Synchronizing Tasks for Distributed Learning in Connected and Autonomous Vehicles

Pawan Subedi, Beichen Yang, and Xiaoyan Hong

***Abstract*—Autonomous driving relies greatly on deep learning to comprehend the surroundings and activities of the road systems. The learning models are traditionally trained off-line and used during driving. However, recent research on federated learning has enabled distributed deep learning for model adaptation with new data inputs from end users. Similarly, the recent research on continual learning has enabled the upgrading of the learned model with newer rounds of training, without losing previously acquired knowledge. For autonomous and/or connected vehicles, these mean it is possible to take new, maybe real time, data inputs from various sensors from multiple vehicles in vicinity, to train and update the preloaded models. The updated models can improve the safety and reduce human involvements when driving through unfamiliar situations. This paper tackles one important issue, namely, the model data dissemination for enabling such a distributed learning. The model data dissemination consists of steps of soliciting workers, transmitting model, and collecting the updates. In a mixed autonomous vehicles and connected vehicles network scenario, communication overhead and network dynamics are the major challenges. We present analyses on critical latency issues pertained to the various aspects of the model data dissemination to gain insights on the feasibility of federated learning for such a scenario. Also, we introduce a communication architecture and a publish-subscribe system for the model data dissemination. Our system is built with an information-centric networking paradigm via a tiered edge network architecture. Such a system organizes the steps of the model data dissemination clearly and manages the dynamics of the participating vehicles easily. The results show that the presented system reduces overall communication overhead and delay. It also provides high resiliency to packet losses in the mixed wireless connected and autonomous vehicular network.**

***Index Terms*—Autonomous vehicles, connected vehicles, continual learning distributed learning, federated learning, model distribution, named data networking, synchronization groups.**

I. Introduction

A vision for autonomous driving has been acknowledged where in the presence of unfamiliar driving conditions (the conditions that did not have a sizable training data at the time of building the AI model), new data from devices on-the-scene, such as from moving vehicles, can help update the AI model and make it available in a timely fashion. Such a vision faces its challenges, especially with the current general practice for model training, which uses centralized computing platforms in the cloud where all the training data needs to be available upfront. Some challenges also relate to communication limitations and data privacy. The latter is strongly scrutinized by the data providers due to the risk of exposing the sensitive information.

Recent work of federated learning (FL) describes a distributed training process involving the edge devices [1], which offers a solution to the aforementioned challenges. With federated learning, the end devices participate in the training process with their local data. It therefore brings two advantages to autonomous driving: Richer dataset from distributed data sources; and built-in privacy preservation because each distributed training process uses own set of data and a participating vehicle's data never leaves the owner. On the other hand, with the progresses on several communication technologies for vehicular communication and networking (e.g.,V2X, 5G), future transportation systems will include a mix of connected vehicles (CVs), autonomous vehicles (AVs) and connected autonomous vehicles (CAVs). All these vehicles are rich in sensors and sensed data, and are able to communicate. Utilizing their local data, computing and communication resources, federated learning can be deployed to bring substantial advantages for model updates and the driving safety of AVs and CAVs. While a general consideration has been to preserve the computing capacity on the AVs and CAVs, the CVs, or even personal devices within the CVs, can be the potential participants in distributed training with their local sensor data. Such a paradigm can be realistic if incentives are given [2], [3].

Although the opportunities provided for autonomous driving by the connected vehicles and federated learning are tremendous, many research endeavours are required to apprehend these possibilities. The researches have been addressing challenges to federated learning relating to heterogeneity of the end devices, statistical heterogeneity of the data at the end devices, emerging privacy requirements, etc [4]. However, one important, yet mostly under-looked issue by these work is the model data dissemination, i.e., transferring the learning model to the vehicles that are able to act as workers and then receiving the updates from them in a dynamic and mobile edge wireless environment. The scenario presents challenging issues in managing several communications tasks, addressing mobility induced network dynamics, and being parsimonious on bandwidth usage.

To fill the gap and address the challenges, we present a

Manuscript received October 28, 2021; revised June 3, 2022; approved for publication by Abbas Jamalipour, Division 2 Editor, July 7, 2022.

*The work was supported partly by the National Science Foundation under Grants No.1719062. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

P. Subedi, B. Yang, and X. Hong are affiliated with Department of Computer Science, The University of Alabama, Tuscaloosa, Alabama, USA, email: {psubedi, byang12}@crimson.ua.edu, hxy@cs.ua.edu.

P. Subedi is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2022.000028

1229-2370/22/$10.00 © 2022 KICS

network system that synchronizes the model data dissemination tasks for distributed learning in CAV environment, terms as Sync4DL in short. It includes a distributed model training management system and a hybrid edge and fog network architecture supporting it. The model training management system uses synchronization groups to manage the four tasks for the data dissemination. These tasks are (1) model acquisition, (2) model dispatch (including worker solicitation and model retrieval), (3) update collection, and (4) update submit. To easily manage the various message exchanges in the different synchronization groups, the system utilizes a pub-sub mechanism. The pub-sub mechanism is carefully designed to simplify various possible modes of communication in distributed manner required in the groups.

The underlying vehicular communication environment for such a system can build on matured multihop vehicular ad hoc network (VANET) and encounter based vehicular delay tolerant network (VDTN) [5]. But the aforementioned mobility challenge leads to the dynamics of the worker groups and the connectivity to the CAVs (which are the users and solicitors), as well as system resilience concerns. Bandwidth challenge also exists when corresponding solutions handling mobility could incur excessive communication overhead. Thus, the hybrid Edge and Fog network architecture is critical to show promises in addressing these challenges.

The architecture has three tiers to facilitate organized communications involved in the four tasks, where each tier includes different networked nodes; and the communications are between nodes from different tiers. The challenges of mobility and bandwidth call for an information-centric approach to reduce the complexity in handling addresses and dynamic topology, and to the promote information sharing through caching.

As such, named-data networking (NDN) is used for the proposed architecture [6] and to support the training management system. We build the pub-sub mechanism for each of the synchronization groups using NDN for the training management system. The publication topic is carried by the name prefix in each packet so a packet is able to identify the synchronization group it represents. Such a pub-sub mechanism, therefore, doesn't require a central broker for maintaining the groups. Further, the sharing nature of NDN communication model allows effective use of wireless broadcast channel, leading to lower communication overhead. In addition, the name based data addressing scheme of NDN eliminates the burden for learning and managing the addresses of the CAVs and CVs in the vehicular scenario with constant topology changes and network disruptions [7], [8]. The advantages and enhancements of NDN over VANET and VDTN are also being discussed in earlier work [5], [9]–[12].

Our main contributions in this paper are summarized below:

- We introduce the design details relating to architecture, synchronization groups, naming and implementation useful for model data dissemination.
- We build the synchronization groups on top of information-centric networking of NDN. Through such design choice, our system has been able to handle the mobility induced participants' dynamics effectively.
- Our system has also been able to substantially reduce the bandwidth and channel contention by enabling reception of a single transmission by multiple nodes, and by enabling in-network caching through NDN.
- We incorporate a tiered system design for our system. By making such a choice, we have been able to achieve high resiliency to packet losses in the mixed connected and autonomous vehicular network.
- We provide extensive evaluations of the system, both analytically and empirically. Through the analytical evaluation, we offer useful insights on the impact of vehicle mobility on the latency in model distribution. Such evaluation have helped in studying the feasibility of using FL for model enhancement under certain vehicular environments.
- Through empirical analysis, we have compared the proposed tiered architecture with a non-tiered architecture. Our analyses consider the different presence ratios of CAVs and CVs. Through the analyses, the proposed system is found to decrease the worker solicitation delay, improve the model transfer completion probability, and decrease the update collection delay. Therefore, our proposed system works effectively and efficiently in terms of the communication overhead and resilience.
- The paper extends our previous work [13] greatly to include synchronization mechanism for the model distribution, and it includes substantially more analytical and empirical results.

The paper is organized as follows. Section II presents related work and a brief background on NDN and NDN-based pub-sub communication. Section III introduces the design of the proposed system. Section IV presents numerical analysis, followed by simulation evaluations in Section V. Section VI concludes the paper.

II. Related Work and Background

A. Related Work on FL

The distributed deep learning usually uses multiple CPUs and GPUs in distributed systems centered at cloud environments [14]. Recently, the use of end devices collectively to provide the computation and storage needs of the learning systems at the edge of the network has emerged, such as federated learning [15]. Federated learning faces performance issues of the learning process and problems posed by the wireless communications, especially those related to connectivity. Some works have analyzed the performance of learning process considering the challenges of wireless network jointly. Nishio *et al.* [16] study the selection of subset of workers considering the workers' resources and channel condition in order to minimize the loss function of the learning. Chen *et al.* [17], [18] analyze the convergence time for the learning process by presenting the selection of the subset of the workers pertaining to the limited bandwidth as the wireless issue. Similarly, several works [19], [20] study the energy efficiency of the worker nodes as the wireless issue for obtaining certain training accuracy. The transmission scheduling policy and the

inter-cell interference in the cellular network are considered as the wireless constraints for federated learning convergence in the works by Yang *et al.* [21]. The similarities of all these existing works can be traced in terms of two factors: All of these are based on one-hop cellular networks and the wireless issues discussed in these papers are all related to the physical properties of the communication channel. None of these work has studied the model distribution and the update collection process as a challenge in a wireless scenario, nor has studied the scenario of a hybrid edge and fog architecture for future transportation system with CAVs and CVs.

