Accelerating Federated Learning at Programmable User Plane Function via In-Network Aggregation

Chanbin Bae*, Hochan Lee*, Sangheon Pack*, and Youngmin Ji†
*School of Electrical Engineering, Korea University, Seoul, Korea
E-mail: {bin6050, ghcks1000, shpack}@korea.ac.kr
†Korea Electronics Technology Institute, Gyeonggi-do, Korea
E-mail: ym.ji@keti.re.kr

***Abstract*—Recently, 5G mobile networks are evolving with emerging real-time interaction applications such as AR/VR, which require high throughput and low latency. To meet this demand, user plane function (UPF) should support high-speed data plane and protocol extensions with continuously evolving specifications. Therefore, UPF can be offloaded to a programmable data plan (PDP), which supports flexible packet processing and protocol extension. Meanwhile, as machine learning (ML) models have grown in size and privacy concerns have increased, federated learning (FL) was proposed as a distributed manner solution in mobile networks. To improve the performance of FL, PDP can be used to enhance communication efficiency and decrease learning delay by utilizing in-network aggregation (INA). In this context, solutions for accelerating FL at UPF can be implemented on PDP. In this paper, we present AccelFL that is designed to accelerate FL at UPF by aggregating local gradients in networks via INA. Our experimental results demonstrate that AccelFL can reduce job completion time (JCT) and communication overhead by 30% and 36.9%, respectively.**

***Index Terms*—User Plane Function, Federated Learning, In-Network Aggregation.**

I. Introduction

Emerging 5G mobile network applications, such as autonomous driving or AR/VR, require high throughput and low latency. To support these requirements, user plane function (UPF), a data plane of 5G core networks, must perform high-speed data processing. In addition, the UPF functionalities can be written in match-action rules and continuously updated with evolving specification changes by interacting with the control plane. Since programming protocol-independent packet processing (P4), which is the most popular programming data plan (PDP) language, enables flexible packet processing and the update of match-action tables with control plane, a PDP with P4 is a candidate for implementing high-speed data processing UPF [1]. In this context, UPF can be offloaded to a PDP which enables line-rate, flexible, and extensible packet processing [3].

Meanwhile, in traditional machine learning (ML), an ML model is trained on a central server with data from clients. Since the central server should collect data from distributed clients, privacy concerns have arisen. To address this problem, federated learning (FL) has been introduced, which runs a ML model in each distributed device with remaining data on the local devices. This solution allows distributed learning with five steps [8]: 1) FL server trains the initial model, 2) FL server transmits the model to clients, 3) each client trains the model locally using its own data, 4) each client transmits the local gradients to FL server, and 5) FL server aggregates the gradients to train global model and broadcast the aggregated global model. However, transmitting local updates causes high communication costs, which leads to performance degradation.

On the other hand, PDPs are an appropriate place to support ML due to their location in the middle of networks [5] and their ability with flexible packet processing, which enables computing in networks. Leveraging these features of PDPs can accelerate distributed deep learning (DDL) via in-network aggregation (INA) by computing local gradient updates in programmable switches [6], [7]. Furthermore, these schemes can alleviate communication overhead by reducing gradient packets. With these technologies, we can accelerate FL in mobile networks via PDP. Moreover, UPF is a suitable place for implementing FL acceleration schemes because every data packet in mobile networks should pass through UPF. However, to the best of our knowledge, there is no such research that implements integrated INA and UPF functionality together in PDP.

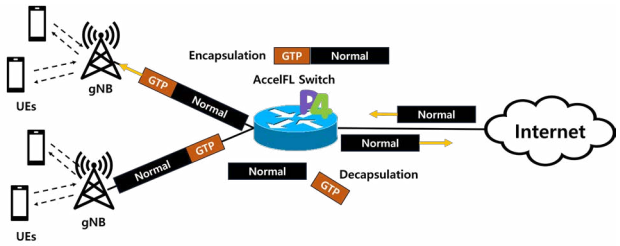
In this paper, we propose AccelFL, which integrates UPF functionality and INA for FL acceleration. AccelFL is implemented on a PDP with UPF and gradient packet aggregation. It can accelerate the training time for FL and reduce the training traffic. Evaluation results show that AccelFL can accelerate learning models in terms of job completion time (JCT) performance and communication efficiency compared to traditional FL.

The remainder of this paper is organized as follows. We describe the architecture and design of our proposed scheme, AccelFL, in Section II and Section III. Then, we show the performance evaluation of AccelFL in terms of JCT and communication overhead in Section IV. Finally, we conclude the paper in Section V.

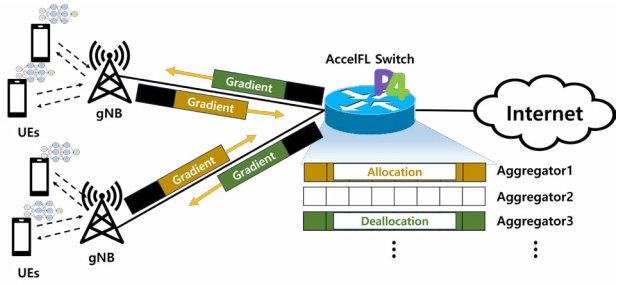
II. System Model

Figure 1(a) and Figure 1(b) illustrate the architecture of AccelFL, which consists of user equipment (UE), gNodeB (gNB), and an AccelFL switch.

In our system model, UEs serve as participants in FL. UEs generate two types of packets: local gradient packets for training a global model and regular data traffic for general data



(a)



(b)

Fig. 1: (a) AccelFL with normal packets. (b) AccelFL with gradient packets.

