KAFKAFED: Two-Tier Federated Learning Communication Architecture for Internet of Vehicles

Saira Bano*,†, Nicola Tonellotto* Pietro Cassarà†, Alberto Gotta†
*Department of Information Engineering, University of Pisa, Pisa, Italy
†Information Science and Technology Institute "A. Faedo", National Research Council, Pisa, Italy
saira.bano@phd.unipi.it, nicola.tonellotto@unipi.it
pietro.cassara@isti.cnr.it, alberto.gotta@isti.cnr.it

Abstract: In the current era of the Internet of Vehicles (IoV), vehicle to vehicle data sharing can provide customized applications for Connected and Autonomous Vehicles (CAVs). The advancement of Deep Learning (DL) methodologies is one of the key driving forces for CAVs, allowing elaborating a massive amount of data by the resource-constrained onboard devices. In a traditional centralized DL approach, vehicle data are transmitted to the cloud for the training of models. This approach leads to significant communication overhead, high delays, and data privacy concerns. Conversely, Federated Learning (FL) performs the training using the local models in a distributed fashion and mitigates the data privacy risks by sharing only the model parameters with the server, optimizing the FL to be used with resources-constrained devices. In this paper, we propose the design of a scalable communication infrastructure to support the FL procedure based on Information-Centric Networking (ICN) using Apache Kafka, called KAFKAFED. The ICN-based infrastructure allows to overcome the shortcomings of current client-server architectures for FL, in which routing is content-based or name-based to achieve efficient data retrieval for mobile nodes. In ICN, data are stored at intermediate nodes to provide efficient and reliable data delivery. A proof of concept of the KAFKAFED communication architecture is developed and tested in an emulated environment. The performance of the proposed framework compared to the client server-based FL architecture, i.e., FLOWER showed a boost of almost 40% with just 32 clients in addition to several other advantages of scalability, reliability, and security

***Index Terms*—Federated Learning, Apache Kafka, Connected and autonomous vehicles, Publish/Subscribe model**

I. INTRODUCTION

In recent years, the exponential growth of the Internet of Things (IoT) data is being fueled by the rapid adoption of connected devices that are nearly more than 7 billion in the world [1]. Such connected devices include Connected and Autonomous Vehicles (CAVs). For vast numbers of CAVs estimated to be deployed during the next few years, it is vital to consider their behavior, influence on people, and consequences, both based on their own experiences and other customers [2]. These CAVs are aimed to provide many kinds of user applications, including infotainment, safety, traffic flow optimization, and overall efficient use of transportation infrastructure. The performance of most CAVs applications relies heavily on AI-based solutions, especially DL algorithms. Since these applications require advanced sensing and computing capabilities, so in order to provide DL functionalities for these CAVs, which have constrained computational resources, we need to address new research challenges [1], [3]. The most easy way of using IoT data for model training in DL applications consists of sending all the data to a centralized entity, as in the classic cloud-centric approach, and utilize it to produce inference models. However, from the perspective of privacy rights of users, this strategy is unfeasible, because it exposes the local information of the users. Furthermore, transferring big amount of data towards the cloud imposes a burden for both the network and for the on-board devices, particularly in unstructured data applications, such as video analytics [4]. Moreover, this conventional approach causes unacceptable latency in CAV time-sensitive applications. To guarantee user's privacy and to keep the data on user's device, a decentralized approach for training AI models is proposed by McMahan in 2016, called Federated Learning (FL) [5].

In FL, interested clients collaboratively train a model on their local information and exchange their model parameters with the central server to generate a global model shared among all the involved entities [6]. However, to simulate the concept of FL in real-world scenarios is challenging for researchers because they lack the necessary resources to train their federated models on millions of real-world devices, except for a few who work for companies like Google and Facebook. Nevertheless, since the concept has been introduced, there has been a few open-access FL frameworks developed that emulate a federated environment, such as TensorFlow Federated (TFF) [1] introduced by Google, LEAF [7], and FedEval [8]. TFF provide a framework to implement decentralized training, LEAF offers datasets for FL applications, and FedEval provides customized strategy options for communication protocols and aggregation methods for FL. Other frameworks include FedML [9], which provides support for real-world IoT devices by using FedML-Mobile and FedML-IoT, PySyft [10], that offers the so-called remote workers for federated setting with PyTorch compatibility, FATE [2] by WeBank, to ensure secure aggregation in FL, and FLOWER [11], for mobile and edge devices to simulate FL in a scalable manner.

