Alg. 1 presents a generic scheduler, which can find a feasible cyclic scheduler that may not be realized by EPS. This issue can be intuitively inferred by considering the gap between necessary condition of schedulability (i.e., $\vec{d}(t) \leq M$) and a network load threshold that guarantees existence of at least one FPD (i.e., $\vec{d}(t) \leq M\ln(2)$) [33]. However, in moderate-size or large-size vehicular networks, steady state graphs become too large to store and process. Based on the available computational resources at the server, a threshold is needed to be set on the network size to efficiently switch between EPS and the general scheduler, and to gain a suitable trade-off between performance and complexity.

How DDVS (EPS and General Schedulers) Mitigate Stragglers Effect. After incorporating processing delays into the deadlines, if a straggler exists in the network, the network load drastically increases. When such an increase pushes the network load beyond the threshold of schedulability, DDVS removes the vehicles with the shortest deadline. These vehicles are less-capable vehicles, and likely pose high processing delays. Therefore, DDVS treats the schedulability of the network by ignoring stragglers with high processing delays.

Algorithm 2 Extended polynomial scheduler (EPS).

```
1: Input: The network deadline $\vec{d}(t)$
2: if $\vec{d}(t) = \vec{d}(t-1)$ then
3:     Keep current scheduler
4: else
5:     $\mathcal{Q} = \mathcal{N}(t)$
6:     Sort $\vec{d}(t)$ in increasing order
7:     while 1 do
8:         for $i = 1, 2, \cdots, |\mathcal{Q}|$ do
9:             Set $\vec{\tilde{d}}(t) = [d_i 2^{\lfloor \log_2(d_1/d_i) \rfloor}, \cdots, d_i 2^{\lfloor \log_2(d_{|\mathcal{Q}|}/d_i) \rfloor}]$
10:            if $l(\vec{\tilde{d}}(t)) \le M$ then
11:                goto line 14
12:        $\mathcal{Q} \longleftarrow \mathcal{Q} \backslash \{1\}$
13:        Update $\vec{d}(t)$
14:    while $|\mathcal{Q}| > 0$ do
15:        if $\sum_{i=1}^{|\mathcal{Q}|} 1/\tilde{d}_i > 1$ then
16:            Find the smallest $k$ such that $\sum_{i=1}^{k} 1/\tilde{d}_i \ge 1$
17:        else
18:            Set $k = |\mathcal{Q}|$
19:        For $[d_1, d_2, \cdots, d_k]$, use FSC to find a feasible
           scheduler and aggregate it into $\mathcal{S}$
20:        $\mathcal{Q} \longleftarrow \mathcal{Q} \backslash \{1, 2, \cdots, k\}$
21:        Update $\vec{d}(t)$ and $\vec{\tilde{d}}(t)$
22:    Return $\mathcal{S}$
```

Summary of EPS. Alg. 2 summarizes EPS. It first sorts the vehicles based on their deadline in a non-decreasing order and then maps it to multiple FPDs. If EPS finds an FPD with load less than M, it uses that FPD for scheduling; otherwise, it removes the vehicle with the shortest deadline and repeats this procedure. Once an FPD with a load less than M is found, it partitions the FPD into multiple groups and constructs a scheduler for each of them. The aggregation of schedulers for different groups leads to a desired scheduler.

VI. Concurrent Vehicle Polling Scheme (CVPS)

Concurrent vehicle polling will significantly improve the FL convergence, and uplink MU-MIMO is an approach to achieving concurrent vehicle polling. While uplink MU-MIMO has been well studied in WiFi and cellular networks, existing techniques are limited to stationary or semi-stationary networks as they assume the perfect time alignment of uplink transmissions. This assumption, however, is not valid in vehicular networks. This is because, while the frequency synchronization can be achieved using GPS or other techniques, the time misalignment of uplink transmissions (caused by signal propagation delay, packet processing delay, clock jitters, etc.) is hard to eliminated in dynamic vehicular networks. To address this issue, we propose an asynchronous uplink MU-MIMO transmission scheme to enable concurrent vehicle polling. It should be noted that the asynchronism CVPS deals with is different from that in asynchronous FL. CVPS deals with the signal-level asynchrony, while FL deals with the message-level asynchrony. In Asynchronous FL, the server receives delayed local models even after the termination of a global iteration. The asynchronism in asynchronous FL is in the order of packets or frames. In contrast, CF4FL deals with the synchronous FL where all local models from the selected vehicles will be received by the server within the corresponding global iteration. The PHY-layer asynchronism in CF4FL is in the order of signal samples.

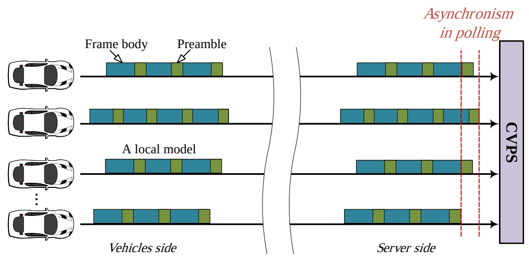


Fig. 6: Asyncronism in local model transmissions of the selected vehicles.

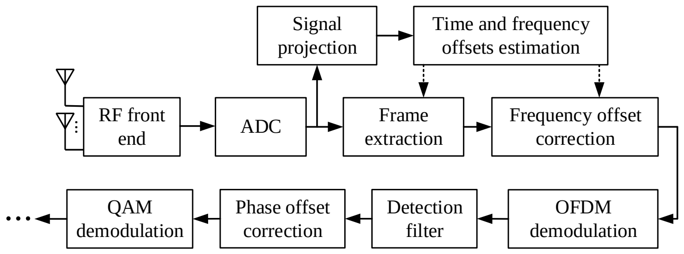


Fig. 7: PHY-layer structure of CVPS.

When DDVS initiates a global iteration, the selected vehicles simultaneously send their local models to the server as shown in Fig. 6. The vehicles transmit their local models through multiple frames within a stream. This is because a frame must lie within channel coherence time, which is relatively short in vehicular networks. Per 802.11p standard, each frame comprises a preamble including a short training field (STF) and a long training field (LTF), a signal field, and payload (frame body).

