Federated learning for vehicular coordination use cases

László Toka*, Márk Konrád,
István Pelle, Balázs Sonkoly, Marcell Szabó
TMIT, VIK, Budapest University of Technology and Economics,
ELKH-BME Cloud Applications Research Group,
MTA-BME Network Softwarization Research Group, Hungary

Bhavishya Sharma, Shashwat Kumar,
Madhuri Annavazzala, Sree Teja Deekshitula,
Antony Franklin A
Indian Institute of Technology Hyderabad, India
*Corresponding: toka.laszlo@vik.bme.hu

***Abstract*—Vehicular coordination and communication tasks are crucial aspects of enabling autonomous driving, guaranteeing safety and efficiency. In our present work, we explore methods for collecting and distributing information among participants by employing collaboratively-built high-definition maps that contain fine-grained contextual data. We leverage a hierarchical federated learning structure and anticipatory onboarding of the maps through a mobility-aware content caching scheme and minimize the delay of data delivery in both subsystems. We provide analytical models built on queuing theory and integer linear programming and evaluate essential system parameters in an emulation testbed. Based on our results, we conclude that we can significantly reduce the delay in delivering timely information to vehicular clients by introducing intermediary layers in the federated learning structure and by pre-loading current map tiles corresponding to vehicle paths.**

***Index Terms*—vehicular communication, edge platform, content distribution, federated learning, end-to-end latency**

I. Introduction

Through diverse virtualization options, cloud computing has started to replace high-resource local compute equipment, so much so that by now, it weaves through our everyday lives. It backs some of the most mundane applications that we use, most often accompanied by artificial intelligence (AI) solutions. While for some time, the direction of moving compute tasks was clearly targeting the cloud, in recent years—due to latency or data location considerations—this trend has taken a slightly modified trajectory with the appearance of edge and fog computing. Software leveraging these concepts and respective devices can run closer to end-users and provide lower response times for an enhanced user experience. The process is further fueled by the newer generations of mobile networks that provide edge computing resources that are already being leveraged by public cloud providers [1]–[3].

This work was supported by: a) the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund through projects i) no. 135074 under the FK_20 funding scheme, ii) 2019-2.1.13-TÉT_IN-2020-00021 under the 2019-2.1.13-TÉT-IN funding scheme, iii) 2019-2.1.11-TÉT-2020-00183 under the 2019-2.1.11-TÉT funding scheme, b) the ÚNKP-22-5-BME-317 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund, and c) the Department of Science and Technology (International Cooperation Division), Government of India through the project "Autonomous driving enabling fog computing platform with edge cloud orchestration and edge analytics". L. Toka was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

Although at a much slower pace, a similar direction is observable in the case of vehicular control tasks, albeit incorporating the edge concept right from the beginning. While initially vehicles relied solely on onboard electronics for providing driving assistance, coupling them with roadside units (RSU) via vehicle-to-infrastructure (V2I) communication enabled them to extend their sensing and computing powers. As the increase in vehicular traffic infers more issues from traffic control, road safety, and environmental aspects, these extensions are more than welcome. Besides the increase in onboard [4] and RSU [5] compute resources, closer interactions between vehicles and the cloud [4] are also expected in the not-so-distant future. These paired with low-latency, high-throughput network connections can enable sophisticated driving assist or autonomous driving functionalities.

This progress marks out applications and platforms that will provide features for efficiently handling the innate hierarchy of the infrastructure resources that range from onboard devices through edge or RSU equipment up to centralized cloud data centers as a continuum. To leverage such environments for providing advanced vehicular support functions, we identify the following goals to accomplish. First, our solution has to be prepared for high-mobility vehicular environments with moving vehicular and stationary roadside units. Each unit in this hierarchy can pose different characteristics in energy consumption, computation power, and latency, among which we need to find the best balance. Although (beyond) 5G networks aim to support high throughput with low latency, the amount of data transmitted via the wireless link is still crucial and needs to be minimized. Thus, our second goal is to take into account latency and resource footprint aspects as well. Our solution needs to perform data acquisition, aggregation, and processing on the most adequate hierarchy level and provide options for sharing and exchanging source and derived data quickly and efficiently, considering data locality. The frequent updates on road conditions and traffic participants' behavior can be exploited using AI applications that support human drivers in the form of assisted driving or be part of the control loop in autonomous driving. Thus, our final goal is to support such scenarios in anticipation of AI applications.