The frameworks discussed above are based on the client-server communication (network-centric) paradigm for updates

This research was supported by TEACHING project funded by the EU Horizon 2020 research and innovation programme under GA n. 871385

[1]https://www.tensorflow.org/federated
[2]https://fate.fedai.org/

between local nodes and the data-fusion server. In this network-centric architecture, mobile users can suffer from disconnections resulting in data unavailability when outside the server's reach. Solutions like in-network caching (essential to make data available) are missing in naive network-centric communication. Information-Centric Networking (ICN) is a novel paradigm, where connectivity may well be intermittent, and data becomes independent from location, application, and means of transportation, enabling in-network caching and replication. The expected benefits are improved efficiency, better scalability concerning information/bandwidth demand, and better robustness in challenging communication scenarios.

Current literature [12], [13] argues that ICN seems to replace classical network-centric communication, but we foresee it as an overlay network running on top of a network-centric based model as the actual content delivery still requires the TCP/IP interface for communication.

In this paper, we propose a novel efficient communication framework called KAFKAFED for FL applications training, based on an information-centric architecture, such as the publish/subscribe implemented in Apache Kafka [14]. Apache Kafka aims to provide high scalability, high data availability, and low latency for mobile nodes. In addition, the paper considers the usefulness of FL over the edge computing network for effective learning by the exchange of model updates between intermediate storage nodes, called Kafka brokers, on the edge, and the vehicles. Thus, low latency in vehicular learning applications can also be achieved. This paper details the development of a complete FL framework using Kafka. Our developed framework is also compared to FLOWER [11], a network-centric FL framework based on Remote Procedure Control (gRPC) protocol for communication. Finally, we evaluate when our information-centric architecture outperforms the network-centric paradigm by considering different vehicular communication scenarios.

The remainder of the paper is organized as follows. Section 2 provides a brief review of the recent advances in FL communication and Apache Kafka. Section 3 presents our proposed methodology. We evaluate the proposed framework in different communication scenarios to understand its performance in Section 4, and finally, Section 5 draws our conclusion.

II. RELATED WORK

As far as this work focuses on the communication paradigm of FL, we cover the literature review related to FL and communication perspectives. Regarding the latter, we are providing the background and related work of the Apache Kafka publish/subscribe paradigm.

A. Federated Learning

FL consists of two main components: a server that determines the structure of model updates and clients that train the model on locally accessible data. The global model is then computed by the server using the parameters provided by all the clients. There are two key factors to consider when deploying the FL in real-world scenarios: computational cost and communication costs. In this paper, we are analyzing the communication costs of FL based on the Pub/Sub mode of communication. To date, several studies have investigated the communication costs for FL. In [15], the authors proposed an adapting FedAvg scheme by using Adam optimization and compression schemes for communication efficient FedAvg (CE-FedAvg) to reduce the number of rounds needed for global model convergence at the cloud and show more robustness to aggressive communication reduction. In [16], FL framework using quorum blockchain technology and Apache Kafka has been developed for the secure communication of model. In [17], authors presented the Federated Learning with Quantization (FLQ) framework, to reduce the exchange of data between the cloud and edge and in the opposite direction for efficient communication. Chen et al. [18] offer a communication-efficient FL technique based on a layer-wise asynchronous updating strategy that takes into account both client and server operations for reducing overall communication cost. The main intention of KAFKAFED is to provide a communication efficient framework that can also be beneficial for the users that are not static.