As shown in Fig. 7, CVPS employs M antennas of the server to mitigate inter-vehicle interference and recovers all the transmitted frames within streams. To do so, the received signal samples from M antennas first go through the signal projection module, which decomposes the signaling space into M subspaces in the time domain. The projection of signal in each subspace is used for time and frequency synchronization. Once time and frequency offsets are compensated, the signals will be converted to frequency domain using OFDM demodulation. A spatial detection filter is then designed for each interfered frame. The spatial detection filter not only suppresses inter-vehicle interference, but it also equalizes the unknown channel. The recovered frame is demodulated after phase offset compensation. In what follows, we describe the key components of CVPS.

Synchronization via Signal Projection. In the vehicular scenario shown in Fig. 6, synchronization of streams is challenging as each stream is polluted by inter-vehicle interference. To alleviate the interference, we project the time-domain signal samples into M orthogonal subspaces. Let us denote the nth received samples from all antennas by $\mathbf{y}(n) \in \mathbb{C}^{M\times 1}$. The basis of signal subspaces at sampling index n, $\mathbf{B}(n)$, can be calculated through eigenvalue decomposition as follows:

$$[\mathbf{B}(n), \mathbf{\Lambda}(n)] = \text{EVD}\left(\frac{1}{2L_s+1}\sum_{i=n-L_s}^{n+L_s}\left(\mathbf{y}(i)\mathbf{y}(i)^{\mathsf{H}}\right)\right), \quad (7)$$

where EVD$(\cdot)$ denotes eigenvalue decomposition, $(\cdot)^{\mathsf{H}}$ is conjugate transpose operation, L_s is an integer number that defines a window length in calculation, $\mathbf{\Lambda}(n) \in \mathbb{C}^{M\times M}$ is a diagonal matrix containing eigenvalues, and $\mathbf{B}(n) \in \mathbb{C}^{M\times M}$ has corresponding eigenvectors with $\mathbf{B}_j(n)$ being its jth columns. $\mathbf{B}_j(n)$ is the base for the jth subspace and can be used to project received signal samples at sampling index n onto subspace j. The projected signals are then used for synchronization. To find the appropriate subspace for a certain stream, we try all subspaces and choose the one with highest cross-correlation peak in time synchronization. That said, if the jth subspace is chosen for stream i, $\tilde{y}_i(n) \triangleq \mathbf{B}_j(n)^{\mathsf{H}}\mathbf{y}(n)$ is employed for time and frequency offset compensations of stream i. The beginning of a frame in stream i is then calculated w.r.t. the peak of correlation between LTF waveform used by vehicle i and $\tilde{y}_i$. Also, the carrier frequency offset is computed by $\theta_i = 1/K \cdot \angle(\sum_{n=n_0}^{n=n_0+K-1}\tilde{y}_i(n)\tilde{y}_i(n+K)^{\mathsf{H}})$, where $\angle(\cdot)$ is the angle of a complex number, K is the FFT size, and n_0 is the position of the first LTF sample in a frame. The calculated offset is corrected before further processing.

Spatial Detection Filter. The synchronized signals are first translated into the frequency domain by OFDM demodulation. Let us focus on the first frame of stream i coming from vehicle i. In the frequency domain, the received signal can be written as:

$$\mathbf{Y}(l,k) = \mathbf{h}_{ui}(k)x_i(l,k) + \sum_{j\in\mathcal{S}(t), j\neq i}(\mathbf{h}_{uj}(k)x_j(l,k)), \quad (8)$$

where $\mathbf{Y}(l,k) \in \mathbb{C}^{M\times 1}$ and $x_i(l,k) \in \mathbb{C}$ are the received signal at the server and the transmitted signal from vehicle i on subcarrier k and sample l, respectively. Also, $\mathbf{h}_{ui}(k) \in \mathbb{C}^{M\times 1}$ denotes the channel from vehicle i to the server on subcarrier k. Although the channel gain may vary over the stream, it is assumed to be unchanged over one frame. For recovering a frame in stream i, we particularly look for filter $\mathbf{P}(k) \in \mathbb{C}^{M\times 1}$ that nullifies $\sum_{j\in\mathcal{S}(t),j\neq i}\mathbf{h}_{uj}(k)x_j(l,k)$ and equalizes the effect of channel $\mathbf{h}_{ui}(k)$. The filter can be constructed as:

$$\mathbf{P}(k) = \left[\sum_{(l,k')\in\mathcal{R}_{ik}}\mathbf{Y}(l,k')\mathbf{Y}(l,k')^{\mathsf{H}}\right]^{-1}\left[\sum_{(l,k')\in\mathcal{R}_{ik}}\mathbf{Y}(l,k')R_i(l,k')^{\mathsf{H}}\right], \quad (9)$$

where $R_i(l,k')$ is the reference signal on sample l and subcarrier k' of the preamble used by vehicle i and $\mathbf{Y}(l,k')$ denotes the corresponding received signal samples over all the antennas. $\mathcal{R}_{ik}$ is the set of reference signal samples located within a pre-defined sliding window around subcarrier k. With this filter, interference mitigation and channel equalization can be achieved through $\hat{x}_i(l,k) = \mathbf{P}(k)^{\mathsf{H}}\mathbf{Y}(l,k)$, where $\hat{x}_i(l,k)$ is the estimated signal symbol. (9) suggests that the design of $\mathbf{P}(k)$ is not reliant on CSI and it only needs pre-known reference signal samples in the preamble of desired frame, which is the case for LTF and STF samples in IEEE 802.11p.

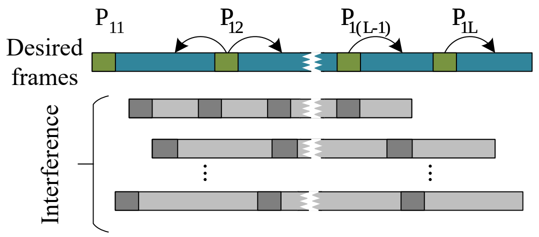


Fig. 8: Illustrating the idea of CVPS.

Computational complexity of Filter Design. The computational complexity of designing a spatial filter is independent of the size of vehicular networks since at most M vehicles will be polled in each global iteration. The design of such a filter requires matrix multiplication, addition, and inversion. The overall computational complexity of designing a spatial filter is $\mathcal{O}(N_{sc}M^3)$, where N_{sc} is the number of subcarriers.