B. Named Data Networking

NDN is a network architecture that performs routing and forwarding based on named data pieces [6]. In NDN, data pieces can be hierarchically named. The communication starts when an application, as a customer, expresses an interest (via an *Interest* packet) for a named data piece; a producer, having the name-matched data, responses by returning the data (via a *Data* packet). NDN routers forward both types of packets based on the names of the data, not IP addresses. FIB tables at NDN routers contain pointers to the producers of the data. A NDN router can cache the data pieces when receiving them. This allows a router to return one cached piece of data when its name matches the one expressed in an *Interest* packet. Likely, a single *Interest* packet can be answered by a closer data source other than the original one. Such in-network caching provides not only natural data sharing with reduced network overhead but also resiliency to network disruptions.

In the environment of connected vehicles and connected autonomous vehicles, the broadcast nature of the wireless communications can be explored during the dissemination of both *Interest* and *Data* packets. Typically, an *Interest* packet simply floods through the network, yet a returned *Data* packet can be cached by the nodes who overhear the transmission of *Data* packet. This allows extensive data sharing for the vehicular networking [7], [22].

C. Publish-subscribe System

Publish-subscribe system has been a popular information dissemination model for applications where the communication requirement is beyond point-to-point but involves one-to-many, many-to-one, or many-to-many. In a pub-sub system, the communication is facilitated by the means of named topics [23]. Data creators publish their data to a topic to which one or many consumers subscribe. Whenever a new data is published to the topic, the subscribers are notified by the pub-sub system so that they can fetch the data.

The current implementation of pub-sub model on top of TCP/IP architecture requires mapping from the topic name of the data piece to the node's address through a complex mapping system involving a central broker node. However, the named topic and the sharing nature of the pub-sub model can be better implemented using NDN architecture without the involvement of the broker [24].

The pub-sub model in NDN is also above network layer as a middle-ware. Pub-sub model mostly starts with publishing a new message to a named topic. Given that communication model in NDN is consumer-driven, i.e., the consumer first issues an *Interest* for a named-data piece, publishing a new message (a data piece) to a topic can not be implemented directly because there is no *Interest* for the new *Data*. Several work are proposed to address this proactive data issue through sync protocols [25]–[27]. The concept of long-lived *Interest* packets presented in Chronosync [25] and PSync [26] follows the natural *Interest-Data* communication behavior, but requires *Interest* to be kept for a long while. This causes extra overhead at the nodes for maintaining the network state. The VectorSync scheme gets rid of the long-lived *Interests* by using a leader-based group system architecture to manage the *Interest* and *Data*. It incurs extra overhead in maintaining the groups, especially in a dynamic scenario of vehicular networks. The work described in [24] developed a dataset synchronization (sync) protocol. It does not use long-lived *Interest*, but adds a new sync type of *Interest* and *Data* packets. This addition is used to break one pub-sub messaging pair into two *Interest-Data* packets pairs. The paper uses the scheme to collect network management data. This scheme is the building block for the proposed system.

III. System Design of Sync4DL

A. Network Architecture

The edge-fog communication architecture for the envisioned vehicular environments builds on the underlying network technologies of vehicular-NDN (V-NDN in short) [9]–[12]. The architecture supporting the training model dissemination in Sync4DL consists of three tiers, namely, the tier of edge servers (TES), the tier of connected autonomous vehicles (TAV), and the tier of connected vehicles (TCV). The first tier TES includes the edge servers that serve as distributed distribution centers for the training models. They host the initial models. They also receive model updates and forward them back to the cloud for model aggregation. An edge server connects to backend cloud servers through internet. The second tier TAV consists of the CAVs. They can request models from an ES server or a peer CAV according to their driving needs. These CAVs also act as solicitors for workers to participate in continual learning. They later collect the partially updated models and use them to enhance own driving performance. The third tier TCV consists of the CVs, some of them will participate in distributed training process of continual learning using their own sensor data and computing power. In a more general way, many CAVs can participate in soliciting more CVs to increase capability of the training process. Or, many CAVs can directly benefit from the partial updates sent by the CVs through the broadcast nature of wireless communications. The CAVs are then responsible for forwarding the updates to ESes.

With the proposed system, when new driving situation occurs, a CAV can solicit CVs in proximity to help train a model using CVs' new local data about the situation. The CVs, after receiving the model from the CAV, train the model so that both the environmental conditions as well as the actions taken

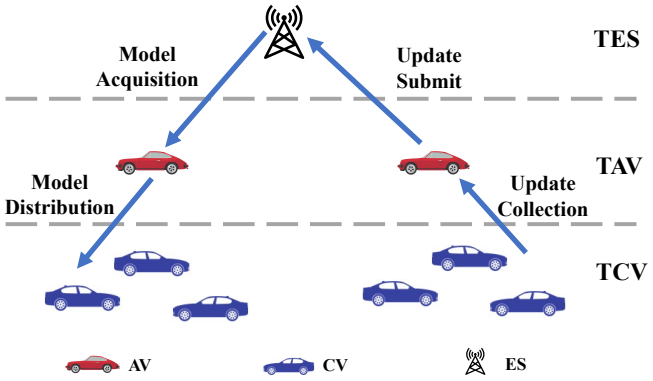


Fig. 1. Three-tier architecture and synchronization groups.

by the driver in a CV based on these conditions can be fed to the model. These worker CVs, upon completing the training, will send the partial updates of the model back to the CAV. The updated model will help the CAVs navigate through the new driving condition. Further, the CAV can submit the partial updates to an edge server. The edge servers are connected via internet to share the partially updated models and aggregate multiple of them. The enhanced model will be hosted by the edge servers for subsequent requests from the CAVs.

To facilitate these communications, synchronization groups (sync groups) are used. The concept of a sync group is defined as an abstract group formed by the nodes publishing or subscribing to the same topic, following [24]. A sync group, therefore, is an abstract of the pub-sub system on a common topic associated with the task such that the communication within the sync groups starts with publishing a new message to a named topic. The pub-sub system organizes message exchanges among multiple publishers (producers) and subscribers (consumers) on the task. Under a common topic, many messages (or data pieces) can be exchanged with more specific titles or tags as well as the contents. This sync mechanism using pub-sub system can enable every types of communication modes: One-to-many, many-to-one as well as many-to-many or any-to-any in a distributed manner.

In the text below, we discuss the details of the pub-sub system and the cases such a pub-sub system needs to handle in Sync4DL. Similarly, we also discuss the design details relating to the convention to name the topic and subsequent message contents to reflect the topics of the sync groups.

B. Pub-sub System

Our pub-sub system builds using the dataset synchronization (sync) protocol (DSP), operating over a sync group [24]. Two special packet types are used in DSP. A *Sync Interest* packet of a topic sent by a publisher to the sync group announces all the names of the data pieces that have been published to the group known by this publisher including his own publishes. This *Sync Interest* packet serves for two purposes. First, this *Interest* packet announces new publications as well as serves as an acknowledgment (or a negative acknowledgment) to others' publications. Upon receiving the *Interest* packet, if a new publication is present, a node sends a general *Interest* packet to retrieve the new publication. Thus, in this case, a three-way message exchange occurs. Second, this packet shows the missed publications at the sender to the other nodes, therefore, serves the purpose of showing interest in new data pieces that other nodes have. Thus, if a receiver discovers this is happening, i.e., it has publications other than the ones listed in the interest packet, it will respond a *Sync Data* packet with the names of the publications that the sender missed. The original sender can then use regular *Interest* and *Data* pair to retrieve the missing publication. To handle many name prefixes, the *Sync Interest* packet typically uses an invertible bloom lookup table (IBLT) [28] to index the names of the publications, which allows quick insertion, deletion, access, and listing of the key-value pair of available publications by the nodes.

C. Naming Scheme and Synchronization Groups

The naming scheme design is an important aspect of overall system design to facilitate the communication within the sync groups. It is the name, reflecting different publication topics of the published data, that binds a sync group together and distinguishes one group from other groups. In addition, the naming scheme will facilitate the routing and forwarding for all the sync groups.

In Sync4DL's name scheme, therefore, the sync packets and the topics of sync groups are of foremost importance in retrieving information. They will be the first two fields. Additional information relating to models is also specified in the name scheme. The details of the name scheme and the semantics of the elements are described below.

$$Sync\,\langle topic\rangle\,\langle model\_fields\rangle\,\langle src\_id\rangle\,\langle seq\_i\rangle$$

- *Sync:* This is a keyword. It identifies the special type of interest and data packets, the *Sync Interest* and *Sync Data* packets, from a regular *Interest* and *Data* packets. A *Sync Interest* is given higher priority and processed by the pub-sub system. A *Sync Interest* might bring back *Sync Data* but not always. Transmission of a *Sync Interest* can be scheduled in a periodic fashion.
- *topic:* This field identifies a sync topic of a particular sync group. The pub-sub system handles the communications within the sync group based on the given topic.
- *model_fields:* In order to identify multiple models, this field can contain multiple subfields. Three common subfields are: (i) *model_id:* It is the identifier of the training model; (ii) *model_version:* A training process involves multiple rounds/epochs. This field identifies the round of the model with *model_id*. It can be the incremental versioning of the trained model; and (iii) *part_number:* It identifies the part of the model that the packet carries.
- *src_id:* It is an optional identifier of the node looking out for the data and could include vague geographic location to help forward the packet.
- *seq_id:* It is the sequence number of this packet when it was created, serving as version control.

Furthermore, as identified in the text above, there are four tasks associated with the sync groups such that each sync group is responsible for one task. These tasks include:

- Acquiring initial new or updated model.
- Soliciting workers and dispatching the model for training.
- Collecting the updates.
- Submitting the updates to edge servers.