B. Apache Kafka

Apache Kafka is an open-source distributed messaging framework used to process data streams [14]. In Apache Kafka producers (vehicles in our case) produce the messages and store them over the broker in the form of topics, while consumers who subscribe to these topics can consume these messages from the broker. Every topic can have many partitions that are copied across the Kafka brokers to provide fault tolerance capabilities of the Kafka cluster. Kafka consumers read messages in the same order to which they are appended to partitions, as messages are written to partitions in an append-only method. Another design choice that makes Kafka very scalable and suitable for distributed systems, is that it can exploit multiple brokers instances. To simplify the coordination of the instances, Kafka employs ZooKeeper in order to (i) detect the addition and the removal of both brokers and consumers, (ii) maintain the relationship among brokers and consumers, (iii) trigger a rebalancing of the workloads when either brokers or consumers are added or removed. In contrast to Kafka, there are also some information-centric communication models available such as RabbitMQ, ActiveMQ, and MQTT [14]. However, Kafka outperforms in terms of messages sent per second by both data producers and consumers, [19]. As a result, Kafka appears to be a better fit for CAVs applications that require high throughput.

There is a relatively small amount of literature that exists for deep learning applications with ICN and then Kafka. Authors in [20] developed an efficient and low-latency distributed message delivery system for Connected Vehicle (CV) applications that enables a data-focused view of the entire CV ecosystem. Instead of doing a training on static data, researchers in [21] proposed the scheme of training and inference of machine learning models on continuous data streams using Apache Kafka. Feraudo et al. [22] suggested an asynchronous participation mechanism for IoT devices in FL model training based on a publish/subscribe architecture, in which each participant publishes its intention to participate in the round for FL model training. In [23], authors proposed an edge-enabled cloud-assisted system for distributed intelligence covering advanced ML algorithms by collecting all the data at the edge over the brokers using Apache Kafka and updating the global model using reinforcement learning.

In contrast to earlier findings, this is the first study in which Apache Kafka has been integrated into the FL for the exchange of model updates between clients and the FL server.

1) Comparison studies of Pub/Sub with Request-Response: There are few studies available in which researchers compare network-centric (request-response) models with information-centric (pub/sub) models. In [24], authors compute the cost model of pub/sub with the polling and client-server architecture and claimed that publish/subscribe system is well suited for a distributed real-time system. In [25], authors compare pub/sub with request-response model based on different parameters such as mobility support, adaptability, timeliness, and efficiency. They also proposed a communication model based on the integration of both systems. Eugster et al. provide a comparison between the traditional request-response scheme with pub/sub scheme based on decoupling in three different dimensions i.e space, time, and synchronization [26].

III. Proposed Methodology

In this paper, we propose a communication framework for improving the efficiency of an FL-based procedure for connected autonomous vehicles, assuming that some components in the infrastructure can be resource constrained. In traditional communication, a high level of accuracy can be achieved by transferring all the data from the vehicles to the central server in the cloud or to the Road Side Unit (RSU). This approach is not feasible in dynamic vehicular communication because it requires a large amount of bandwidth. The target of our optimization is to reduce this communication overhead. This optimization can be achieved by designing an FL-based procedure that involves the optimized transmission of the local model parameters. Note that the FL solutions described in Section I are based on a client-server based communication paradigm; instead, in the literature, such as in [27], it is shown that information-centric networking based on the publish/subscribe paradigm performs better than the previous one in terms of communication overhead.

In this work, we present a novel framework KafkaFed for FL based on information-centric architecture by using the Pub/Sub mode of communication provided by Apache Kafka. In KafkaFed, the interaction between vehicles and the central server (Cloud or RSU) for model updates is achieved via Kafka brokers forming a two-hop communication infrastructure. In this communication scenario, the central server is devoted to aggregate the models received from the vehicles. Figure 1 depicts the overall structure of KafkaFed and shows the interaction between FL server and vehicular clients using Kafka broker.

In this work, we assume that Kafka broker runs on the edge, the FL server is running on the cloud, and each vehicle trains its model locally. The core of our system is the Kafka broker, which stores the models' parameters coming from the clients and acts as an "orchestrator" between the FL server and the vehicles. The configuration service for Kafka is achieved using the tool ZooKeeper, which is a centralized service for maintaining configuration information, providing distributed synchronization naming and group services; Kafka broker and Zookeeper are setups on the same node of the adopted communication infrastructure.