In our view, these goals can be fulfilled by leveraging federated learning (FL) [6], content caching, and component

orchestration mechanisms. FL is a machine learning setting where agents collaboratively train a model. Low-level entities use their data for local model training and merge them under the supervision of a high-level entity that determines the initial model, learning period, exchange and merge operations as well as redistribution of the merged models. In our opinion, this concept can be translated to a multi-tiered infrastructure where vehicle nodes can supply data, edge/RSUs can learn localized data while merges can happen at higher levels. As in FL only model weights are exchanged, we can take advantage of FL's privacy-preserving and data traffic reduction features, enhancing the latter with caching mechanisms that help to distribute the acquired knowledge efficiently.

Consequently, our contributions are threefold. First, for the upstream direction, we showcase an FL-based subsystem for gathering and aggregating data coming from vehicles. Second, for the downstream direction, we specify an ephemeral content caching framework. Third, we evaluate our solution highlighting its prowess in low-latency information distribution.

In order to discuss these aspects, we give a more detailed use case description in §II. We review related work in §III discussing edge/cloud execution, machine and federated learning platforms in general, and from a vehicular-centered perspective too. Later, we describe our system model in §IV, detailing the FL-based upstream in §IV-A and our downstream caching framework in §IV-B. We highlight important aspects of our testbed, the used software stack in §V while emphasizing the details of our test scenarios and their evaluation. Finally, in §VI, we summarize our work and draw conclusions.

II. Use Case Description

While a plethora of sensors is mounted on Autonomous Vehicles (AV) constantly monitoring the surrounding environment, the limited sensor range and always-present inaccuracies pose many challenges for fully realized real-time autonomous navigation, especially in dense urban settings. Digital High Definition (HD) maps providing high-fidelity, centimeter-level environmental data can be crucial enablers for self-driving vehicles, helping them perceive the precise localization [7] and surrounding environment beyond their sensing capabilities, apply context awareness of their environment, and process local road rules to make safer decisions and plan proactively.

HD maps, mainly built for self-driving purposes, have a high level of accuracy as vehicles need precise localization and environment data to maneuver in real-time. Initially, HD maps are created using special vehicles outfitted with high-precision sensor equipment, like Differential GPS, a multitude of cameras, and highly accurate laser scanners to collect the present obstacles and the traffic rules which apply to the surrounding environment. As depicted in Fig. 1, HD maps can contain multiple layers registering data that changes on different time scales. The static layer can contain infrequently changed information—e.g., a road map—and the transient static layer can show conditions that remain unchanged for an extended period of time—i.e., road work. The transient dynamic layer can provide information that changes frequently,

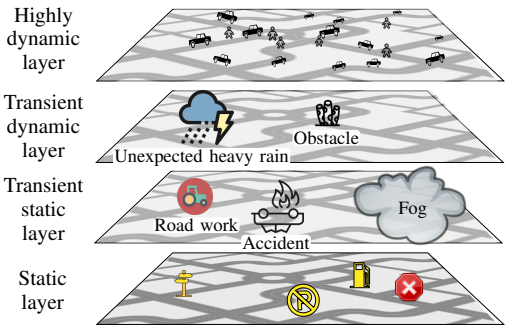

Fig. 1: HD map serving static and dynamic information.

and highly dynamic layers can feed the real-time data of the surrounding environment to the vehicle—such as vulnerable road users. Since traffic is in constant flux, e.g, due to traffic jams, construction works, accidents, or the current status of adaptive traffic signs, the map has to be updated continuously to provide the vehicles with up-to-date information. It is quintessential to receive an HD map update in real-time to facilitate autonomous driving. Edge computing presents a solution to significantly reduce network latency in this regard.