The four sync groups inheriting from the aforementioned four tasks are *Model_Acquisition* and *Update_Submit*, two sync groups formed between TES and TAV; and *Model_Dispatch* and *Update_Collection*, two sync groups formed between TAV and TCV. The three tiers and the direction of data dissemination in the four sync groups are shown in Fig. 1. Each sync group is managed by the pub-sub mechanism using its publication topic, which is carried in the name prefix of the *Sync Interest* and *Sync Data* packets.

- *Model_Acquisition Group:* The group is for CAVs to acquire a model or an updated model from ESes. The group topic is *Model_Acquisition*. ESes are the publishers and CAVs are the subscribers. Typically, when one ES publishes a model, multiple CAVs can receive it. This demonstrates the one-to-many mode of communication. The *Sync* and *Model_Acquisition* keywords are used in Sync Interest generated from this group for the dissemination of a name list of available models. Normal *Interest* and *Data* packets are used for subsequent retrieval of a training model by an CAV upon receiving the list.
- *Update_Submit Group:* This group enables the submission of newly collected partial updates of a model from CAVs to ESes for further aggregation. The CAVs publish the updates to the *Update_Submit* topic so that ESes subscribing to the topic will be able to receive them. Typically, one ES can receive from multiple CAVs, the many-to-one mode of communication occurs in this group. The *Sync* and *Update_Submit* keywords along with other model identifying information in the name prefix are used when an AV publishes the update. An ES will use normal *Interest* and *Data* packets to retrieve the updated model from the CAV.
- *Model_Dispatch Group:* The topic is for CAVs to solicit CVs to perform distributed training. Interested CAVs are the publishers and capable CVs are the subscribers. Multiple CAVs can publish the same model or different models asking for updates, while multiple CVs can receive a model they wish to work on. The model flows from the CAVs to the CVs resembling an any-to-any mode of communication. The name prefix in the initial publication (soliciting CVs) consists of *Sync* and *Model_Dispatch* keywords. Subsequent dispatching the model is performed using normal *Interest* and *Data* packets initiated by a CV to retrieve the model from the CAV.
- *Update_Collection Group:* This group is for CAVs to collect partial updates when distributed local training are completed at the CVs. The publishers are the CVs who publish new partial updates, while subscribers are the CAVs. The any-to-any mode of communication describes the information exchange within the group. Similarly, a CV initiates an interest with *Sync* and *Update_Collection* keywords. AVs will use normal *Interest* and *Data* packets to retrieve the update.

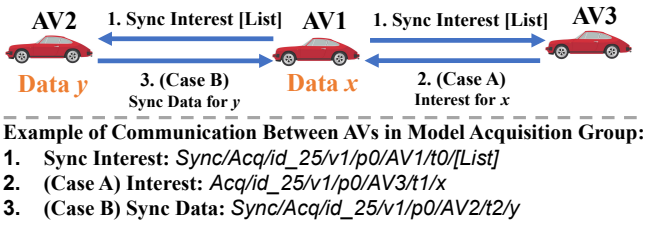


Fig. 2. Synchronization example in model acquisition.

D. Sync Group Communication Example

Here we use an example of message exchanges relating to *Sync Interest* and corresponding responses to illustrate the two ways that (Sync Interest) serves to synchronize the communication tasks. The example comes from the model acquisition task, given in Fig. 2. CAV1 publishes a list of available models through a *Sync Interest* to the topic of *Model_Acquisition* (*Acq* in short in the figure) with other related naming fields. In Case A, CAV3 learns that the *Data x* in the list is what it needs. CAV3 then follows up with a normal NDN *Interest* to the topic *Acq* to retrieve *Data x*. In Case B, CAV2 notices that CAV1 doesn't have *Data y*. Thus, it responds with a *Sync Data* packet with *Data y* included to the topic *Acq*.

E. System Layering

The Sync4DL system is layered according to the dependence of functionalities. The pub-sub communications is built on top of NDN network architecture. The pub-sub communications directly use the network-level *Interest* and *Data* packets, as discussed earlier using *Sync* special type. Fig. 3 shows the function blocks building on top of NDN primitives.

- *Data Generator:* This module handles new data pieces, when they are generated either as a model received from the cloud, or a partial update produced after a local training. The module submits the meta about the data piece (e.g., version number, model Id, etc.) to the *Name Generator*. and obtains a new name to attach to the data piece. This newly produced piece of named data is ready for storage and for publish.
- *Name Generator:* This is the component responsible for creating the names for the data in consideration. It uses the name convention when doing so.
- *Sync Module:* This is the module supporting all the synchronization groups. The pub-sub topics are handled by this module. Serving as a middleware, it connects to primitives in NDN that not only deal with special *Sync* packets but also regular NDN packets.
- *NDN Interface Module:* This is the NDN network that handles the forwarding and receiving of both *Sync* and regular *Interest* and *Data* packets.

F. Discussions on Design Considerations

In this section, we discuss two representative challenges in the training process that are closely-related to our work,

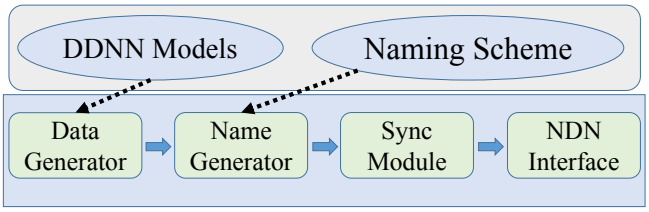


Fig. 3. System modules.

in hope of inspiring future researches. They are regarding the usefulness of training updates from slow workers and the time to stop the training process before the model starts over-fitting the training data. We also offer possible ways that the Sync4DL would address the challenges. Other challenges include the evaluation of minimum CPU and GPU capabilities within the worker nodes so that a training process can be completed timely, mechanisms of various hyper-parameter tuning required for the deep learning process, proper batch size for the training data, and so on. Researches on these issues are beyond the scope of this paper. Interested readers can refer to the seminar paper by Keuper *et al.* [29].

The first challenge is regarding the usefulness of the training updates from slow workers. Currently, with most of the deep learning systems, once the parameters get updated, the new round of training starts with the updated parameters [14]. In case of the distributed system presented as the scenario in this work, cases arise when some workers are late to send back the updates or are still working with the past version of the model and the central system has already updated the parameters such that the faster workers have already started their training process with the current parameters. In such a situation, two different questions are important: first, should the proposed system have a special mechanism such that the slow workers, who haven't yet completed their training process, are notified that their training is lagging behind the current model version? Second, should the lagging worker stop its current training round and restart with the newer version of the parameters? Regarding the first question, Sync4DL already has notifications built-in. The *Sync Interests* sending from CAVs looking for CVs, and the various Data packets sending updated models, all contain the version number of the current training model. In addition, NDN in wireless broadcast allows nodes to process these packets to know of the version updates. Thus, a separate mechanism is not required. Regarding the second question, with the recent development in the continual learning systems [30], we expect even the outdated training updates being useful for the overall learned model. The details of such investigations are outside the scope of this paper.

The second challenge is when should the training process stop or when does the server decide that the learned model is complete and usable for the end AV devices. Such a challenge can be evaluated by being focused on the subtlety of continual learning against the current deep learning systems [31]. With continual learning, a trained model is deemed useful when it passes certain test threshold, but the learning process is lifelong and the model keeps improving with new rounds of training [31]. This is specially important for the autonomous driving since new driving conditions and environments keep emerging and the model needs to be updated for these environments. The advantage of the global system presented in this paper lies in this requirement of continuous improvement of the learned model. Thus, we expect with the maturity of the continual learning systems, the learned model becomes useful at a certain point in time after passing the testing threshold, and becomes more useful with every new round of training, which implies that the training process will indeed be a continuous process. More extensive discussion on this issue is not intended in this paper.

IV. Numerical Analysis

In this section, we will investigate impact of mobility brought by more challenging vehicular communication scenarios. The analyses focus on how well a CV may receive a solicitation for workers and how well the finished updates may be received by a CAV.

Our analyses consider an area with N CAVs, M worker CVs that are willing to participate in the training process, and other CVs in the connected vehicular environments. CAVs and CVs can communicate when they are within the transmission range of one another. The vehicular edge-fog network environment can occur in a mix of VANET and VDTN. The former will help Sync4DL distribute a model in a short time period via multi-hop packet relays. But with the latter, communication opportunities occur only when CVs or CAVs encounter one another.

Five major performance aspects are studied in this section: (1) The latency for workers to receive solicitation *interests*; (2) still the latency, but considering the case that the solicitation messages can expire; (3) the average latency for collecting all possible updates; (4) the latency to collect the first update considering the probability that an update is not ready at a worker; and (5) the number of updates can be received. The various symbols used in the analysis are summarized in Table I.

A. Worker Solicitation Delay

In this subsection, we will investigate what aspects would influence the overall delay of worker solicitation process. Worker solicitation occurs in model dispatch task when a CAV publishes a *Model_Dispatch Interest* for workers. A CV receives the *Interest* is called a holding vehicle. If the holding vehicle is a worker CV, it then follows up to retrieve the model. The solicitation delay is the time needed for a worker CV to meet one of the holding vehicles. The delay mainly constitutes the time needed for encountering a holding vehicle, which may take multiple encounters because only a part of the vehicles are holding vehicles.

The encounter opportunity of two mobile nodes in the VDTN scenario is one important mobility property. Earlier work has shown that the time for any pair of nodes to meet follows a Poisson distribution and thus the inter-meeting time of the pair follows an exponential distribution, Robin *et al.* [32]. The work shows that such a result applies to the

TABLE I
LIST OF THE SYMBOLS USED IN THE ANALYSIS.