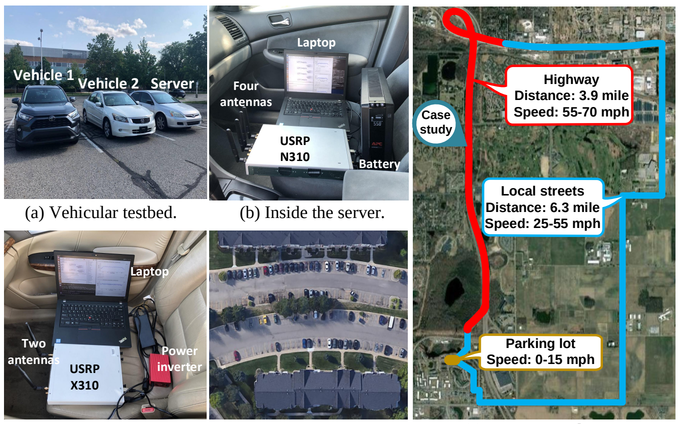
Mitigating Preamble Misalignment. The detection filter in (9) can remove unintended streams if those streams interfere with the preamble of the desired frame. This requirement cannot be met at the first frame of each stream due to the lack of network-wide synchronization. As an example, consider the transmitted streams shown in Fig. 8, where the preamble of the first frame from stream 1 is not collided with stream 2. If the reference signal samples in the preamble are leveraged to design a detection filter like P_{11}, this filter cannot mitigate the interference caused by stream 2. To address this issue, we do not use the preamble of the first frame for filter design. Instead, once the filter P_{12} is designed for the second frame, it is used for both first and second frames. Here, we have assumed that time misalignment does not exceed the length of a frame. It is worth noting that time misalignment is not a challenge for the frames located on the tail of streams. This is because a non-interfering stream at the preamble will not interfere with the rest of the frame. Therefore, starting from the second frame, the detection filter leverages the reference signal samples to recover the frame's body within the same frame as shown in Fig. 8 for frames 2 to L.

VII. Performance Evaluation

In this section, we evaluate the performance of CF4FL and its two components (DDVS and CVPS) using experiments and trace-driven simulation.

A. Evaluation Methodology

We first implement CVPS on a wireless vehicular testbed and investigate its performance on parking lots, local streets, and highways. The measurement results will be used to simulate vehicle polling in large vehicular networks with $N(t) = 5 \sim 25$ vehicles. In our simulation, DDVS uses the general scheduler if $N(t) \leq 8$ and EPS otherwise. Through trace-driven simulation, we then evaluate CF4FL in dynamic vehicular networks of different sizes.


Vehicle 1
Vehicle 2
Server
Laptop
Four antennas
USRP N310
Battery
Laptop
Two antennas
USRP X310
Power inverter
Case study
Highway Distance: 3.9 mile Speed: 55-70 mph
Local streets Distance: 6.3 mile Speed: 25-55 mph
Parking lot Speed: 0-15 mph

(a) Vehicular testbed. (b) Inside the server. (c) Inside a vehicle. (d) Parking lot. (e) Experimental routes

Fig. 9: Experimental scenarios and our vehicular testbed for evaluating polling approaches.

Vehicular Testbed. Fig. 9 shows our small-size vehicular testbed used to evaluate CVPS. It has three vehicles: one acts as the server, and the other two act as four virtual vehicles. The server is implemented using a USRP N310 radio with four antennas ($M = 4$) for the transmission/reception of RF signals, a ThinkPad T480 with Quad-Core i5-8250U CPU for baseband signal processing, and an APC 1500VA UPS battery as shown in Fig. 9(b). Each of the two client vehicles carries a USRP X310 device, a ThinkPad T480 with Quad-Core i5-8250U CPU, and a BESTEK 300W power inverter as shown in Fig. 9(c). Since USRP X310 has two independent RF chains, we use the two client vehicles to emulate four client vehicles, each of which has one antenna for radio signal transmission and reception.

Experimental Route. We evaluate CVPS using sequential polling (i.e., single-user MIMO) as the comparison baseline in three scenarios: a parking lot as shown in Fig. 9(d) at $0 \sim 15$ mph speed, local streets at $25 \sim 55$ mph speed on 6.3 miles, and a highway at $55 \sim 70$ mph speed on 3.9 miles, as shown in Fig. 9(e). The two client vehicles keep staying within $50 \sim 300$ ft distance from the server during several laps on the experimental route.

Trace-Driven Simulation. We simulate CF4FL for large networks with different numbers of vehicles based on our collected experimental results. Specifically, we assume a network with size (number of vehicles) $N(0)$ at the beginning, where a vehicle can join/leave the network based on the arrival/leave global iterations drawn from Poisson distribution with parameter λ. In our simulation, we let $N(0) \in \{10, 15, 20\}$ and $\lambda \in \{0.02, 0.04\}$, with the same probability for a vehicle joining and leaving the network. In a simulated network, each vehicle has an integer deadline drawn from uniform distribution between 2 to 10. Also, we assume a vehicle collects a batch of data during each global iteration.

B. FL Task

As a case study, we use FL to classify images of digits 0 to 9 in our evaluation. The digit classification is a useful tool in vehicular environments for different purposes, such as recognizing road sign of speed limit, identifying the information on traffic sign, recognizing clearance limits, weight limits, etc.

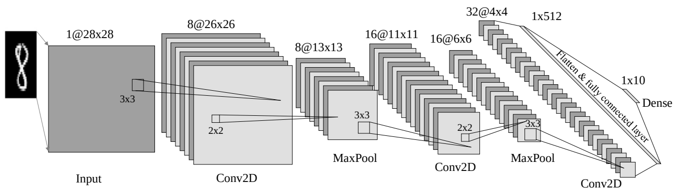


Fig. 10: CNN-based FL application for digit classification.

Dataset. We use MNIST dataset. It includes 70,000 images of handwritten digits, where 60,000 are used for training and 10,000 for test. Each image has 28×28 pixels and labeled with a number from 0 to 9. In our experiments, the dataset is partitioned among vehicles in an iid manner.

Neural Network Architecture. We use a convolutional neural network (CNN) as shown in Fig. 10 to perform the desired FL task. The input is 28×28 pixels. The first 2D convolutional layers are followed by batch normalization, ReLu, and max pooling layers. The outputs of the last 2D convolutional layers are flattened and then flowed into a dense layer. A softmax layer is applied to the output of dense layer to represent the predicted digit.