HD maps are specific to location and vehicles in the same geographic area request identical HD map data for autonomous driving. The repetitive transfer of large-volume HD map data through the core network stresses the capacity-constrained backhaul connections. Caching the HD maps on Multi-access Edge Computing (MEC) servers in the vehicular network's edge (e.g., RSUs) can alleviate the backhaul load and substantially reduce the latency, one of the key factors in autonomous driving. Thereby, vehicles can obtain the required HD maps from their RSUs via vehicle-to-infrastructure (V2I) communication without going through the core network. However, the HD map caching problem differs from normal content caching due to frequent changes in data that must be sent to all caching locations which generate periodic traffic on links to RSUs. Existing works target only content caching in vehicular networks or consider only static content and fail to address the dynamic information change specific to HD maps which necessitates new solutions for the HD map caching problem.

III. Related Work

A. Edge-focused Kubernetes platforms

Targeting the edge cloud calls for a suitable platform. K3s and MicroK8s are streamlined, edge-centered Kubernetes (container orchestration) distributions providing better performance in such scenarios than the full-fledged Kubernetes [8]. They ease the deployment process and reduce resource footprint to as low as ~ 0.5 GB of memory and 1 CPU core. According to our own observations, K3s is easier to configure and comes with slightly lower resource demands. Microbenchmarks with real-world workloads applicable to edge computing scenarios testing CPU and memory utilization, I/O, and networking performance show a minimal difference between the two edge distributions [8]. In cases where cold

startup latency, disk throughput, CPU-bound tasks (e.g., matrix multiplication or solving linear equations) are tested, they clearly outperform the traditional Kubernetes. However, the full Kubernetes system still has the edge in cases where a heavy load is managed by only a single replica and no scaling is allowed. If scaling is also permitted, then MicroK8s performs the best, although K3s is still close in every metric [8].

B. Federated learning in vehicular networks

At the time of writing this paper, the most used open-source FL frameworks are FATE [9], Flower [10], and Tensorflow Federated [11]. After experimenting with these solutions, we found that none of them are flexible enough architecturally to serve as a basis for our more general, multi-layered V2X solution as they mostly focus on creating machine learning algorithms for FL. The novel Federated Vehicular Network architecture [12] assigns a manager to a certain group of vehicles that acts as a proxy between a worker vehicle and a cloud server. The manager aggregates the worker models and distributes the model updates via the Federated Vehicular Cloud. This work, however, considers only stationary vehicles and hence cannot be applied to our high-mobility scenario.

C. Distributed caching in vehicular networks

Map sharing can imply serious energy consumption needs which are minimized in a study [13] where an RSU serves a vehicle only if the energy required to receive data from the RSU and for basic movement is less than the remaining energy of the vehicle. For such vehicles, the data is divided among all the RSUs in proportion to the received power to provide the service. A different work [14] shows a joint spectrum assignment and power control policy that maximizes the total data rate in the V2X-enabled network used for disseminating the HD map. The authors study the interference effect on data transmission and formulate a model that describes the interference control problem during dissemination. They suggest a cooperative delivery of HD maps through V2I and V2V communications where—on the basis of data volume and infrastructural environment—they divide the HD map into data blocks. Other authors discuss HD map caching in vehicular networks that support autonomous driving when vehicular trajectories and requests are unknown [15]. They use a reward function based on tile request history. A new architecture is also proposed that combines MEC and Software Defined Networking (SDN) to enable HD map-aided autonomous driving [16]. A two-tier server structure is presented with MEC and cloud servers to achieve low resource utilization and network scalability. The applications and services are deployed on the MEC server using Network Function Virtualization (NFV) at the edge. Other authors propose a MEC system framework for HD map applications discussing application mode, functional modules, HD map data distribution workflow, and communication of the autonomous vehicle client and its server [17].

IV. System Modeling

A. Federated learning-based data aggregation and sharing

We minimize the end-to-end latency of the hierarchical federated network illustrated in Fig. 2 by choosing an optimal number of intermediate layers and nodes per intermediate aggregation node. Every edge device sends its parameters to the corresponding $I_{m,k}$ intermediate node after a given number of iterations as if it were a Poisson process assuming that the intermediate node's processing time follows an exponential distribution. These are typical selections in settings where only latency (and other specific uncertain factors) can be accounted for when describing service time in a model which holds true in our case. In our model, every $E_{I_{M,k},i}$ edge node has to wait for every $E_{I_{M,k},j}$ under the same intermediate node and every node has the ability to send its parameters only after a set number of iterations. Consequently, the distribution of necessary arrivals before the first parameter aggregation start can be modeled with a product of Erlang distributions:

$$f_{I_{m,k}}(x; n, \lambda) = \frac{1}{(n-1)!} \prod_{i=1}^{L} \lambda_i^n x^{n-1} e^{-\lambda_i x} \tag{1}$$

where L is the number of edge devices under the $I_{m,k}$ intermediate node, λ_i is the device-specific rate for calculations, and n is the number of iterations completed before sending parameters upstream. After finishing the n iterations, an edge node has to wait to receive the aggregated data which can lead to significant idle times. The mean waiting time at the edge nodes can be calculated as follows:

$$E_{I_{m,k}}(f_{I_{m,k}}) = n \sum_{i=1}^{L} \frac{1}{\lambda_i} \tag{2}$$

Increasing the number of iterations not only saves both up- and downlink bandwidth but significantly decreases the idle time of the edge devices. Since uplink traffic only occurs from the edge devices to the aggregation nodes at every n iterations, the waiting time is cut down from $\left(\frac{1}{\mu_{I_{m,k}}} + 2T\right) n$ to $\frac{1}{\mu_{I_{m,k}}} + 2T$, where μ is the processing rate, and T is the

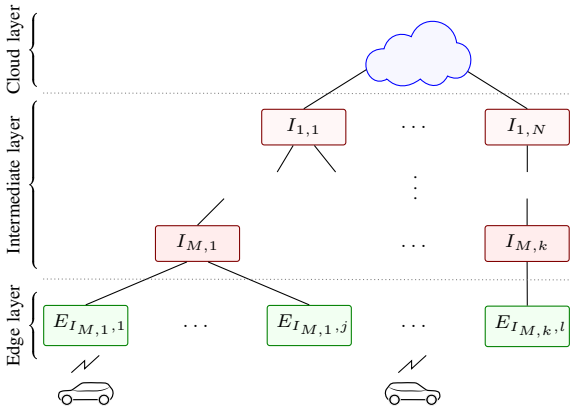

Fig. 2: Multilayered federated learning architecture.

transfer latency. Thus, branches under an aggregation node are idle n times less in n iterations.

A global iteration aggregates $nMKL$ locally trained models. Thus, the total end-to-end latency for one global iteration in the whole network can be calculated as:

$$M\left(\frac{1}{\mu}+2T\right)+n\sum_{m=1}^{M}\sum_{k=1}^{K}\sum_{l=1}^{L}\frac{1}{\lambda_{I_{m,k},l}} \tag{3}$$

The end-to-end latency can be further decreased by optimizing the physical layout and coordinating up/downlink parameter transfers. In large-scale networks, the intermediate aggregation nodes should be positioned strategically to maximize the benefit gained from having multiple layers. This optimal positioning should take into account variables such as predicted edge device density and distance from the parent node in the previous layer. The highest optimization gain for such a network can be attained from coordinating the downlink parameter flow by using a caching mechanism akin to the one presented in §IV-B. Since most V2X machine learning applications only require a fraction of the global data, clients can operate on localized parameter sets. This approach results in a significantly reduced waiting time because clients do not have to wait for the whole network, only for their localized intermediate aggregation node. With this improvement, the total waiting time of each client decreases to:

$$\frac{1}{\mu}+2T+n\sum_{l=1}^{L}\frac{1}{\lambda_{I_{m,k},l}} \tag{4}$$

Although connection loss could thwart such timely aggregations, for our model to hold true, we only need to assure that data from a previous iteration cannot be merged in the current FL model. Adding timestamps to each iteration (at a lower level) and verifying them during merges can achieve this.