Notation	*Description*
N	Number of CAVs in the scenario
M	Number of worker CVs in the scenario
$t_{i,j}(n)$	nth meeting time of vehicles i and j
λ	parameter describing vehicles' meeting in unit time
$\rho_{i,j}(n)$	nth inter-meeting time of vehicle i and j
W	A single worker CV
$R+1$	Total number of CVs in the scenario
F	Absorbing state when W receives the *Interest*
S	State space of Markov chain analysis for worker solicitation
$b_i = N+i$	the number of vehicles having *Interest* at the state $N+i, i=0,1,\cdots,R$
$d_i = (R-i)$	The number of remaining non-worker CVs
$X = x_i$	Random variable describing the trajectory involving transition from state $N+i$ to F
p_{x_i}	The probability of the trajectory $X = x_i$
D_{x_i}	The expected delay along trajectory $X = x_i$
t	A time slot in slotted time analysis
$p(j,t)$	The probability that a worker j has a model at time t.
τ	The probability of model expiring due to ageing
$q(j,t)$	The probability that j didn't receive a model for training at t
$n(i,j,t)$	The probability that worker j receives a model from CAV i at time t
$m(t)$	The expected number of workers having the model up to time t
m^*	The expected number of solicited workers
$E(D_{UC})$	The expectation of delay of update collections for m^* update-ready participating workers
F_y	The encounter progression matrix for a CAV y
$f(i,j)$	The progression of y having encountered CV i, then connecting CV j
K	The set of states of every CVs not having updates
Z	The set of states of every CVs with updates
$\mathbf{R}$	The transition matrix for each CAV
r_{ij}	The transition probability of transiting from encounter of the CV i to the encounter of the CV j
$f(i,j)$	The encounter progression probability from i to j
$p(j)$	The probability of the CV j for having an update
$\mathbf{B}$	A diagonal matrix constructed with $p(j)$
$\overline{\mathbf{B}} = I - \mathbf{B}$	The projection onto the orthogonal complement of subspace of $\mathbf{B}$
$w(i,t)$	The probability of a CAV collecting a new update at a time instance t
$u(j,t)$	The probability that an update from worker j was received at the CAVs by the time t
$v(j,i,t)$	The probability that a worker j establishes a connection with a CAV i to send the update at time t
μ_t	The expected number of participating workers whose updates were received by the CAVs until the time t
$\hat{v}(j,i)$	The probability of establishing a connection between a participating worker j and a CAV i

Manhattan mobility, the random waypoint model, or simply random direction. In addition, various empirical studies on real-life mobility traces of the vehicles have also shown that exponential distribution is followed by the inter-meeting times. Thus, we assume the same for the vehicles in our scenario. Suppose $0 \le t_{i,j}(1) < t_{i,j}(2) < \cdots$ are the series of the time points when two vehicles i and $j (i \ne j)$ meet. The processes $\{t_{i,j}(n), n \ge 1\}, 1 \le i,j \le T, i \ne j$, is the independent Poisson processes with parameter λ. λ describes how many times can a pair of vehicle meet with each other in unit time. Further, let $\rho_{i,j}(n) = t_{i,j}(n+1) - t_{i,j}(n)$ be the n-th inter-meeting time of vehicle i and j. The random variables $\{\rho_{i,j}(n)\}_i, j, n$ are independent with each other and follow the exponential distribution with mean $1/\lambda$. $1/\lambda$ is the expected inter-meeting time before any pair of vehicles meet again with each other.

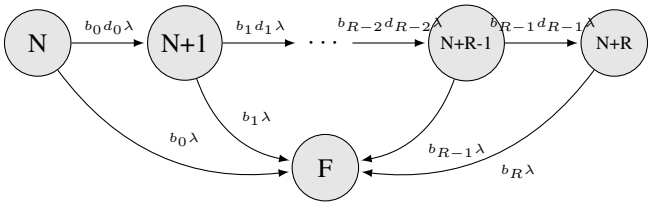


Fig. 4. Markov chain for a single worker solicitation.

A Markov chain model is used in the analysis considering the case of a single worker CV (named W) receiving the solicitation *Interest* in the area with total $R+1$ CVs. Let a state denotes the number of vehicles in the area that have the *Interest*. Here we assume N AV nodes initially hold the *Interest*, i.e., the initial state being N. Let F be the absorbing state when W receives the *Interest*. Let $N+i$ represents the state that ith non-worker CVs have received the *Interest*. State $N+i$ transitions to the next non-absorbing state $N+i+1$ when the *Interest* is transmitted to another non-worker CV. State $N+i$ transitions to the absorbing state F when the *Interest* is transmitted to W. The solicitation process finishes. Thus, the state space, S of the Markov chain includes $R+1$ non-absorbing states and one absorbing state F, denoted as $S = \{N, N+1, N+2, \cdots, N+R-2, N+R-1, N+R, F\}$.

Starting from the initial state N, one CAV encounters one of the R CVs at the rate $R\lambda$, and thus the aggregate transition rate from the state N to $N+1$ is $NR\lambda$. On the other hand, if W receives the solicitation from one of the CAVs, the aggregate rate from state N to the absorbing state F is $N\lambda$. In general, let $b_i = N+i$ be the number of vehicles having the *Interest* at the state $N+i, i=0,1,\cdots,R$, and let $d_i = (R-i)$ be the number of remaining non-worker CVs. The aggregate transition rate from ith state $N+i$ to the $(i+1)$th state $N+i+1$ is $b_i d_i \lambda$, and the aggregate rate from state $N+i$ to the absorbing state F is $b_i\lambda$. These transition rates are shown in Fig. 4.

Let a series of state transitions from $N \to N+1 \to \cdots \to N+i \to F$ be a complete trajectory ending at the absorbing state for W. The random variable X such that $X = x_i$ describes the ith trajectory which transitioning from the last state $N+i$ to F. Further, let p_{x_i} be the probability of the trajectory $X = x_i$, i.e., $P(X = x_i)$. For example, $X = x_0$ means a transition from the state N to F, and $X = x_1$ refers to the transitions from the states N to $N+1$ then to F.

At state $N+i$, there are $N+i$ number of holding vehicles, and $R-i$ non-worker CVs without the solicitation. As seen from Fig 4, two types of transitions would happen to them. The first type happens when one of the former meets one of the latter, hence, transits to state $N+i+1$. There are $(N+i)(R-i)$ possible occurrences of this type. The other happens when one from the former meets the worker W, leading to the absorbing state. There are $N+i$ possible occurrences. Thus, the total possible events are $(N+i)(R-i)+(N+i)$. As such, the

probability for transition from state $N+i$ to $N+i+1$ is $(N+i)(R-i)/(N+i)*(R-i)+(N+i) = (R-i)/(R-i+1)$, whereas the probability for transitioning from state $N+i$ to F is $(N+i)/(N+i)(R-i)+(N+i) = 1/(R-i+1)$.

The occurrence of a particular trajectory takes a series of state transitions. Its probability combines the probabilities of each horizontal transition in Fig. 4 and the transition to state F. Specifically, the probability p_{x_i} for trajectory x_i is given in (1).

$$p_{x_i} = \left[\prod_{j=0}^{i-1} \frac{(R-j)}{(R-j+1)}\right] \frac{1}{R-i+1} \tag{1}$$

Similarly, the delay along a trajectory x_i has to consider a series of times spending at each state waiting for an encounter to happen so to transit to the next state until encountering W. The latter leads to state F. Taking the aggregation factors at each state into consideration, for transitioning between horizontal states, the factor at state $N+j$ is $1/(N+j)(R-j)$; for transitioning to state F from $N+j$, the factor is $1/(N+j)$. The expected delay along trajectory x_i is the summation of the delays at each horizontal states, denoted as D_{x_i}. Recall that the expected pair-wise inter-meeting time is $1/\lambda$, D_{x_i} is given by (2).

$$E[D_{x_i}] = \left[\sum_{j=0}^{i-1} \frac{1}{(N+j)(R-j)\lambda}\right] + \frac{1}{(N+i)\lambda} \tag{2}$$

Take trajectory $X = x_0$ as an example, the probability of N CAVs encountering W so transiting to state F, p_{x_0}, is given as $N/(RN+N) = 1/(1+R)$, where the average delay transitioning from state N to the absorbing state F is $1/(N\lambda)$ due to the aggregation of N CAVs encountering W. Further, take trajectory $X = x_1$ as another example. Trajectory x_1 transits from state N to state $N+1$, then F. The existence of state $N+1$ is based on the event that one of the AVs at state N has passed the *Interest* to one of the non-worker CVs. Thus, the probability is $RN/(RN+N) = R/(R+1)$. At state $N+1$, there are $N+1$ holding vehicles, and $R-1$ non-worker CVs plus the worker W without the *Interest*. Thus, the probability that one of the $N+1$ vehicles passes the *Interest* to W is $(N+1)/(R-1)(N+1)+(N+1) = 1/R$. Combining the two probabilities, we have the probability of trajectory x_1 be $[R/(R+1)]*(1/R) = 1/(R+1)$. The delay of trajectory x_1 has to count the delays at states N and $N+1$, which is given as $1/(N\lambda)+1/((N+1)\lambda)$.

Now, with (1) and (2), the expectation of the solicitation delay over all the trajectories is given by (3).

$$E[D] = \sum_{i=0}^{R} E[D_{x_i}]p_{x_i} \tag{3}$$

Based on (3), the expectation of the solicitation delay at a particular value of λ decreases with the increase in number of nodes having the initial solicitation, N. Also, with the increasing value of N, the rate of change of average delay decreases.

To sum up, the most important factor that influence the Worker solicitation delay is the number of CAVs which initially hold the computation task waiting to be solicited to a particular CV. The more CAVs we can have, the less solicitation delay would that designated CV experience.