Training and Convergence. For training digit classifier, the learning rate is set to 0.001 and it is not decayed as the data samples in vehicles will be discarded after consumption (past, current, and future data samples are equally valuable). The number of global iterations is also not pre-set. We assume that the global model converges when the classification accuracy change of two consecutive global iterations is less than 0.1%.

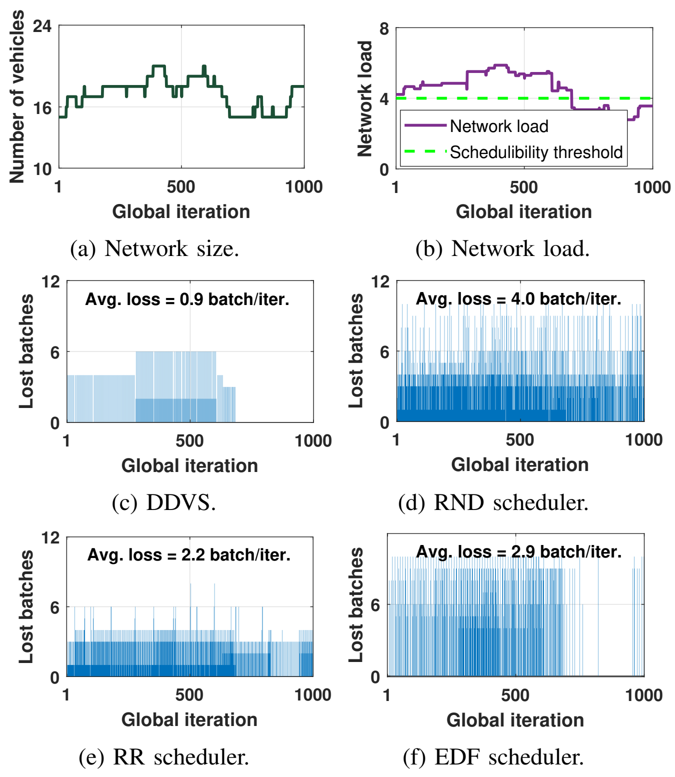
Benchmarks. For DDVS, we employ the following three schedulers as the performance benchmark.

- *Random Scheduler (RND)*: At each global iteration, RND scheduler selects M vehicles among $N(t)$ vehicles with equal probabilities. The selection is performed regardless of vehicles status (i.e., $\vec{d}(t)$ and $\vec{p}(t)$).
- *Round-Robin Scheduler (RR)*: The RR scheduler is essentially a cyclic time-sharing scheduler. It selects M vehicles in each global iteration such that in a large number of global iterations, all the vehicles are polled with equal probability.
- *Earliest Deadline First Scheduler (EDF)*: In each global iteration, EDF scheduler selects M vehicles that are closest to their corresponding deadline (i.e., M vehicles corresponding to M smallest elements in $\vec{d}(t) - \vec{p}(t)$). If more than M vehicles are found with the same closeness, the ones with more data samples are selected.

For CVPS, we employ sequential polling (SP) as the performance baseline. While CVPS polls M vehicles in a global iteration concurrently, SP polls M vehicles sequentially in a global iteration. Finally, for CF4FL, we combine the benchmark schedulers with SP to provide three benchmarks: RND+SP, RR+SP, EDF+SP.

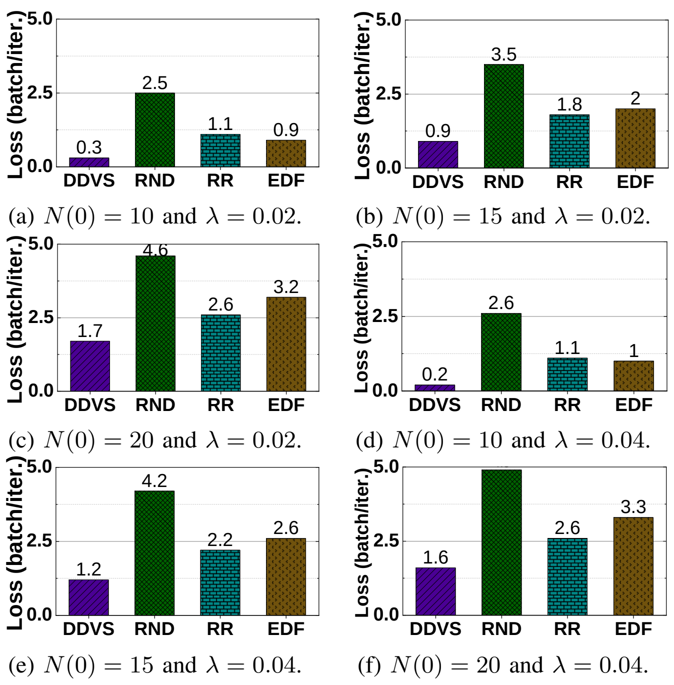
C. Performance of DDVS

A Case Study. We simulate a vehicular network starting with $N(0) = 15$ vehicles at the first global iteration. Vehicles



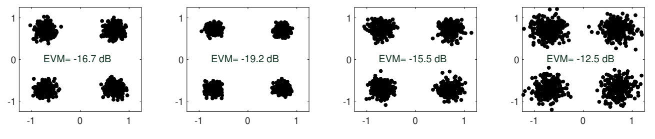
(a) Network size. (b) Network load. (c) DDVS. (d) RND scheduler. (e) RR scheduler. (f) EDF scheduler.

Fig. 11: Data loss of different schedulers for a vehicular network with $N(0) = 15$ and $\lambda = 0.02$.



(a) $N(0) = 10$ and $\lambda = 0.02$. (b) $N(0) = 15$ and $\lambda = 0.02$. (c) $N(0) = 20$ and $\lambda = 0.02$. (d) $N(0) = 10$ and $\lambda = 0.04$. (e) $N(0) = 15$ and $\lambda = 0.04$. (f) $N(0) = 20$ and $\lambda = 0.04$.

Fig. 12: Data loss of DDVS, RND, RR, and EDF schedulers in different vehicular networks.