B. Information distribution and caching for moving clients

Let an HD map divide a geographical area in $G=\{g_1,\ldots,g_j\}$ tiles which clients stitch together to form a seamless view. Each cache-enabled RSU provides coverage to multiple tiles, updates, and transmits the HD map. As per §II, the HD map has $l_{1,\ldots,4}$ layers in each tile: l_1 contains static information (e.g., roads) which changes over a long period of time, and l_4 contains highly dynamic information (e.g., LIDAR environment information) changing frequently. Let $A_1>\ldots>A_4$ be the respective refresh times of these layers and $S_{g,l}$ be the size of the HD map of tile g layer l. The MEC server on each RSU has a caching capacity of C with variable $c^j_{g,l}$ representing when a layer l of tile g is cached at RSU j (see Table I for notations). The path of vehicle i is represented by $P^i_g,\ \forall g\in G$, where a value of 1 represents if tile g is present on the path of vehicle i. The J_g travel time to cross tile g and the B_g bandwidth of each vehicle in tile g depends on the vehicle density in the RSU. Thus, $D^j_{g,l}=S_{g,l}/B_g$ gives the time required to download layer l of tile g from RSU j and variable $d^{i,j}_{g,l}$ is set to 1 when vehicle i downloads layer l of tile g from RSU j. We assume vehicles know their destinations before starting a journey and that paths are also known and shared with RSUs as vehicles travel. An RSU has the updated HD map of a fraction of the tiles cached in its coverage area and possibly some tiles from neighboring RSUs all the time. If a requested tile is not cached on an RSU, it can fetch it from the cloud or the RSU covering the tile. Thus, we formulate the HD map content caching and tile downloading problem as the following integer linear program with the objective of minimizing the download time under the discussed constraints:

TABLE I: Notation used in the caching problem formulation.

Symbol	Description
$c^j_{g,l}$	1, if layer l of tile g is cached at RSU j, else 0.
P^i_g	1, if tile g is on the path of vehicle i, else 0.
$d^{i,j}_{g,l}$	1, if vehicle i downloads layer l of tile g from RSU j, 0, otherwise.
$D^j_{g,l}$	Time spent on downloading layer l of tile g from RSU j.
$S^i_{g,l}$	Size of HD map layer l of tile g.
C_g	Cache size of RSU g.
J_g	Time taken by a vehicle to cross tile g.
A_l	Update interval of layer l of the HD map tiles.
B_g	Bandwidth provided by the RSU to each vehicle in tile g.

$$\begin{aligned}
&\min \textstyle\sum_{i\in V}\sum_{j\in G}\sum_{l\in L}\sum_{g\in G} c^j_{g,l}P^i_g d^{i,j}_{g,l}D^j_{g,l}\\
&\text{Subject to:}\\
&(C_1)\colon \sum_{l\in L}\sum_{g\in G} c^j_{g,l}S^i_{g,l}\le C_g\\
&(C_2)\colon d^{i,j}_{g,l}\le P^i_g\\
&(C_3)\colon d^{i,j}_{g,l}J_g\le A_l\\
&(C_4)\colon c^j_{g,l}J_g\le A_l\\
&(C_5)\colon d^{i,j}_{g,l}\le c^j_{g,l}\\
&\forall\ i\in V,\ \forall\ j\in G,\ \forall\ g\in G,\ \forall\ l\in L\\
&c,d,P\in\{0,1\}
\end{aligned} \tag{5}$$

Constraint (C_1) ensures that the sum of HD maps stored in cache at RSU j cannot exceed its cache capacity C. (C_2) enforces that the vehicle downloads only the tiles that are present on its path. (C_3) makes sure that the download time does not exceed the update interval of the tile's layer, and (C_4) ensures that if a tile is cached on an RSU then the travel time from this RSU to the cached tile does not exceed the update interval of the tile. Finally, according to (C_5), the vehicle downloads only the cached tiles from an RSU.

V. Evaluation

To verify our theoretical results, we utilized an on-premises cluster of OpenStack-managed virtual machines, each having 8 vCPUs and 16 GB of memory with 950 Mb/s links providing $\sim$1 ms latency. We used K3s as the orchestration layer on our edge nodes due to its ease of deployment, low base resource footprint, and seamless container orchestration capabilities on the edge, as discussed in §III. Our custom-written FL framework sits on top of these layers aiding us to flexibly fine-tune specific parameters. As FL's transparent but crucial