B. Worker Solicitation Constrained by Message Lifetime

The next important aspects we would like to explore is how many worker CVs can finally be solicited given a certain time constrain. The need for updating a model usually only stays valid for a certain amount of time. Thus, the knowledge about the number of workers needed for helping the training within a time frame is important. This analysis focuses on the number of participating workers with regards to time. The analysis is based on the assumption that the time is slotted. The duration of the time slot is long enough for establishing a connection and completing the model transmissions. For a time slot t, the reasons that a worker CV doesn't have a model can be either due to the expiration of a received model at time $t-1$, or due to the fact that the worker didn't have a model at time $t-1$ and it doesn't receive a model at time t from any CAV either.

In the analysis, let $p(j,t)$ be the probability that a worker j has a model at time t. And let τ be the probability that the model expires due to aging. Further, suppose $q(j,t)$ be the probability that j didn't receive a model for training at t; and $n(i,j,t)$ be the probability that worker j receives a model from CAV i at time t. Given there are a total of N CAVs, $q(j,t)$ can be expressed in (4). As such, the problem in question can be described by (5).

$$q(j,t) = \prod_{k=1}^{N} (1 - n(k,j,t)) \tag{4}$$

$$1 - p(j,t) = \tau p(j,t-1) + (1 - p(j,t-1))\, q(j,t) \tag{5}$$

Now, let the expected number of workers having the model up to time t be $m(t)$, given by (6).

$$m(t) = \sum_{k=1}^{M} p(k,t) \tag{6}$$

Solve (4) and (5) iteratively. Then assume t be large enough such that the products approach zero, the limits of $p(j,t)$ $q(j,t)$ and $n(k,j,t)$ become $p(j)$, $q(j)$ and $n(i,j)$ respectively. Thus, we obtain the limit $p(j)$ in (7):

$$p(j) = \frac{1-q(j)}{1-q(j)+\tau} = \frac{1-\prod_{l=1}^{N}(1-n(l,k))}{1-\prod_{l=1}^{N}(1-n(l,k))+\tau} \tag{7}$$

Based on (7) and (6) can be replaced by the limit of the expected number of solicited workers, m^*. It is given in (8).

$$m^* = \sum_{k=1}^{M} \frac{1-\prod_{l=1}^{N}(1-n(l,k))}{1-\prod_{l=1}^{N}(1-n(l,k))+\tau} \tag{8}$$

According to (8), if the model doesn't expire, every worker will receive the model for training; whereas if the model expires, the number of workers receiving the model is limited by the value of m^*. Also, if a connectivity couldn't be

Fig. 5. Markov chain for update collection.

established with a CAV within a fixed time limit, a worker will never receive the model for training.

As the conclusion, the network connectivity have a major impact on the number of worker CVs that can be solicited when facing a message lifetime constrain. If the connectivity is good, more worker CVs can receive the computation tasks before the message expire.

C. Expected Delay in Update Collection

Delay brought by update collection process is another factor influence Sync4Dl's performance. The task of *Update_Collection* starts after the participating workers publish the names of the available updates after finishing the training. Here, we derive the latency for overall update collection assuming all the updates are always available. Given the collection process starts after the model distribution, we use the results obtained from the above section, i.e., the limiting expected number of workers m^* and the limiting probability that a worker has a model $p(j)$ from (8) and (7), respectively.

Since the process of update collection is similar to the process of worker solicitation, a Markov chain model can be similarly used to calculate the overall update delay. This Markov chain model for describing the update collection process is shown in Fig. 5. Here, the state denotes the number of participating workers who currently have an update which has not been collected by any CAV yet. For example, that there are m^* workers have the updates initially, so the first state is simply denoted as m^*. Meanwhile, the terminal state T means all the workers have successfully uploaded their updates to the CAVs.

The link between two adjacent states describes one of the participating workers uploads its update to any of the CAVs. So the transfer rate between the state i and state $i-1$ can be denoted as $iN\lambda$. Here, i denotes the number of updates that has not been collected at state i. N is the number of CAVs in the system. λ describes how many times can a pair of vehicle meet with each other in unit time as we have discussed in Section IV-A. Then the state transfer delay between two adjacent states can be simply denoted as the reciprocal of the state transfer rate. For example, the delay of collecting first update is $1/m^*N\lambda$ and the delay of collecting second update is $1/(m^*-1)N\lambda$. As the consequence, the overall expectation of delay of update collections for m^* update-ready participating workers $E(D_{UC})$ is the sum of all the state transfer delay alone the Markov chain, which can be denoted as:

$$E(D_{UC}) = \sum_{i=1}^{m^*} \frac{1}{iN\lambda} \tag{9}$$

And if the vehicular network is dense, there can be a large number of participating workers who have updates to be collected. So m^* is a large number. $E(D_{UC})$ can be further approximately equal to $(ln(m^*)+\gamma)/(N\lambda)$. Here, γ is the Euler-Mascheroni constant, which is approximately 0.57721.

In short, the update collection delay is determined by the number of CAVs. The more CAVs present in the system, the quicker the updated model can be collected by them.

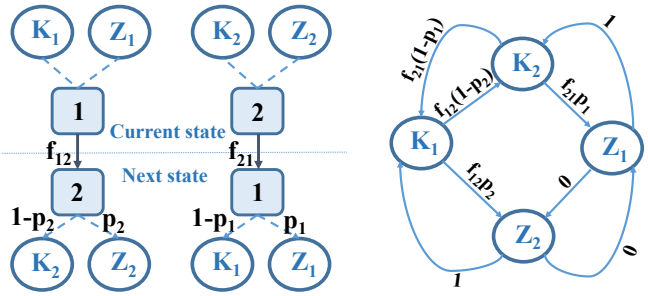


Fig. 6. An example state transition of two nodes for the update collection and the respective Markov chain.

D. Delay in Collecting the First Update

It is important to know how soon the first update can be received by a CAV after the name of the available update is published. Like in a real situation, in this analysis, we consider the case that a CV having an update be a random act according to the encounter history.

To analyze when the first update can be collected by a CAV, we start with capturing the mobility history that a CAV may encounter several non-participating worker CVs before encountering the one with an update, we use an encounter progression matrix. For a CAV y, let F_y be its encounter progression matrix. An element (i,j) in the matrix is the probability $f(i,j)$ that describes the progression of y having encountered CV i, then connecting CV j. Given there are total M worker CVs, F_y is an $M \times M$ matrix as shown below 10, where $f(k,k) = 1, k = 1, 2, \cdots, M$.

$$F_y = \begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,M) \\ f(2,1) & f(2,2) & \cdots & f(2,M) \\ & & \cdots & \\ f(M,1) & f(M,2) & \cdots & f(M,M) \end{bmatrix} \tag{10}$$

The update collection process can be described similar to worker solicitation in Section IV-A. A CAV y may encounter multiple CVs without being able to collect any updates until the time of meeting the CV i. Now, during the encounter progression, if the newly connected CV j has an update, we say the progression yielded an update collection for the AV.

To analyze the encounter progression for y, let's create a Markov chain where the state space includes every possible state for all the CVs in our analysis. Regarding every possible state for a CV, it can either have the update or not, meaning these two states completely define the possible states for a CV. As such let's create two different sets from every possible state in our analysis. The first set $K = \{k_1, k_2, \cdots, k_m\}$ contains the states for not having update of every CV, and the second set $Z = \{Z_1, Z_2, \cdots, Z_m\}$ contains the states for having update of every CV.

With the above definition of every possible states in terms of two sets, let's first define the transition matrix $\mathbf{R}$ for

the CAV y based on the entries in the matrix. An entry (i, j) in $\mathbf{R}$ is the transition probability r_{ij} for transiting from encounter of the CV i to the encounter of the CV j, where the transition probability r_{ij} is expressed in terms of the encounter progression probability $f(i, j)$ and the probability of the CV j for having an update, $p(j)$, as given by (11). Fig. 6 presents an example scenario involving two nodes for the transition. For the two nodes scenario, the AV y could have been initially in connection with either of the two nodes. Similarly, these two nodes could have been in either of their own two states (having updates or not having updates). Now, if the AV y moves such that there is the change in the connectivity, it either moves from the connectivity of node 1 to node 2 or from node 2 to node 1. These new nodes could have been in either of their states with the respective probability of the states. Based on this, we construct the Markov chain. One thing that needs to be noticed in Fig. 6 is that if the node transitions from a state of already having update to a new state without an update, we have the transition probability to be 1 since the AV already has the required update. Similarly, we have the probability of transition from state of already having update to the new state of already having update, the transition probability is 0, as we do not allow such transitions in our scenario (of first update collection by an AV).

$$r(i,j) = \begin{cases} f(i,j)(1-p(j)), & \text{if } State(i) \in K, State(j) \in K \\ f(i,j)p(j), & \text{if } State(i) \in K, State(j) \in Z \\ 1, & \text{if } State(i) \in Z, State(j) \in K \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

Based on the definition of the entries, we then construct the entire matrix $\mathbf{R}$. For this, we create a separate $M \times M$ diagonal matrix, $\mathbf{B}$, where each diagonal element is the probability, $p(j)$ for each CV. For example, the diagonal element in third row and third column has the probability of having update for the CV 3. Further, we define $\overline{\mathbf{B}} = I - \mathbf{B}$, where I is the identity matrix. Therefore, using (10) and (11), and matrix $\mathbf{B}$, the transition matrix $\mathbf{R}$ is expressed by (12), where each term is $M \times M$ matrix.

$$R = \begin{bmatrix} \mathbf{F_y}\overline{\mathbf{B}} & \mathbf{F_y}\mathbf{B} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{12}$$

Thus, based on (12), we can conclude that the transitions follow the characteristics of a terminating Markov chain, for which, the upper triangular part $\mathbf{F_y}\overline{\mathbf{B}}$ entirely characterizes the transition matrix. Matrices $\mathbf{0}$ and $\mathbf{1}$ are $M \times M$ matrices with all elements being 0 or 1, respectively. Based on the terminating Markov chain of the transitions, we next derive the probability of a CAV collecting a new update at a time instance t, $w(i, t)$. For such, if $B(i, 0)$ is the initial connectivity of AV i, based on the phase type distribution, the probability that an update is collected by an AV i at time t for the first time, $w(i, t)$ is given by (13).

$$w(i,t) = B(i,0)\left(\prod_{k=1}^{t-1} F_i\overline{\mathbf{B}}\right)(F_i\mathbf{B}) \tag{13}$$

Then the expectation delay of collecting the first update for CAV i can then be represented as,

$$D_i^{CAV-Update} = \sum_{t=0}^{\infty} t * w(i,t). \tag{14}$$

Thus, as seen from (14), the latency of the first update collection decreases with the increase in the probability of the CVs having model for training.