(a) Stream 1 (b) Stream 2 (c) Stream 3 (d) Stream 4

Fig. 13: The EVM performance of CVPS when decoding four concurrent data packets.

join or leave the network at the beginning of each global iteration, following a Poisson distribution with $\lambda = 0.02$. An instance of such a network is shown in Fig. 11(a). The network size varies between 15 to 20 vehicles during $1,000$ global iterations. The shortest and longest interval between two subsequent changes are 1 and 97 global iterations. The network load also varies between 2.8 to 5.9 as shown in Fig. 11(b). Also, we assume $M = 4$ and, therefore, the necessary condition for the schedulibilty of network load is $l(\vec{d}(t)) \leq 4$. We leverage all benchmark schedulers along with DDVS on this network and measure the lost batches of data in each iteration. The maximum data loss is 6, 10, 8, and 10 batches in an iteration for DDVS, RND, RR, and EDF schedulers, respectively. The average data loss of DDVS, RND, RR, and EDF schedulers is 0.9, 4.0, 2.2, and 2.9 batch per global iteration, respectively.

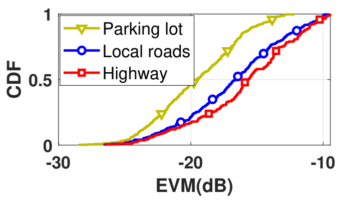
Extensive Simulation. By the same token, we repeat the above study through extensive simulation to measure the gain of DDVS scheduler in different network sizes and different network dynamics. Fig. 12 presents the loss of data, from which we have following observations: i) compared to RND scheduler over all cases, DDVS reduces the data loss by 76.1% on average; ii) compared to RR scheduler, DDVS reduces the data loss by 53.9% on average; and iii) compared to EDF scheduler, DDVS reduces the data loss by 59.0% on average.

D. Performance of CVPS

A Case Study. The case study is conducted at the location marked in Fig. 9(a). We first conduct sequential polling and send a stream from each antenna of vehicles 1 and 2 one by one. The error vector magnitude (EVM)[2] of decoded signals are -19.8 dB, -20.0 dB, -19.1 dB, and -17.3 dB. The average data rate achieved by sequential polling is interpolated to 11 Mbps. We then conduct CVPS, which concurrently polls four streams from vehicles 1 and 2. The constellations of first decoded frame in all streams are shown in Fig. 13. The EVM of decoded frames is -16.7 dB, -19.2 dB, -15.5 dB, and 12.5 dB. Collectively, CVPS yields 34 Mbps data rate. As a global iteration includes the polling of $M = 4$ vehicles in uplink and a broadcast in downlink, CVPS reduces the time consumption of a global iteration by 2.2× compared to sequential polling.

Extensive Experiments. We perform extensive experiments to measure the EVM of decoded signals polled by CVPS and sequential approaches at parking lot, local streets, and highway. The cumulative distribution function (CDF) of measured EVMs is illustrated in Fig. 14. The average EVM of decoded signals with CVPS is -19.5 dB, -16.8 dB, and -15.9 dB at parking lot, local streets, and highway, respectively. The

[2]EVM is calcuated by EVM $= 10\log_{10}(\frac{\mathbb{E}[|\hat{S}(l,k)-S(l,k)|^2]}{\mathbb{E}[|S(l,k)|^2]})$, where $\hat{S}(l,k)$ and $S(l,k)$ are the lth estimated and original modulated symbols on the kth subcarrier, respectively.


(a) CVPS.

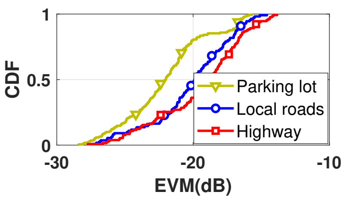
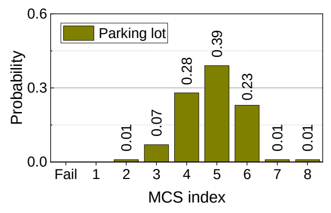
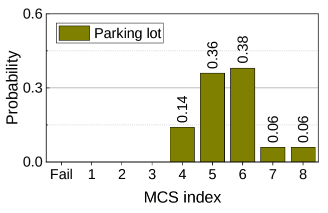
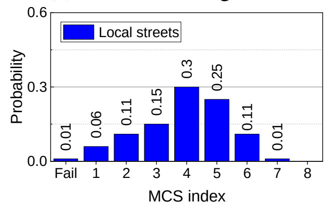
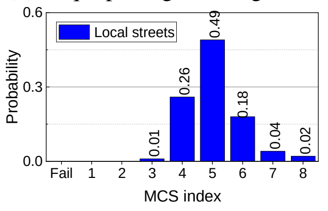

(b) Sequential polling

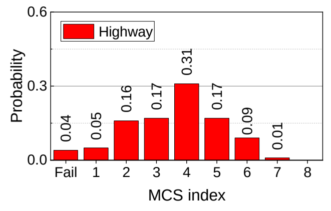
Fig. 14: EVM of decoded frames via CVPS and sequential polling in parking lot, local streets, and highway.


(a) CVPS: Parking lot.


(b) Sequ. polling: Parking lot.


(c) CVPS: Local streets


(d) Sequ. polling: Local streets.


(e) CVPS: Highway.

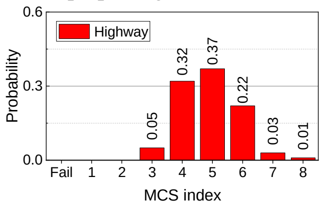

(f) Sequ. polling: Highway.

Fig. 15: Comparison of CVPS and sequential polling in terms of MCS selection probability.

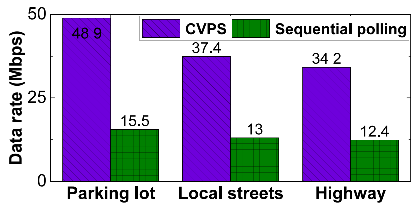
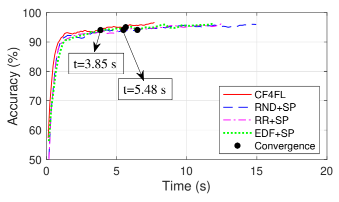
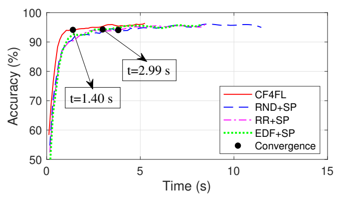
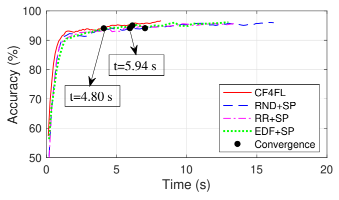
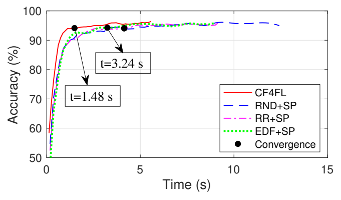


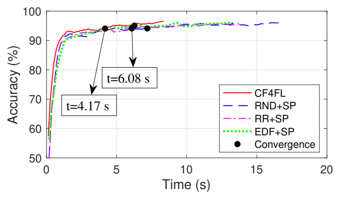
Fig. 16: Data rate achieved by CVPS and sequential polling.