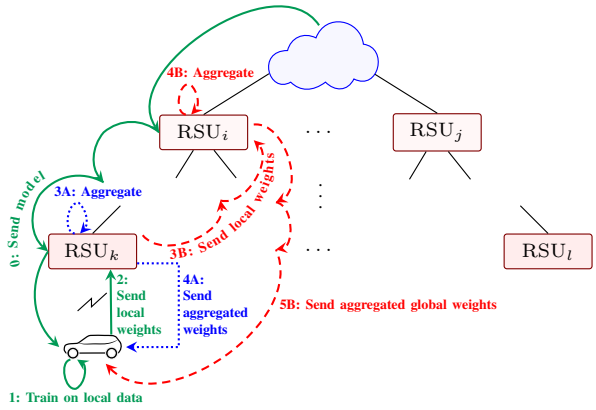

Fig. 3: Flow of one upstream iteration with initialization.

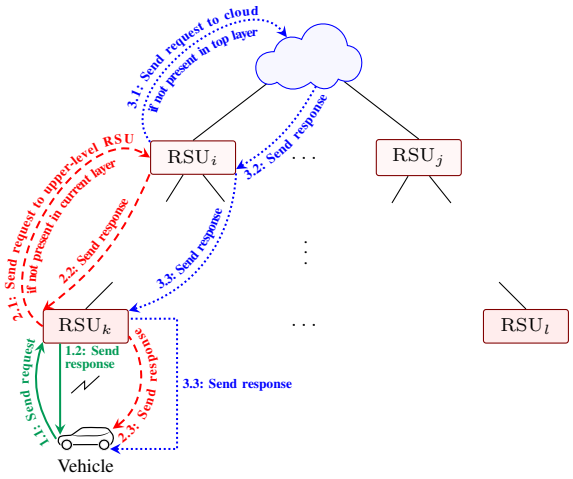

Fig. 4: System flow of the HD map caching.

communication layer, we used the open-source, low-footprint, high-speed, asynchronous ZeroMQ (ZMQ) [18] messaging library that operates without a central broker entity making it ideal for edge scenarios with scarce resources.

A. Upstream: aggregation with federated learning

Our proof-of-concept upstream follows a fully synchronized communication pattern and a centralized FL architecture, uniquely having multiple intermediate layers (contrary to [19]). We define three distinct peer groups: *i*) clients (autonomous vehicles) that are connected to *ii*) a server (or multiple if high availability is essential) via *iii*) RSUs forming a distribution tree with an arbitrarily high number of hierarchy levels, each deployed as containers on our edge infrastructure. As shown in Fig. 3, during initialization, the server distributes the model to be trained down this tree to the clients (step 0). After training the model on locally available data at each client (step 1), model weights are pushed upstream to the closest directly connected RSU (step 2). Based on the configuration and current iteration number, RSUs can *i*) run aggregation locally and return weights following the blue path (steps 3A–4A), within their local training epoch (LTE) or *ii*) forward the weights in the next iteration further up the tree to delegate aggregation following the red path (steps 3B–5B). Each RSU in the tree operates in this fashion, ensuring that end-to-end latency stays low. Consider a tree with an upper RSU_1 having an LTE of 2 and a lower RSU_2 having an LTE of 3. Here, RSU_2 sends weights to RSU_1 on every third, while the server performs global aggregation on every sixth iteration. Iterations last until a preset loss is reached or if some other constraint is met. Note that this aggregation scheme can be generalized to fit other application domains as well.

B. Downstream: HD map caching

In the downstream HD map caching, vehicular clients have predefined random travel paths that lead over various tiles in the coverage of different RSUs. A fraction of the tiles within the RSU coverage is cached locally, and we assume that the latest versions of these tile layers are available at the respective RSU. Upon connection, a client requests the tile corresponding to its position from the RSU that covers the geographic area, as depicted in step 1.1 in Fig. 4. If the requested tile is cached at the RSU, it directly replies with the cached tile layers (step 1.2). Otherwise, it requests the RSU above it in the hierarchy (step 2.1). This happens recursively until either the tile is found or the cloud server is reached (step 3.1). The delay for each layer is added accordingly. We consider each lowest level RSU to handle 8 tiles and $i = 4$ layers for each tile, l_1 being the most static and l_4 the most dynamic. Layer l_i has a size of $10i$ MB and a cache validity of $10(5 - i)$ ms to emulate this dynamicity.