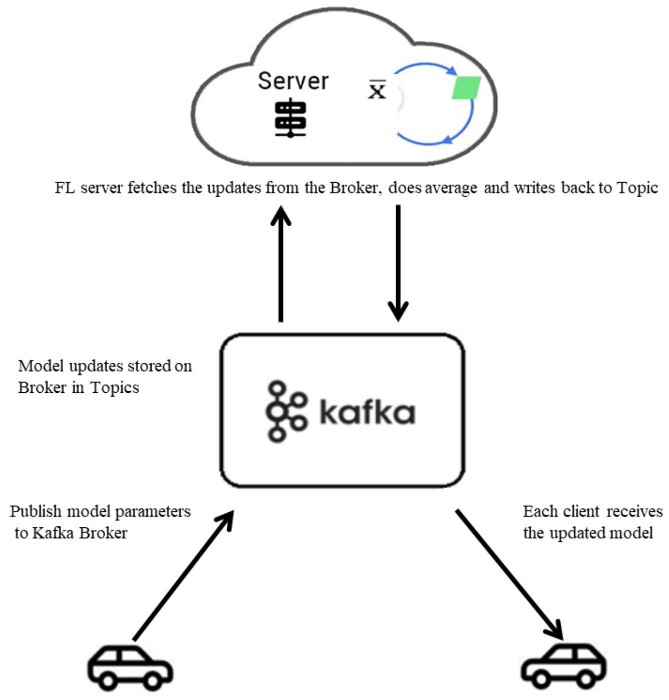
Server
$\overline{x}$
FL server fetches the updates from the Broker, does average and writes back to Topic
Model updates stored on Broker in Topics
kafka
Publish model parameters to Kafka Broker
Each client receives the updated model

Figure 1: Overview Of Proposed Framework

On the Kafka broker, we defined the "topics" that are the logical channels separating messages coming from the uplink and downlink flows. These messages are the model updates to and from the server and clients in the FL process. The topic name is the key value used by the client for subscribing to the broker for receiving and transmitting data in Kafka. We have used a total of 2 topics *clients_data* and *averaged_result*. Topic *clients_data* is used to receive the model updates from the clients in the uplink direction and on *averaged_result*, the server publishes its global model update. We also applied *gzip* compression over both topics so that we can compress model updates shared between vehicles and server.

In the federation process, each vehicle trains its local model for several epochs and acts as a producer to send its local model parameters at a given data rate towards the broker. The server aggregates the data by fetching the model updates of all the clients from topic *clients_data*. The whole model update for each client is divided into small batches of data sizes equal to 10 KB before sending to the broker. Each message packet within each batch contain information about the source of origin, i.e. each client id so that we can retrieve the messages of each client at the server for the aggregation of the model. The model updates are divided into small batches because DL algorithms consist of millions of parameters, and it is not feasible to send all the parameters in a single batch. After dividing the model into batches, Kafka producer API pushes these messages over the broker (edge) and append them within the specified topic. The FL server runs the Kafka consumer API and fetches the model updates of all the clients for averaging. After averaging, the FL server also divides the averaged model into batches of 10 KB and sends it back to the clients by publishing over the broker on the topic *averaged_result*. Each client reads the averaged