(a) Parking lot: $N(0) = 10$ and $\lambda = 0.02$.


(b) Parking lot: $N(0) = 20$ and $\lambda = 0.04$.


(c) Local streets: $N(0) = 10$ and $\lambda = 0.02$.


(d) Local streets: $N(0) = 20$ and $\lambda = 0.04$.


(e) Highway: $N(0) = 10$ and $\lambda = 0.02$.

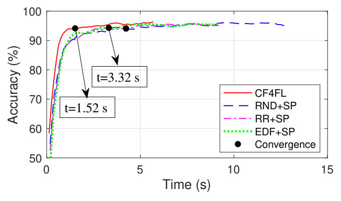

(f) Highway: $N(0) = 20$ and $\lambda = 0.04$.

Fig. 17: Convergence of CF4FL and benchmark approaches in different vehicular scenarios.

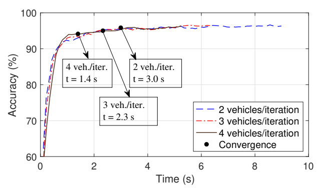
average EVM of decoded signals with sequential polling is -22.0 dB, -20.2 dB, and -19.4 dB at parking lot, local streets, and highway, respectively. Apparently, CVPS has a slight EVM degradation compared to sequential polling. This degradation is caused by the residual inter-vehicle interference of concurrent transmissions.

Fig. 15 presents probability of MCS selection for uplink transmissions in both CVPS and sequential polling. It shows that CVPS causes 1% and 4% data packet loss in local streets and highway, respectively. In contrast, no loss is observed for sequential polling. This is because sequential polling selects a single vehicle for uplink transmissions+, while CVPS selects four vehicles for concurrent uplink transmissions. The data packet loss is caused by the poor uplink channel. Fig. 16 shows the data rate achieved by two polling strategies. It is proportional to the data rate of local model polling. Evidently, CVPS offers a much higher data rate for local model polling in the uplink, thereby shortening the time consumption of each global iteration. Quantitatively, CVPS alone reduces the duration of a global iteration by 58.3%, 52.4%, and 52.4% in a parking lot, local streets, and highways, respectively.

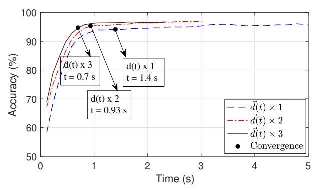
E. Performance of CF4FL (DDVS + CVPS)

Finally, we evaluate the performance of CF4FL by comparing it with RND+SP, RR+SP, and EDF+SP benchmarks in two cases: i) $N(0) = 10$ and $\lambda = 0.02$, and ii) $N(0) = 20$ and $\lambda = 0.04$. Two cases are simulated in the three environments (parking lot, local streets, and highways), and a total of six scenarios are evaluated. The performance of CF4FL and its benchmarks are presented in Fig. 17. And we have the following observations.

- **FL Convergence Speed.** On average over all scenarios, CF4FL reduces the convergence time by 48.2%, 34.9%, and 35.3% compared to RND+SP, RR+SP, and EDF+SP, respectively.
- **Data Collection Speed.** As shown in Fig. 17(a)-(f), CF4FL obtained 60,000 data samples in a shorter period of time compared to the benchmarks. On average, data



(a) Impact of number of selected vehicles.



(b) Impact of scaling deadlines

Fig. 18: Effect of vehicle selection and deadlines on FL training.

collection speed of CF4FL is 2.2×, 1.8×, and 1.7× faster than RND+SP, RR+SP, and EDF+SP, respectively.

F. Effect of Vehicle Selection and Deadlines on Learning

To determine how the number of selected vehicles per global iteration and vehicles' deadlines affect FL training, we have considered two additional test scenarios on a vehicular network at a parking lot with $N(0) = 20$, $\lambda = 0.04$, and a four-antenna server. First, we investigate the effect of selected vehicles by putting an intentional limit on the number of selected vehicles per global iteration. In the second scenario, while four vehicles are selected per iteration, the deadlines are scaled by a factor of $1, 2,$ and 3. As shown in Fig. 18(a), increasing the number of selected vehicles (up to $M = 4$) per global iteration will accelerate the learning process. When the server selects 2, 3, and 4 vehicles per iteration, the global model converges within 3.0s, 2.3s, and 1.4s. Referring to Fig. 18(b), doubling and tripling the deadlines reduce the convergence time by 23.3% and 53.3%, respectively.

VIII. Conclusion

In this paper, we studied vehicle scheduling and polling problems associated with FL in vehicular networks, with the aim of accelerating the convergence speed of FL training process. To tackle the above two problems, we presented CF4FL, a vehicular communication framework for FL training process. CF4FL comprises two complementary components, namely DDVS and CVPS. DDVS is a scheduler for each global iteration of FL, which reduces data loss under deadline constraints. CVPS takes advantage of multiple antennas at the server to enable concurrent local model polling, thereby significantly reducing the time duration of each global iteration and leading to a faster convergence of FL. We have evaluated CF4FL through a blend of experimentation and trace-driven simulation. Our results show that CF4FL reduces the convergence time by 39.4%, and it collects data samples 1.9× faster than existing solutions in parking lots, local streets, and highways.

Acknowledgments

The work of P. Kheirkhah Sangdeh, H. Pirayesh, S. Zhang, and H. Zeng was supported in part by NSF Grant CNS-2100112. The work of C. Li and Y.T. Hou was supported in part by ONR under MURI Grant N00014-19-1-2621, Virginia Commonwealth Cyber Initiative (CCI), and Virginia Tech Institute for Critical Technology and Applied Science (ICTAS).