E. Number of Collected Updates

Lastly, we would also like to study how many update can be collected within a certain time constrain. In *Update_Collection* task, each worker produces a unique update for the model. CAVs will collect as many unique updates from the participating workers as possible. Our interest is how many updates can be collected. The question is equivalent to how many participating workers, those with the updates, a CAV will meet. Without loss of generality, we assume the training process takes the same time at the workers. The analysis on the expected number of unique updates received by CAVs is similar to Section IV-A. Here, let j be the worker who currently has an updated model waiting to be uploaded. Let $u(j, t)$ be the probability that an update from worker j was received at the CAVs by the time t. Also, let $v(j, i, t)$ be the probability that a worker j establishes a connection with a CAV i to send the update at time t. Thus, the probability that the update from the worker was received at the CAVs by the time t is given by (15).

$$u(j,t) = u(j,t-1) + (1-u(j,t-1))\left[1 - \prod_{k=1}^{N}(1-v(j,k,t))\right] \tag{15}$$

As such, the expected number of participating workers (mu_t) whose updates were received by the CAVs until the time t can be expressed by (16).

$$\mu_t = \sum_{j=1}^{m(t)} u(j,t) \tag{16}$$

Now, let us assume expected probability of establishing a connection between a participating worker j and a CAV i be $\hat{v}(j, i)$. By using the same approach for deriving the (7), we can expand the (15) and get the following:

$$u(j,t) = 1 - \left[\prod_{k=1}^{N}(1-\hat{v}(j,i))\right]^t. \tag{17}$$

When t approaches infinity, $u(j, t)$ approaches 1. Thus, the limiting value of the updates that a CAV receives, μ_∞, can be represented in (18). This means that all the updates will eventually be received by the CAV if the time limit is optimal.

$$\mu_\infty = \sum_{j=1}^{m(\infty)} 1 = m^* \tag{18}$$

As what we have shown in (8), $m(\infty) = m^*$ which means the number of workers receiving the training model when

giving enough time. In other word, if there is no boundary of time, the number of updates can be collected by CAVs purely depends on the the number of worker CVs who can receive the training model.

V. Performance Evaluation

Sync4DL enhances the performance of managing distributed learning tasks by adopting *V-NDN*, and by the three-tier synchronization architecture. Hence, in this section, we evaluate Sync4DL's performance in terms of the two aspects respectively. The first aspect will be evaluated by comparing the network performance when using directly the NDN-based or the IP-based system in the model dispatch task. We do not present the plots for the simulation results for these comparisons but provide useful discussions on them for the completeness of this work. The interested reader is referred to our previous conference paper [13] for the plots. Note that without tiers, the ESs will distribute a model to the worker CVs, and subsequently collecting updates from the workers. The second aspect will be evaluated by comparing the network performance of the direct NDN-based system with Sync4DL in the model dispatch task. For Sync4DL, model dispatch is initiated by CAVs. The results will still be comparable because CAVs are able to acquire a model at a separate time. The connected vehicle scenario considered in the evaluation includes both ESs and CVs. They communicate using VANET protocols, where multi-hop networking occurs between the vehicles but mobility may cause link break and change the network topology constantly.

The open source network simulator ndnSIM is used. ndnSIM implements NDN at the packet-level with full details based on NS3 [33]. For our simulation, we adopt the CCLF modification to ndnSIM, which provides a forwarding strategy called as content connectivity and location-aware forwarding (CCLF) for VANET applications [8]. In our simulation, configurations are made to generate V-NDN scenarios. For Sync4DL, the 3-tier architecture is logical. Its implementation is at application layer by configurations where a random subset of vehicles are CAVs acting at TAV, and another random subset of vehicles are the worker CVs. Their role in the communication is determined by the sync group they are in. For the model dispatch task used in the simulation, a CAV in TAV publishes a model, and the worker CVs subscribe to the model published by sending an *Interest* for the model. Any CAV in TAV can respond back with the data packet of the model. The *sync Interest* for Model_Dispatch is implemented by adding the keyword following the naming scheme. The vehicle mobility traffic used in the simulations is generated by the traffic simulator SUMO [34], in our case, an urban scenario. The mobility traces are then used by ndnSIM for evaluating the network performance.

A. Simulation Configuration

A few network setups are designed for the simulation. For evaluating the performance of direct NDN-based system, two setups are used. The first setup has only one edge server, while the second setup has four edge servers. Correspondingly, we have NDN-ES1 and NDN-ES4 for the direct NDN-based communication. For evaluating the Sync4DL-based communication, additional setups are made to capture potential different numbers of CAVs in the network that Sync4DL will use. As such, we have three setups NDN-CAV0.2, NDN-CAV0.3, and NDN-CAV0.4. The CAV penetration rates are 0.2, 0.3, and 0.4, respectively. Such setups allow the simulation to run multiple times with different overall vehicles. The comparisons will be made with the setups of NDN-ES1 and NDN-ES4.

A one-square-kilometer area map in urban San Francisco is imported to SUMO for the vehicle mobility simulation. The area chosen consists of 9 streets and 7 avenues with some alleys to better represent general urban scenario. Vehicles are initially placed at random positions within the area. These vehicles then select their random destinations within the area and move towards them. The speed of the vehicles is determined by the actual speed limits of the road sections. When a vehicle reaches its destination, it picks another random destination and proceeds until the end of simulation at 100 s. In NDN-CS1, one edge server is present at the center of the area, whereas in NDN-CS4, four ESes are placed in the four midpoints of the four subareas on the map. The transmission range of the edge servers is limited to 100m in the simulation. Thus, multi-hop communication between the ESes, the CVs, and the CAVs will occur in forwarding packets. A random subset of vehicles are selected as the worker CVs that participate in actual training process. Another random subset of the vehicles are selected as the CAVs. These configurations are the same for both the basic NDN and Sync4DL systems. In all the simulations, only one model is available for dispatching by the producers in the case of NDN.

B. Evaluation Metrics

The following metrics are used to evaluate the performances.

- *Satisfaction ratio (SR):* The metric shows how successfully a system can deliver messages in a dynamic mobile network. For NDN-based schemes, SR is the overall number of satisfied *Interests* per total *Interests* initiated during the simulation. For IP-based schemes, it is the ratio of overall number of packets received at the server to the number of request packets sent by vehicles.
- *Average delay:* It measures how quickly a request brings back the data packet, i.e., the time after *Interest* is issued and until the reception of the *Data* for NDN-based systems. The average delay is normalized so it is the ratio of the sum of all the delays to the count of such *Interests*. Equivalently, for IP-based systems, it is all the delays counting from the time from sending request till receiving *Data* normalized to the total number of such requests.
- *Average hop counts:* Average hop counts is the average number of hops travelled by the successfully received *Data* packets normalized to the total satisfied *Interests* for NDN-based systems. Equivalently, for IP-based systems, it counts the number of hops travelled by the *Data* packets averaged over the total number of *Data* packets.

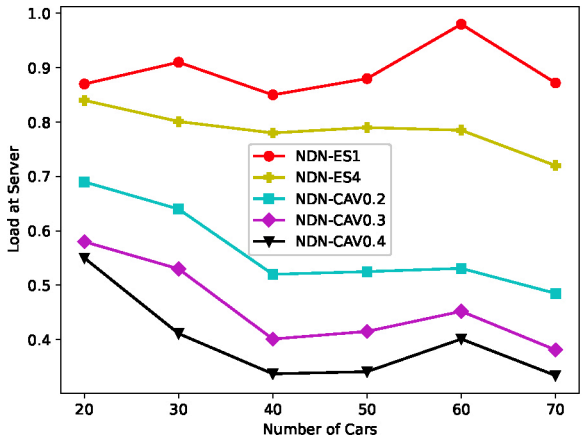


Fig. 7. Server load comparing NDN and Sync4DL with varying total cars.

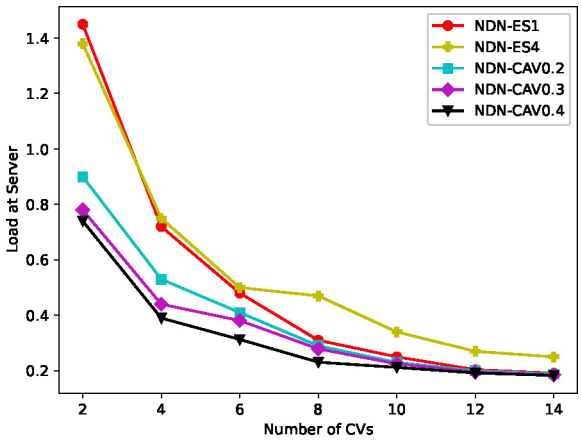


Fig. 8. Load at server comparing NDN and Sync4DL with varying number of CVs.