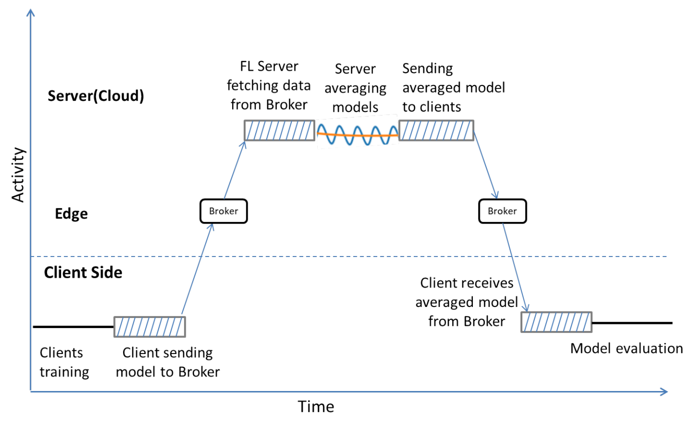
Server(Cloud)
FL Server fetching data from Broker
Server averaging models
Sending averaged model to clients
Activity
Edge
Broker
Broker
Client Side
Client receives averaged model from Broker
Clients training
Client sending model to Broker
Model evaluation
Time

Figure 2: KAFKAFED for one round of FL

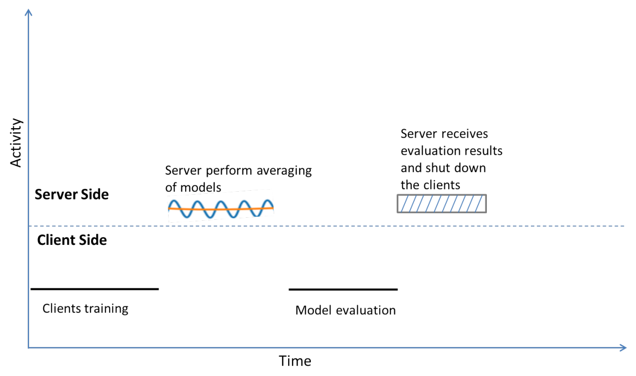
Activity
Server receives evaluation results and shut down the clients
Server perform averaging of models
Server Side
Client Side
Clients training
Model evaluation
Time

Figure 3: FLOWER for one round of FL

model parameters from the *averaged_result* topic and evaluates this averaged model, thus completing one round of FL. Once obtained the desired global model, the server optimizes resource utilization by suspending the global model updating.

IV. METHODOLOGY VALIDATION

A. Experimental Environment

We setup an experimental environment to validate the suggested approach and its hypothesis in reaching a comparable performance as the client-centric based communication in the federated network. For the experimentation, we employ a three-layer neural network for MNIST classification, with the first two layers having 200 neurons and the third layer having 10 neurons. In order to make a fair comparison between both architectures, we use the same parameters. For example size of the training data, batch size, learning rate and the number of epochs are 5000, 128, 0.001 and 5 respectively. For testing the two frameworks, we consider different numbers of clients: 2, 4, 8, 16, 24, and 32 clients each using 5000 images local training subset from MNIST dataset.

1) KAFKAFED: In our experiments, the Kafka system is based on the latest version 3.0.0, orchestrated by Zookeeper 3.6.3, and both are running on a Jetson Xavier NX edge device on which the FL server is also running. This edge device has 64 GB of memory, on which the Debian operating system is installed. For emulation, we established a Kafka cluster with only one broker. We used two PCs on which we run FL clients. We considered one PC as a dense server on which we run a large number of clients, while for calculating values of one complete round of FL we used one client as a probe. In general, each system has a Debian distribution installed with Python 3.8 and all supporting libraries such as Kafka Python, PyTorch, and others.

2) FLOWER: The configuration setup described above is also used to emulate the framework discussed in this section. We wrapped the same model and parameters within FLOWER architecture and change the *Strategy* of FLOWER server so that before a training cycle of FL can begin, a minimum number of clients must be connected to the server. We use four different machines to divide the load among them in such a way that the training time of each client would remain the same, as the training time of each client is dependent on the device's capabilities as well as the number of clients running on each processor. The FLOWER server is running on the Nvidia Jetson Xavier NX device. Figure 3 depicts the implementation process of FLOWER architecture.

3) Dataset Distribution: During the federated procedure, clients train local models on heterogeneous datasets. Indeed, the datasets may be distributed differently among clients, which can have an impact on the training duration and accuracy of trained models. It is necessary while comparing different techniques for a federated solution that the training time for each client in both scenarios should be the same. For this reason, we assigned the same size of subset 5000 samples to each client. We use the MNIST dataset [3], which is a collection of 70k handwritten digits and formed images of size 28x28.