Appendix A
Proof of Lemma 1

For vehicle i, zero data loss is guaranteed if it is polled at least once in every d_i subsequent global iterations. Let us assume the network's deadline does not change within $t \in [0,\ T]$. Polling rate r_i for vehicle i then satisfies $r_i = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} I_i(\mathcal{S}(t)) \geq \frac{1}{d_i}$, where $I_i(\mathcal{S}(t)) = 1$ if $i \in \mathcal{S}(t)$; otherwise, $I_i(\mathcal{S}(t)) = 0$. For all vehicles, we have $M \overset{(a)}{\geq} \sum_{i=1}^{N(t)} r_i \overset{(b)}{\geq} \sum_{i=1}^{N(t)} \frac{1}{d_i}$, where (a) holds as M is the maximum number of vehicles that can be polled by CVPS in global iteration, and (b) is directly concluded from the constraint on polling rate.

Appendix B
Proof of Lemma 2

If the deadlines and number of vehicles are finite, the number of vertices in G is also finite as:

$$|V| = \prod_{i=1}^{N(t)} d_i < \infty, \ \text{ if } N(t) < \infty \text{ and } d_i < \infty \ \forall i \in \mathcal{N}(t). \tag{10}$$

Therefore, there exists two distinct FL rounds t' and t'' with $1 \leq t' < t'' \leq |V|+1$ for which $\vec{p}(t') = \vec{p}(t'')$. Any path with a length larger than $|V|$ includes a cycle. In other words, we cannot infinitely move on the feasibility graph without forming any cycle. If $\vec{p}(t') = \vec{p}(t'')$ and a cycle is formed, all vehicles are polled at least once within $[t', t'']$. This statement itself can be proved based on contradiction. If vehicle i is not polled, we have:

$$\nexists\, t \in [t', t''] \text{ s.t. } i \in \mathcal{S}(t), \tag{11a}$$
$$p_i(t') < p_i(t''), \tag{11b}$$
$$\vec{p}(t') \neq \vec{p}(t''). \tag{11c}$$

where (11b) follows from the fact that buffer state is a monotonic increasing function as long as the vehicle is not polled. (11b) directly results (11c) which is in contradiction to our assumption $\vec{p}(t') = \vec{p}(t'')$. Therefore, all vehicles will be polled at least once within a cycle. The repetition of the cycle on feasibility graph G results in a cyclic scheduler which results $p_i(t) \leq d_i$ for $t > 0$ and $\forall i \in \mathcal{N}(t)$.

Appendix C
Proof of Lemma 3

$\tilde{d}_i$ can be written as $\tilde{d}_i = b/2^{m_i}$, where $m_i \in \mathbb{N}$, m_i is non-increasing with i, and b is an integer. Equivalently, $1/\tilde{d}_i = 2^{m_i}/b$. For the first group including vehicle 1 to vehicle k, we have:

$$s \triangleq \sum_{i=1}^{k} 2^{m_i} \tag{12a}$$
$$(s - 2^{m_k})/b \leq 1, \tag{12b}$$
$$s/b > 1. \tag{12c}$$

(12b) and (12c) follow from (12a) and the definition of a group. Based on (12b) and (12c), s can be expressed as $s = \lceil b/2^{m_k} \rceil \cdot 2^{m_k} = \lceil \tilde{d}_k \rceil \cdot 2^{m_k}$. Given that $m_i \geq m_k$ for all

$i < k$, we have $s \leq \lceil b/2^{m_i} \rceil \cdot 2^{m_i} = \lceil \tilde{d}_i \rceil \cdot 2^{m_i}$ for $i < k$. Therefore, we can obtain an FPD $\vec{g}$ with $l(\vec{g}) = 1$ as $\vec{g} = (\tilde{d}_1, \tilde{d}_2, \cdots, \tilde{d}_k) \cdot s/b$. For $\tilde{g}_i$, $i = 1, 2, \cdots, k$, we have $\tilde{g}_i = \tilde{d}_i \cdot s/b \leq \tilde{d}_i \cdot \lceil \tilde{d}_i \rceil \cdot 2^{m_i}/b = \lceil \tilde{d}_i \rceil \leq d_i$. Therefore, $(d_1, d_2, \cdots, d_k)$ can be mapped to $\vec{g}$ and we can use FSC to find a scheduler for it [33]. We repeat the above procedure up to $(M-1)$ times for remaining groups, and use FSC to find a feasible scheduler for each of them. Then we combine the M different schedulers and get feasible scheduler $\mathcal{S}$.

References

[1] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[2] 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*, pp. 1273–1282, 2017.

[3] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, 2020.

[4] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 440–445, 2017.

[5] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *International Conference on Learning Representations*, 2018.

[6] H. Sun, S. Li, F. R. Yu, Q. Qi, J. Wang, and J. Liao, "Toward communication-efficient federated learning in the internet of things with edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 11053–11067, 2020.

[7] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," *arXiv preprint arXiv:1705.07878*, 2017.

[8] H. Wang, S. Sievert, Z. Charles, S. Liu, S. Wright, and D. Papailiopoulos, "ATOMO: communication-efficient learning via atomic sparsification," in *32nd International Conference on Neural Information Processing Systems*, pp. 9872–9883, 2018.

[9] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *International Conference on AI and Statistics*, pp. 2021–2031, 2020.

[10] S. Wang, F. Liu, and H. Xia, "Content-based vehicle selection and resource allocation for federated learning in IoV," in *2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1–7, 2021.

[11] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "Performance optimization of federated learning over wireless networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2019.

[12] 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 Transactions on Wireless Communications*, 2020.

[13] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Joint device scheduling and resource allocation for latency constrained wireless federated learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 453–467, 2021.

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

[15] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1188–1200, 2021.

[16] C. T. Dinh, N. H. Tran, M. N. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 398–409, 2021.

[17] Y. M. Saputra, D. Nguyen, H. T. Dinh, T. X. Vu, E. Dutkiewicz, and S. Chatzinotas, "Federated learning meets contract theory: Economic-efficiency framework for electric vehicle networks," *IEEE Transactions on Mobile Computing*, 2020.

[18] A. Uprety, D. B. Rawat, and J. Li, "Privacy preserving misbehavior detection in IoV using federated machine learning," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–6, 2021.