C. Evaluation of the federated learning aggregation delay

In our emulated scenarios, we investigate the effects of various parameters of our mathematical model, discussed in §IV, but leave the detailed discussion of the caching aspect for future work. We examine client training (arrival) time of exponential (our model's assumption), normal, and uniform distributions. In our experiments, we consider 2–4 layers. With 2 layers, no intermediate RSUs are deployed, and all the clients connect directly to the server. In the 3-layer scenario, 2 RSUs are present in the single intermediate layer, and clients are equally split between them. In the last scenario, 2 upper-layer RSUs connect to 4 lower-level RSUs in the intermediate layer, and clients are spread equally among them. To speed up execution, we emulate the learning process, which does not affect the applicability of our concept. We also take snapshots modeling maximal load that makes our results applicable for cases with mobility models.

Fig. 5a–c depict the minimum, mean, and maximum delay observed by the clients for different arrival time distributions broken down according to the number of intermediate layers. Results show that the distribution of the arrival time does not have a significant impact on the mean delay observed by the clients; however, the variance is higher in the exponential

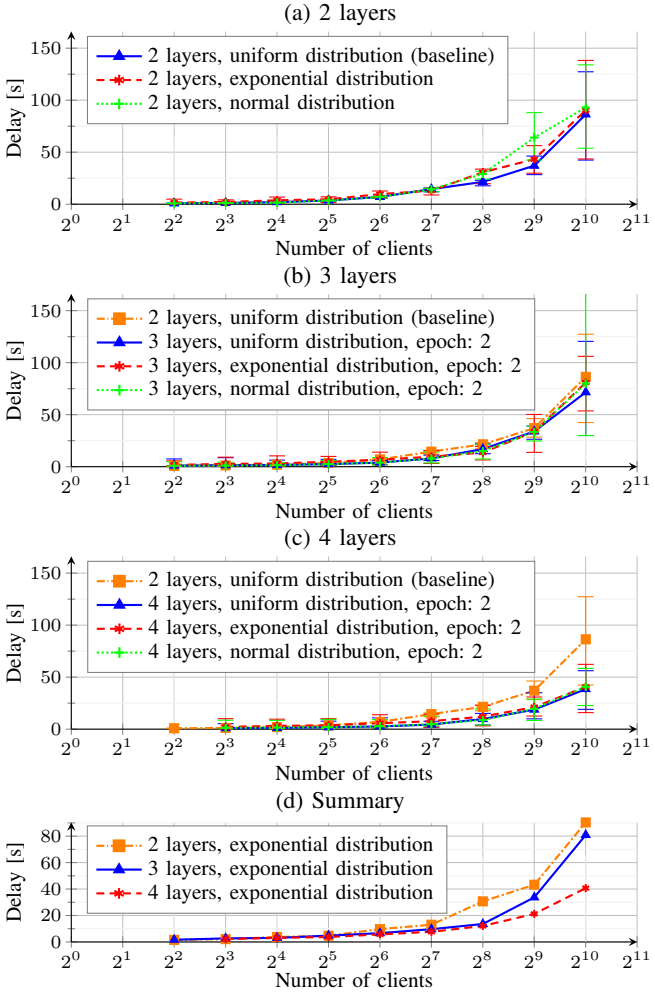

Fig. 5: Measured delay using different number of layers.

distribution's case. Fig. 5d compares the mean delay of every layer configuration in this distribution's case. According to the results, the first significant increase in delay happens at 2^8 clients per RSU, thus adding hierarchy levels can increase the number of clients that the system can serve. Indeed, this delay increase appears at 2^9 clients in the 3-layer scenario (2 RSUs handling 2^8 clients each) and at 2^{10} clients in the 4-layer scenario (2^2 RSUs handling 2^8 clients each). This corroborates our model's theory that we can optimize the delay by adjusting the number of intermediate layers.

Fig. 6 shows the delay when varying the value of the epoch for 32 and 64 clients in the 3-layer scenario, which is the simplest incorporating a cloud layer (and a single intermediate layer). We see a slight reduction in delay in both cases when the epoch is 8, which does not indicate a concrete relationship between the epoch value and the observed delay.