B. Results

During experimentation, we studied the time for one complete round of FL in two different considered scenarios by changing the number of clients, as depicted in Table II and III. The two scenarios that we considered for the vehicular use case are as follows:

- IEEE 802.11p for vehicular communication with a bandwidth of 27 MB/s and 50 ms delay in sending each packet
- 5G for an automotive use case with a bandwidth of 100 MB/s and 1 ms of delays

We emulated the above-mentioned scenarios, using a NetEm [28] tool over Ethernet, a functionality provided by LINUX traffic control to introduce packet delays, jitters, losses, and bandwidth throttling. All the experiments are repeated five times. In real-world scenarios, clients have heterogeneity in the network speed as they are distributed in different regions. However, for the sake of experimentation, we are considering the fixed upload speed.

1) Comparing KAFKAFED and FLOWER: We compare our information-centric framework KAFKAFED with the network-centric based architecture FLOWER. A schematic comparison

[3]http://yann.lecun.com/exdb/mnist/

Table I: Comparison of KAFKAFED and FLOWER

Comparison	KAFKAFED	FLOWER
Heterogeneous clients	Decoupling of clients and server via broker provides support for heterogeneous clients	FLOWER server is unaware of the nature of connected clients so provides support for heterogeneous clients
Scalability	We scaled and tested this framework upto 32 clients	We simulate it upto 32 clients, with FLOWER clients running on 4 different machines
ML framework agnostic	Users can leverage their FL tasks by using any Machine Learning framework	Same applies to FLOWER
Language agnostic	Kafka provides support for Python, Java, C# and C/C++ clients	In addtition to the mentioned languages for Kafka it also provides the support for iOS
Compression	Kafka provides gzip, snappy and LZ4 compression	Not available

of both frameworks is depicted in Table I for the considered properties. We are achieving the same features as FLOWER, with the added benefit of applying compression over Kafka topics to exchange fewer data between clients and brokers. The use of KAFKAFED as an information-centric communication paradigm for FL could help to improve data retrieval efficiency in vehicle scenarios.

2) Performance evaluation for IEEE 802.11p use-case: In this scenario, we assume that each vehicle has On-Board Units with IEEE 802.11p omnidirectional antennas of fixed range communicating with a broker over the edge. Each vehicle can subscribe to the role of both producer and consumer while sending model parameters and reading the model updates from the broker respectively. To simulate this behavior we are using the NetEm tool for setting required upload speed and delay as described earlier. The results for both frameworks are shown in Table II. These results show that with the small number of vehicular clients FLOWER is performing better while KAFKAFED time is doubled. However as we increase the number of participating vehicles, we achieved better time for one round of FL in KAFKAFED. While in FLOWER with the increasing number of clients, the time increases. This can be the consequence of the flash crowd situation in which a large number of clients increases the traffic load on a particular server. Contrary to that in the information-centric paradigm decoupling of FL clients with the server helps to minimize the flash crowd.

Table II: For 802.11p use-case, time (seconds) per round of FL with varied numbers of clients

No. of	KAFKAFED			FLOWER		
Clients	Min	Max	Avg	Min	Max	Avg
2	20.01	21.06	**20.42**	10.01	11.08	**10.54**
4	20.38	21.47	**20.92**	12.74	15.70	**14.51**
8	22.14	23.01	**22.45**	19.07	19.50	**19.24**
16	23.90	24.20	**24.02**	24.06	25.15	**24.60**
24	25.19	25.23	**25.27**	33.23	34.00	**33.70**
32	26.82	27.13	**26.93**	43.01	44.01	**43.50**

3) Performance evaluation for 5G use-case: According to a recent 5G Automotive Association white paper, 5G aims to provide a high data rate (100 MB/s) for vehicular connections with high reliability and low latency (1 ms) features. We have analyzed the values for one round of FL using 5G for both used frameworks. With respect to IEEE 802.11p, we can see a significant improvement in a time for one round as shown in Table III. This behavior is because in 802.11p we have a large value of packet delays. However, the general trend for the two frameworks is the same as we see earlier in 802.11p, that is, for a small number of participating devices FLOWER performs better and for a large number of clients, KAFKAFED achieves better performance as compared to FLOWER.