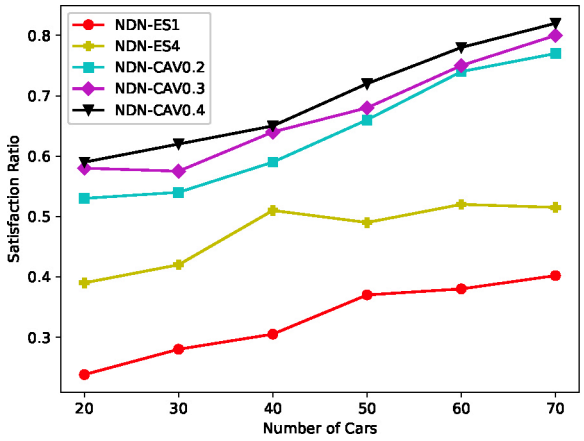


Fig. 9. Satisfaction ratio comparing NDN and Sync4DL with varying total cars.

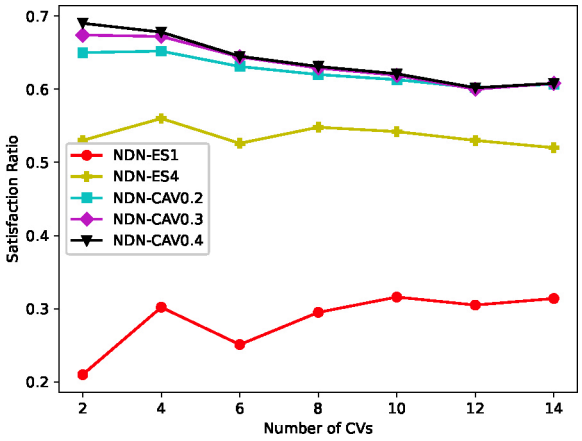


Fig. 10. Satisfaction ratio comparing NDN and Sync4DL with varying number of CVs.

- *Normalized interest transmissions:* It is the overhead created by the *Interest* packets in the wireless channel, defined as the total number of *Interests* created in the network throughout the simulation by total number of satisfied *Interests*.
- *Load at server:* This measures the duplicate *Interests* as extra load at the edge server(s). It is calculated as the total number of *Interests* received at the producers per each satisfied *Interest*.

C. Evaluation Results

In this section, we first provide a discussion on the comparison between the IP-based communication system and the NDN-based communication system (figures not included). Then, we present the results of comparison of direct NDN-based system to the proposed Sync4DL system.

The average delay measure for the satisfied requests for both IP-based and direct NDN-based system are similar, due to only a few requests being satisfied, and the inherent caching of NDN not being able to provide substantial improvement in the delay measure. The delay measure,however, shows better performance for direct NDN-based system in case of the scenario involving 4 edge servers. This is due to the fact that with 4 edge servers, more requests get satisfied, and the inherent caching provided by the NDN network helps in early satisfaction of later requests.

While the average delay of the satisfied requests between NDN and IP systems are comparable for single edge server setup, the average number of hops travelled by such requests differ more drastically. For various number of cars in the simulation, the average number of hops travelled by the *Data* packets is almost always lower by 1 for NDN-based system in comparison to that for the IP system. This shows that some

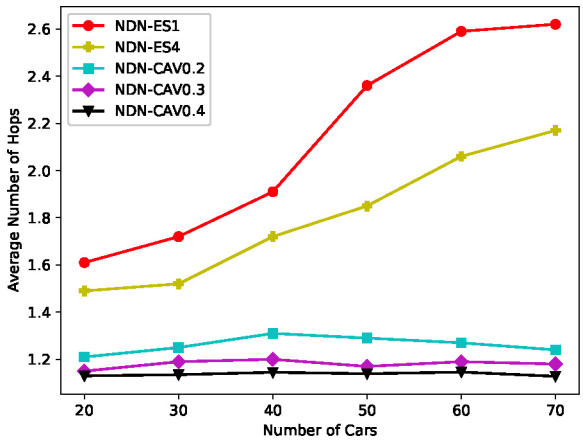


Fig. 11. Average hops comparing NDN and Sync4DL with varying total cars.

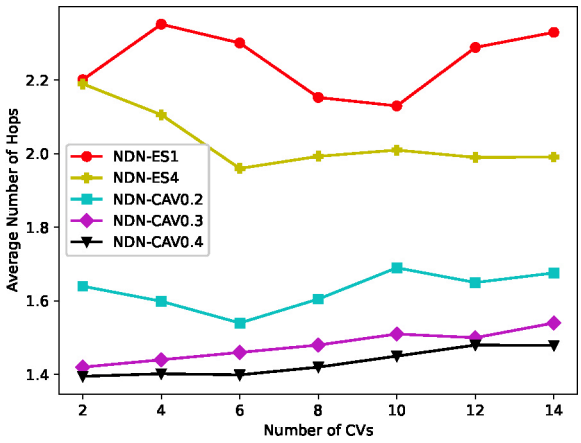


Fig. 12. Average hops comparing NDN and Sync4DL with varying number of CVs.

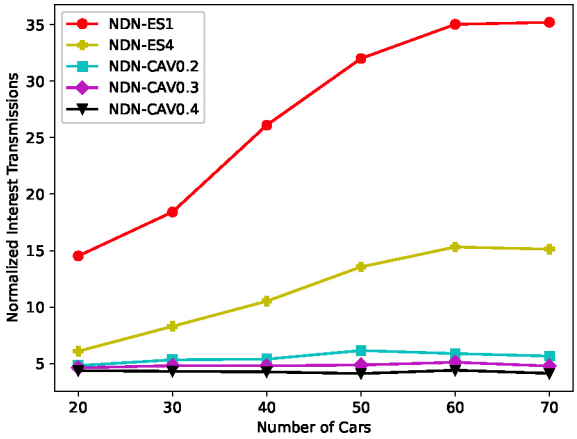


Fig. 13. Normalized Interest transmissions comparing NDN and Sync4DL with varying total cars.

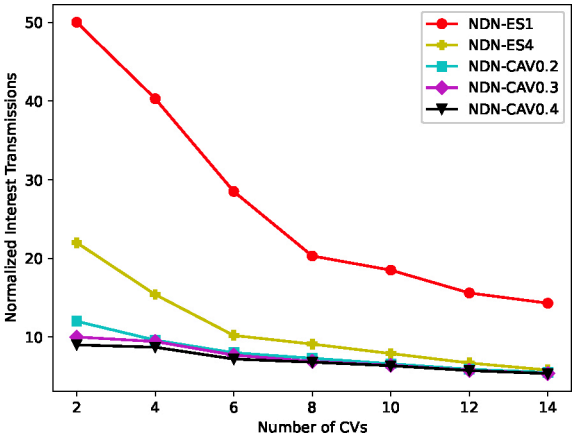


Fig. 14. Normalized Interest transmissions comparing NDN and Sync4DL with varying number of CVs.

of the few satisfied requests for the single edge server setup were quenched from the nearby cache provided by the NDN network. Similar differences can be found for the four server setup as well. The higher number of edge servers provide nearby data source for both IP system and NDN system for the four server setup, but the presence of caching in NDN further improves the hop counts.

The satisfaction ratio for both NDN and IP systems improve with the presence of more sources in 4 edge server setups. Similar to the previous metrics, however, the NDN-based systems have comparatively better satisfaction ratios than IP-based systems.

Figs. 7 and 8 compare the performance of direct NDN-based system and the proposed Sync4DL with respect to the metric of load at server. As seen from Fig. 7, with the proposed solution, evidenced by the lower three lines, there is substantial decrease in the load at the edge server(s) since much of the *Interests* generated in the scenario are satisfied by the CAVs in the TES. Furthermore, with higher penetration rates of CAVs, the load at server decreases more as seen from the comparison between NDN-CAV0.2, NDNCAV0.3, and NDN-CAV0.4. Fig. 8 presents the comparison of load at server against the number of CVs in the simulation scenario. For all five setups, the load at server decreases with the increase in the number of CVs. This occurs as the later *Interests* from the CVs can be satisfied by the cached contents of the earlier CVs that have already received them. Similar to Fig. 7, Sync4DL has better performance than direct NDN, but the difference gets less significant with the increase of number of CVs. This is because, with the higher number of CVs in comparison to the number CAVs, the caches in CVs outsmart those at the CAVs.

The performance comparison between direct NDN-based system and Sync4DL for satisfaction ratio is shown in

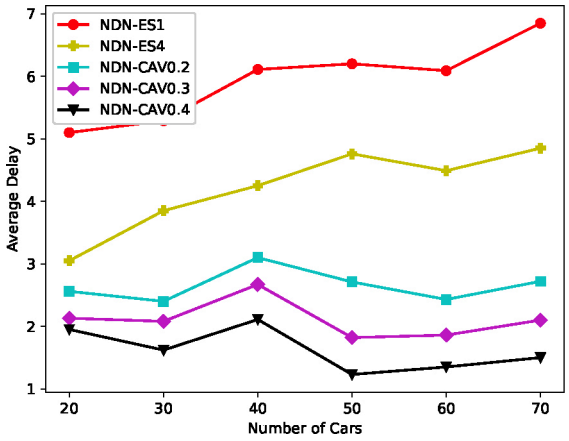


Fig. 15. Average delay comparing NDN and Sync4DL with varying total cars.