[19] X. Huang, P. Li, R. Yu, Y. Wu, K. Xie, and S. Xie, "Fedparking: A federated learning based parking space estimation with parked vehicle assisted edge computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9355–9368, 2021.

[20] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23920–23935, 2020.

[21] Q. Kong, F. Yin, R. Lu, B. Li, X. Wang, S. Cui, and P. Zhang, "Privacy-preserving aggregation for federated learning-based navigation in vehicular fog," *IEEE Transactions on Industrial Informatics*, 2021.

[22] S. R. Pokhrel and J. Choi, "A decentralized federated learning approach for connected autonomous vehicles," in *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1–6, 2020.

[23] S. R. Pokhrel and J. Choi, "Federated learning with blockchain for autonomous vehicles: Analysis and design challenges," *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 4734–4746, 2020.

[24] H. Chai, S. Leng, Y. Chen, and K. Zhang, "A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in Internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[25] A. Taik, Z. Mlika, and S. Cherkaoui, "Clustered vehicular federated learning: Process and optimization," *arXiv preprint arXiv:2201.11271*, 2022.

[26] H. Xiao, J. Zhao, Q. Pei, J. Feng, L. Liu, and W. Shi, "Vehicle selection and resource optimization for federated learning in vehicular edge computing," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[27] S. Liu, J. Yu, X. Deng, and S. Wan, "FedCPF: An efficient-communication federated learning approach for vehicular edge computing in 6G communication networks," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[28] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, pp. 19–35, 2021.

[29] NVIDIA, "Hardware for self-driving cars." https://tinyurl.com/59any9fc. [Online; accessed 23-Sep-2022].

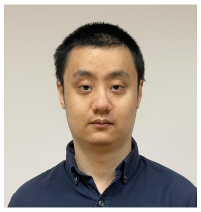
[30] H.-S. Lee and J.-W. Lee, "Adaptive transmission scheduling in wireless networks for asynchronous federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3673–3687, 2021.

[31] R. E. Tarjan, "Two streamlined depth-first search algorithms," *Fundamenta Informaticae*, vol. 9, no. 1, pp. 85–94, 1986.

[32] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[33] C. Li, Q. Liu, S. Li, Y. Chen, Y. T. Hou, W. Lou, and S. Kompella, "Scheduling with age of information guarantee," *IEEE/ACM Transactions on Networking*, 2022.

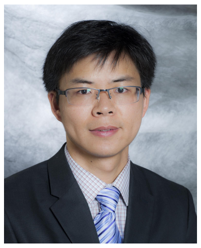
Pedram Kheirkhah Sangdeh received his B.Sc. degree in Electrical Engineering from Iran University of Science and Technology, Tehran, Iran, in 2011, and his M.Sc. degree in Electrical Engineering from the College of Engineering, University of Tehran, Tehran, Iran, in 2014. He is currently working toward his Ph.D. degree in the Department of Computer Science and Engineering at Michigan State University, East Lansing, MI, USA. His research interests include design, performance analysis, and implementation of innovative protocols for intelligent wireless networking.



Chengzhang Li is a Ph.D. student in the Bradley Department of Electrical and Computer Engineering at Virginia Tech, Blacksburg, VA, USA. He received his M.S. degree in computer engineering from Virginia Tech in 2020, and his B.S. degree in Electronics Engineering from Tsinghua University, Beijing, China, in 2017. His current research interests are modeling, analysis and algorithm design for wireless networks, with a focus on Age of Information (AoI), 5G and ultra-low latency research.

Hossein Pirayesh received the B.Sc. degree in Electrical Engineering from Karaj Islamic Azad University, Karaj, Iran, in 2013 and the M.Sc. degree in Electrical Engineering from Iran University of Science and Technology, Tehran, Iran, in 2016. He is currently working toward his Ph.D. degree in the Department of Computer Science and Engineering at Michigan State University, East Lansing, MI, USA. His research focuses are on wireless communications and networking, including theoretical analysis, algorithm and protocol design, and system implementation.

Shichen Zhang is currently a Ph.D. student in the Department of Computer Science and Engineering at Michigan State University (MSU), East Lansing, MI. He received his B.Eng degree in Automation from Beijing University of Technology, Beijing, China, in 2018 and M.Eng degree in Electrical and Computer Engineering from Cornell University, Ithaca, NY, in 2019. His current research interests focus on wireless networks, sensing systems, and machine learning.



Huacheng Zeng (SM'20) received a Ph.D. degree in Computer Engineering from Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, VA. He is currently an Assistant Professor in the Department of Computer Science and Engineering at Michigan State University (MSU), East Lansing, MI. Prior to joining MSU, he was an Assistant Professor of Electrical and Computer Engineering at the University of Louisville, Louisville, KY, and a Senior System Engineer at Marvell Semiconductor, Santa Clara, CA. He is a recipient of the NSF CAREER Award. He received the Best Paper Award from IEEE SECON 2021 and ACM WUWNet 2014. His research interest is broadly in wireless networking and sensing systems.

Y. Thomas Hou (F'14) is Bradley Distinguished Professor of Electrical and Computer Engineering at Virginia Tech, Blacksburg, VA, USA, which he joined in 2002. He received his Ph.D. degree from NYU Tandon School of Engineering (formerly Polytechnic Univ.) in 1998. During 1997 to 2002, he was a Member of Research Staff at Fujitsu Laboratories of America, Sunnyvale, CA, USA. Prof. Hou's current research focuses on developing innovative solutions to complex science and engineering problems arising from wireless and mobile networks. He is also interested in wireless security. He has over 300 papers published in IEEE/ACM journals and conferences. His papers were recognized by nine best paper awards from the IEEE and the ACM. He holds six U.S. patents. He authored/co-authored two graduate textbooks: Applied Optimization Methods for Wireless Networks (Cambridge University Press, 2014) and Cognitive Radio Communications and Networks: Principles and Practices (Academic Press/Elsevier, 2009). Prof. Hou was named an IEEE Fellow for contributions to modeling and optimization of wireless networks. He was/is on the editorial boards of a number of IEEE and ACM transactions and journals. He served as Steering Committee Chair of IEEE INFOCOM conference and was a member of the IEEE Communications Society Board of Governors. He was also a Distinguished Lecturer of the IEEE Communications Society.