Table III: For 5G use-case, time (seconds) per round of FL with varied numbers of clients

No. of	KAFKAFED			FLOWER		
Clients	Min	Max	Avg	Min	Max	Avg
2	14.98	16.58	**16.42**	10.83	11.21	**11.01**
4	16.09	17.02	**16.56**	14.22	15.01	**14.90**
8	17.67	18.20	**18.00**	16.55	17.24	**17.30**
16	20.01	21.55	**20.70**	21.06	22.46	**22.40**
24	23.64	24.04	**23.99**	27.01	30.40	**28.70**
32	24.97	25.60	**25.48**	41.00	42.01	**41.78**

With the proposed framework, the performance of the KAFKAFED FL server with a various number of clients is shown in Table IV for both 802.11p and 5G. As described earlier in Section IV and shown in Figure 2, FL server in KAFKAFED is responsible to fetch the data from *clients_data* topic, doing the average of all the client parameters, and writing back the result to the *averaged_result* topic. Consequently, as the number of clients increases, it has to fetch more data from the topic. That is why the time of server increases with the number of clients.

Table IV: FL server duration for both considered technologies in KAFKAFED

No. of	KAFKAFED			FLOWER		
Clients	Min	Max	Avg	Min	Max	Avg
2	3.45	5.21	**4.62**	3.69	3.77	**3.72**
4	4.50	5.55	**5.02**	3.93	4.60	**4.26**
8	5.45	5.62	**5.50**	4.96	5.02	**4.98**
16	6.98	7.20	**7.03**	7.22	7.60	**7.23**
24	8.30	8.53	**8.40**	8.28	8.94	**8.83**
32	10.18	10.27	**10.22**	10.01	10.64	**10.50**

The above discussed experimental results show that the performance of the proposed model is better in the network with respect to FLOWER. It is also evident from the results that KAFKAFED is more scalable as it shows the linear trend by increasing the number of clients. Furthermore, the proposed architecture allows the vehicles to remain connected to the broker even when there is disconnection with the server, because the connecting time with the broker over the edge is less than the connecting time with the cloud.

V. Conclusion and future work

In this paper, we have presented a novel communication-efficient framework called KafkaFed for federated learning based on the Pub/Sub model that can be used on Internet of Vehicle applications. This framework is based on an information-centric mode of communication for FL. The publish/subscribe paradigm facilitates flexible and dynamic vehicular network services by providing loosely coupled and scalable communication. While in a network-centric paradigm the server may even become a bottleneck when there are a large number of clients. The paper has highlighted most of the design challenges and provided an overview of the implementation. We evaluate the proposed scheme by comparing it with the existing baselines network-centric model i.e. Flower. Simulation results validate that the proposed framework can achieve better results for FL as compared to network-centric-based architectures. The major benefits of this framework for FL are scalability, decoupling in time (client and server do not need to be active at the same time), decoupling in space, data reliability, and data availability as Kafka is highly available in nature because of its distributed platform. In information-centric architecture, a server serves a few brokers that in return serve a large number of vehicles within the vicinity of the particular edge thus reducing the flash crowd situation over the FL server. As a result, the publish/subscribe paradigm may be the most promising communication model for FL. In the future, we will improve the proposed scheme and analyze its performance in real-world FL applications and datasets for vehicular situations. Furthermore, we will also consider the use of a large Kafka collection of brokers on the edge.

References

[1] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[2] C. E. Andrade, S. D. Byers, V. Gopalakrishnan, E. Halepovic, D. J. Poole, L. K. Tran, and C. T. Volinsky, "Connected cars in cellular network: a measurement study," in *Proceedings of the 2017 Internet Measurement Conference*, 2017, pp. 235–241.