VI. Conclusion

In this work, we showcased a collaborative construction of dynamic HD maps for vehicular use cases. To decrease end-to-end latency and communication footprint, we investigated

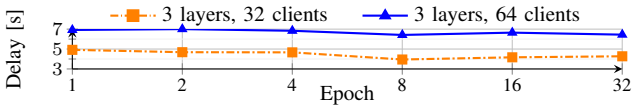

Fig. 6: Delay comparison for different epoch values.

the performance gains of a hierarchical federated learning approach stacking intermediate layers for fast aggregations of local models. We also studied the advantages of content caching for delivering HD maps to clients that do not take part in the learning process or are in need of information that is not local to them. We provided theoretical and experimental analysis and concluded that our proposed techniques can significantly reduce the delay of delivering relevant information to end-users. Indeed, our FL scheme managed to proportionally increase the number of served users by increasing the number of layers while keeping the serving delay steady.

References

[1] Amazon Web Services, Inc, "5G Edge Computing Infrastructure - AWS Wavelength." https://aws.amazon.com/wavelength/, 2023. Accessed: 2023-01-06.

[2] Y. Khalidi, "Microsoft partners with the industry to unlock new 5G scenarios with Azure Edge Zones." https://azure.microsoft.com/hu-hu/blog/microsoft-partners-with-the-industry-to-unlock-new-5g-scenarios-with-azure-edge-zones/, 3 2020. Accessed: 2023-01-06.

[3] A. Phadke, "Bringing partner applications to the edge with Google Cloud." https://cloud.google.com/blog/topics/anthos/anthos-for-telecom-puts-google-cloud-partners-apps-at-the-edge, 12 2020. Accessed: 2023-01-06.

[4] Continental AG, "Continental Continues to Drive Forward the Development of Server-based Vehicle Architectures." https://www.continental.com/en/press/press-releases/20210728-cross-domain-hpc/, 2021. Accessed: 2023-01-06.

[5] Marquez-Barja *et al.*, "Smart Highway: ITS-G5 and C2VX based testbed for vehicular communications in real environments enhanced by edge/cloud technologies," in *EuCNC*, p. 2, IEEE, 2019.

[6] B. McMahan *et al.*, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *AISTATS*, PMLR, 2017.

[7] R. Liu *et al.*, "High definition map for automated driving: Overview and analysis," *J. Navig.*, vol. 73, no. 2, pp. 324–341, 2020.

[8] V. Kjorveziroski *et al.*, "Kubernetes distributions for the edge: Serverless performance evaluation," *J. Supercomput.*, vol. 78, no. 11, 2022.

[9] Y. Liu *et al.*, "FATE: An industrial grade platform for collaborative learning with data protection," *J. Mach. Learn. Res.*, vol. 22, jan 2021.

[10] D. J. Beutel *et al.*, "Flower: A friendly federated learning research framework," *arXiv preprint*, 2020.

[11] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[12] J. Posner *et al.*, "Federated learning in vehicular networks: Opportunities and solutions," *IEEE Network*, vol. PP, pp. 1–8, 02 2021.

[13] J. Xie *et al.*, "An energy-efficient high definition map data distribution mechanism for autonomous driving," *arXiv preprint*, 2020.

[14] X. Wu *et al.*, "A Cooperated Approach Between V2I and V2V for High Definition Map Dissemination in Automated Driving," in *GASS of the International URSI*, IEEE, 2020.

[15] X. Xu *et al.*, "Distributed online caching for high-definition maps in autonomous driving systems," *IEEE WCL*, vol. 10, no. 7, 2021.

[16] H. Peng *et al.*, "SDN-based resource management for autonomous vehicular networks: A multi-access edge computing approach," *IEEE Wirel*, vol. 26, no. 4, 2019.

[17] R. Zhang *et al.*, "The application of edge computing in high-definition maps distribution," in *ACM WSSE*, 2020.

[18] P. Hintjens, "ZeroMQ Guide." https://zguide.zeromq.org/docs/preface/, 2023. Accessed: 2023-01-06.

[19] L. Liu *et al.*, "Client-edge-cloud hierarchical federated learning," in *IEEE ICC*, 2020.