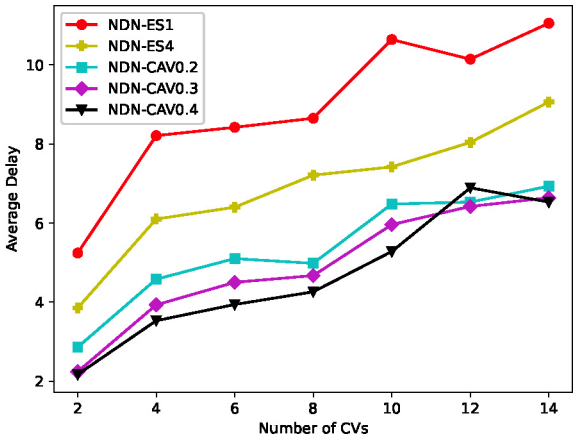


Fig. 16. Average delay comparing NDN and Sync4DL with varying number of CVs.

Figs. 9 and 10. As seen from Fig. 9, the proposed Sync4DL system has very good resilience to the packet losses evidenced with the high satisfaction ratio. Another important observation from the figure is that the satisfaction ratio for the proposed system with different CAV penetration rate is almost similar, meaning even with lower number of CAVs present in the TES, the system becomes resilient to packet losses which clearly proves the advantage of the proposed system. Direct NDN-based system with single ES setup has very low satisfaction ratio, since it is very susceptible to the packet losses in the wireless media. Fig. 10 show that the number of CVs (consumers) in the simulation have minimal impact for the measure of average satisfaction ratio, as evidenced by almost flat curves for all the compared systems.

Similarly, as seen from Figs. 11 and 12, the proposed system has an advantage in terms of number of hops a successful *data* packet has to travel. The presence of the CAVs with the data in the CV's neighborhood in the proposed system help in reducing the number of hops. Another important observation from Fig. 11 is that while the average number of hops for the direct NDN-based system increases with the increase in number of cars in the simulation, for Sync4DL with any CAV penetration ratio, the curves remain flat in spite of the change in number of cars in the simulation. This indicates that the CAVs are more reliable sources of the data than the inherent caching provided by NDN networks. Furthermore, Fig. 12 show that the number of worker CVs (consumers) have no impact in the average number of hops, as the curves for all the five systems remain almost flat with increasing number of CVs for fixed number of total cars in the simulation.

Similarly, the proposed system reduces the wireless transmissions overhead in terms of *Interest* packets significantly as evidenced from Figs. 13 and 14. The advantage of lower transmissions is that the wireless medium is less burdened, which results in lower collision probabilities and eventually higher throughput. Similar to the trends in the average hop counts, while the *Interest* transmission overhead for direct NDN-based system increases with the increase of cars in the simulation, that for the proposed system (with any CAV penetration ratio) remains almost constant. This further strengthens the claim of system resiliency and reliability with the tiered solution. While the number of CVs in the simulation had no impact in the average number of hops, the *Interest* transmissions for direct NDN-based system decrease with the increase in the number of CVs, especially for the case of NDN-ES1 setup. This phenomenon occurs because for a very few *Interest* satisfied in the NDN-ES1 setup, the later requests from the CVs get satisfied by the data present in the previous CVs that have received them. The probability of occurrence of such a phenomenon increases with the increase in the number of CVs (consumers).

At the same time, the average delay for the satisfied *Interest* is also significantly lower for the proposed system as evidenced by the Figs. 15 and 16. An important observation from Fig. 15 is that unlike previous metrics, the average delay metric is less impacted by the number of cars in the simulation. Similarly, as seen from Fig. 16, unlike other metrics discussed before, average delay increases with the increasing number of CVs (consumers) in the simulation for all the five setups. This happens because of the contention at the channel with higher number of consumers.

VI. Conclusion

The paper introduced a three-tier architecture and a pub-sub model for training model distribution. Specifically, multiple pub-sub channels are used to organize multiple phases involved in the distribution of training model to worker vehicles as well as to connected autonomous vehicles as users. The pub-sub model builds on top of NDN platform, which helps to reduce communication overhead, overall delay, and to simplify routing in dynamic network topology. Such a system can also provide high resiliency to packet losses in the mixed wireless connected and autonomous vehicular networks.

REFERENCES

[1] H. B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[2] J. Kang *et al.*, "Incentive design for efficient federated learning in mobile networks: A contract theory approach," in *Proc. IEEE APWCS*, 2019.

[3] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6360–6368, 2020.

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

[5] A. Hamza-Cherif, K. Boussetta, G. Diaz, and F. Lahfa, "Performance evaluation and comparative study of main vdtn routing protocols under small-and large-scale scenarios," *Ad Hoc Netw.*, vol. 81, pp. 122–142, 2018.

[6] A. Afanasyev *et al.*, "A brief introduction to named data networking," in *Proc. IEEE MILCOM*, 2018.

[7] G. Grassi, D. Pesavento, G. Pau, L. Zhang, and S. Fdida, "Navigo: Interest forwarding by geolocations in vehicular named data networking," in *Proc. IEEE WoWMoM*, 2015.

[8] M. Chowdhury, J. A. Khan, and L. Wang, "Leveraging content connectivity and location awareness for adaptive forwarding in NDN-based mobile ad hoc networks," in *Proc. ACM ICN*, 2020.

[9] M. Meisel, V. Pappas, and L. Zhang, "Ad hoc networking via named data," in *Proc. ACM MobiArch*, 2010.

[10] G. Grassi *et al.*, "Vanet via named data networking," in *Proc. INFOCOM WKSHPS*, 2014.

[11] M. Kuai, X. Hong, and Q. Yu, "Delay-tolerant forwarding strategy for named data networking in vehicular environment," *Int. J. Ad Hoc and Ubiquitous Comput.*, vol. 31, no. 1, pp 1–12, 2019.

[12] M. Kuai and X. Hong, "Location-based deferred broadcast for ad-hoc named data networking," *Future Internet*, vol. 11, no. 6, pp. 139–156, 2019.

[13] P. Subedi, B. Yang, and X. Hong, "A fog-based IoV for distributed learning in autonomous vehicles," in *Proc. EAI MONAMI*, 2021.

[14] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," *ACM Comput. Surveys*, vol. 52, no. 4, pp. 1–43, 2019.

[15] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intelligent Syst. and Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[16] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE ICC*, 2019.

[17] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time minimization of federated learning over wireless networks," in *Proc. IEEE ICC*, 2020.

[18] M. Chen *et al.*, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, Jan. 2021.

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

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

[21] H. H. Yang, Z. Liu, T. Q. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 317–333, 2019.

[22] D. O. Mau, Y. Zhang, T. Taleb, and M. Chen, "Vehicular internetworking via named data-an opnet simulation study," in *Proc. EAI TRIDENTCOM*, 2014.

[23] "Mqtt - the standard for iot messaging," [Online]. Available: https://mqtt.org/.

[24] K. Nichols, "Lessons learned building a secure network measurement framework using basic NDN," in *Proc. ACM ICN*, 2019.

[25] Z. Zhu and A. Afanasyev, "Let's chronosync: Decentralized dataset state synchronization in named data networking," in *Proc. IEEE ICNP*, 2013.

[26] M. Zhang, V. Lehman, and L. Wang, "Scalable name-based data synchronization for named data networking," in *Proc. IEEE INFOCOM*, 2017.

[27] W. Shang, A. Afanasyev, and L. Zhang, "Vectorsync: distributed dataset synchronization over named data networking," in *Proc. ACM ICN*, 2017, pp. 192–193.

[28] M. T. Goodrich and M. Mitzenmacher, "Invertible bloom lookup tables," in *Proc. Allerton*, 2011.

[29] J. Keuper and F.-J. Preundt, "Distributed training of deep neural networks: Theoretical and practical limits of parallel scalability," in *Proc. IEEE MLHPC*, 2016.

[30] S. Stojanov *et al.*, "Incremental object learning from contiguous views," in *Proc. IEEE/CVF CVPR*, 2019.

[31] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

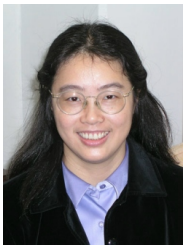
[32] R. Groenevelt, P. Nain, and G. Koole, "The message delay in mobile ad hoc networks," *Performance Evaluation*, vol. 62, no. 1–4, pp. 210–228, 2005.

[33] "Ns-3 based named data networking (NDN) simulator," Apr. 2021. [Online]. Available: https://ndnsim.net/current/intro.html

[34] P. A. Lopez *et al.*, "Microscopic traffic simulation using sumo," in *Proc. IEEE ITSC*, 2018.

Pawan Subedi is a Ph.D. candidate in the Department of Computer Science at the University of Alabama, USA. He received his BE in Electronics and Communication Engineering from Tribhuvan University in 2014. His current research interest includes information-centric networking, distributed systems, and vehicular networks.

Beichen Yang (Student Member, IEEE) received the B.S. degree in Automation (Traffic Information Engineering and Control) from ChangÁn University, XiÁn, China, in July 2015, and M.S. degree in Computer Science from University of California, Irvine, in August 2017. He is currently pursuing the Ph.D. degree in Computer Science at the University of Alabama. His current research interests include wireless network, vehicular communication network, and fog computing.



Xiaoyan Hong is an Associate Professor in the Computer Science Department at the University of Alabama. She received her Ph.D. degree in Computer Science from the University of California, Los Angeles in 2003. Her research interests include mobile and wireless networks, vehicular ad-hoc networks, wireless sensor networks, information-centric networking, and underwater internet of things. Her research focuses on network architecture, routing protocols, mobility, security and privacy issues. Her work also covers distributed machine learning / federated learning in edge computing and fog computing environments. Dr. Hong is a Senior Member of IEEE.