[3] D. Bacciu, S. Akarmazyan, E. Armengaud, M. Bacco, G. Bravos, C. Calandra, E. Carlini, A. Carta, P. Cassarà, M. Coppola *et al.*, "Teaching-trustworthy autonomous cyber-physical applications through human-centred intelligence," in *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)*. IEEE, 2021, pp. 1–6.

[4] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE network*, vol. 32, no. 1, pp. 96–101, 2018.

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

[6] P. Cassarà, A. Gotta, and L. Valerio, "Federated feature selection for cyber-physical systems of systems," *arXiv preprint arXiv:2109.11323*, 2021.

[7] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," 2019.

[8] D. Chai, L. Wang, K. Chen, and Q. Yang, "Fedeval: A benchmark system with a comprehensive evaluation model for federated learning," *CoRR*, vol. abs/2011.09655, 2020. [Online]. Available: https://arxiv.org/abs/2011.09655

[9] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, X. Zhu, J. Wang, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, and S. Avestimehr, "Fedml: A research library and benchmark for federated machine learning," 2020.

[10] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, "A generic framework for privacy preserving deep learning," 2018.

[11] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, P. P. B. de Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," 2021.

[12] S. Arshad, M. A. Azam, M. H. Rehmani, and J. Loo, "Recent advances in information-centric networking-based internet of things (icn-iot)," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2128–2158, 2018.

[13] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE communications surveys & tutorials*, vol. 16, no. 2, pp. 1024–1049, 2013.

[14] J. Kreps, N. Narkhede, J. Rao *et al.*, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, vol. 11, 2011, pp. 1–7.

[15] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in iot," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5986–5994, 2019.

[16] M. R. Behera, R. Otter, S. Shetty *et al.*, "Federated learning using distributed messaging with entitlements for anonymous computation and secure delivery of model," 2020.

[17] N. Tonellotto, A. Gotta, F. M. Nardini, D. Gadler, and F. Silvestri, "Neural network quantization in federated learning at the edge," *Information Sciences*, 2021.

[18] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 10, pp. 4229–4238, 2019.

[19] P. Dobbelaere and K. S. Esmaili, "Kafka versus rabbitmq: A comparative study of two industry reference publish/subscribe implementations: Industry paper," in *Proceedings of the 11th ACM international conference on distributed and event-based systems*, 2017, pp. 227–238.

[20] Y. Du, M. Chowdhury, M. Rahman, K. Dey, A. Apon, A. Luckow, and L. B. Ngo, "A distributed message delivery infrastructure for connected vehicle technology applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 787–801, 2017.

[21] C. Martín, P. Langendoerfer, P. S. Zarrin, M. Díaz, and B. Rubio, "Kafka-ml: connecting the data stream with ml/ai frameworks," *arXiv preprint arXiv:2006.04105*, 2020.

[22] A. Feraudo, P. Yadav, V. Safronov, D. A. Popescu, R. Mortier, S. Wang, P. Bellavista, and J. Crowcroft, "Colearn: Enabling federated learning in mud-compliant iot edge networks," in *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, 2020, pp. 25–30.

[23] P. Bellavista, R. Della Penna, L. Foschini, and D. Scotece, "Machine learning for predictive diagnostics at the edge: an iiot practical example," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.

[24] S. Oh, J.-H. Kim, and G. Fox, "Real-time performance analysis for publish/subscribe systems," *Future Generation Computer Systems*, vol. 26, no. 3, pp. 318–323, 2010.

[25] C. Rodríguez-Domínguez, K. Benghazi, M. Noguera, J. L. Garrido, M. L. Rodríguez, and T. Ruiz-López, "A communication model to integrate the request-response and the publish-subscribe paradigms into ubiquitous systems," *Sensors*, vol. 12, no. 6, pp. 7648–7668, 2012.

[26] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM computing surveys (CSUR)*, vol. 35, no. 2, pp. 114–131, 2003.

[27] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, pp. 1–29, 2019.

[28] S. Hemminger *et al.*, "Network emulation with netem," in *Linux conf au*, vol. 5. Citeseer, 2005, p. 2005.