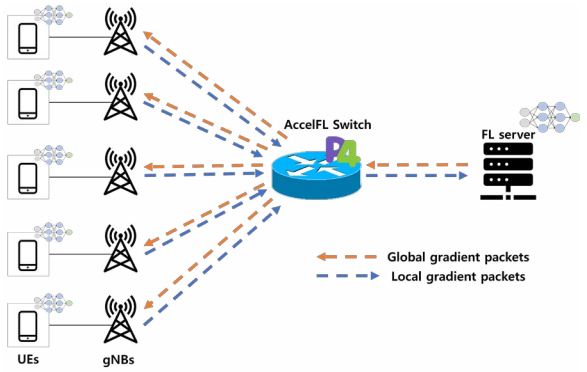


Fig. 2: Experiment Topology.

communication purposes. These packets are sent to the gNB, then gNB encapsulates them with a GPRS tunneling protocol (GTP) header for tunneling purposes. Once the packets reach the gNB, they are forwarded to the AccelFL switch. Subsequently, the AccelFL switch identifies whether the packets are local gradient packets or regular data packets. When the packet is a regular data packet, the AccelFL switch proceeds to forward it to the external internet as part of regular data communication flow as shown in Figure 1(a). However, when the packet is a local gradient packet, the AccelFL switch initiates an INA process. In this case, it aggregates the local gradients contained within the packet. After all local gradients are received and aggregated, the AccelFL switch multicasts aggregated gradient packets to multiple UEs, enabling FL across the networks as shown in Figure 1(b). Following the above procedures, UEs within training the same job can update the local model with globally aggregated gradients.

III. Design of AccelFL

As depicted in Figure 1, AccelFL provides both UPF and INA so the pipeline of the AccelFL switch is constructed in two components: the first one is about UPF and the other one is about INA.

The UPF component in the AccelFL handles regular data packets. It provides several functionalities of UPF such as packet classification, Quality of Service (QoS) enforcement, and GTP tunneling based on match-action tables in PDP. The packet classification classifies the packet's direction, application ID, and traffic class for QoS enforcement and GTP tunneling. AccelFL uses the IP header's destination IP address, port number, and protocol number to match the packet's traffic class and direction. According to this matched information of packets, AccelFL sets meter [4] which measures the traffic rate of a specific traffic class. If the meter indicates RED which means excess of a threshold, AccelFL forces to drop packets for QoS enforcement. GTP tunneling is then performed to encapsulate GTP headers for downlink packets or decapsulate GTP headers for uplink packets.

The INA component in the AccelFL handles gradient packets. It allows the aggregation of local gradients by dynamically allocating a register to store partially aggregated local gradients. The index of the allocated registers is distinguished by a pair of application IDs and a sequence number indicating the ID of FL model and the order of generated gradients respectively. During aggregation, AccelFL drops the gradient packets until they are fully aggregated. When gradients are fully aggregated, It multicasts the aggregated global gradient to UEs and deallocates the register to reuse for other gradients. With this design, AccelFL provides a reduction in training time and communication overhead. Since the aggregation for the global update in traditional FL is done in the FL server, every local gradient packet should be sent to the FL server, which causes massive communication overhead and additional delay. However, AccelFL addresses these problems by aggregating gradients at line-rate processing and dropping the unnecessary packets that are not fully aggregated. In addition, it can provide scalability by dynamically allocating registers during aggregation.

In summary, we can improve the latency of both regular data packet processing and FL training by using AccelFL. Packet processing delays for the regular data packets are reduced by the UPF component, and the communication efficiency and FL training latency are improved by the INA component of AccelFL. Therefore, AccelFL has its strength in mobile networks where low latency and sufficient bandwidth are paramount.

Number of Packets	1000	5000	10000
AccelFL	3.3	16.3	32.6
Baseline Scheme	5.2	25.8	51.7

TABLE I: Total Amount of Traffic (MB).

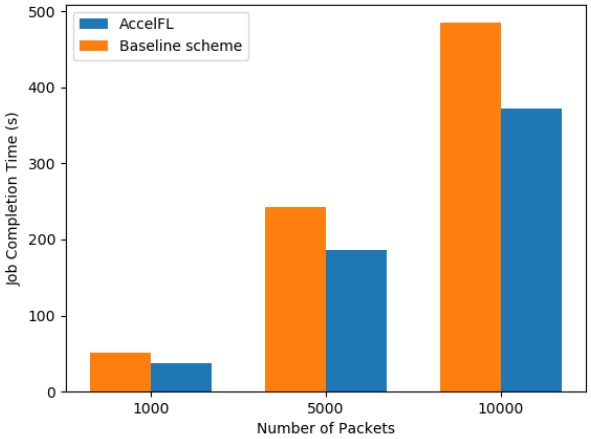


Fig. 3: Job Completion Time.

IV. Evaluation

We configured the topology as depicted in Figure 2. The implementation of AccelFL is based on [3], [7] and targets to BMv2 software programmable switch. The topology is composed of an AccelFL switch and 5 UEs that generate local gradient packets, and the FL server that is used in a baseline scheme. For the purpose of comparison, we compared the performance of AccelFL with the baseline scheme which employs the traditional FL process. In the baseline scheme, UEs generate local gradient packets and the FL server is responsible for aggregating local gradients and updating the global model. To demonstrate the performance of training acceleration and communication overhead with AccelFL, we measure two metrics: JCT and the total amount of traffic. JCT indicates the total time consumed until the last global gradients arrive at the UEs. The total amount of traffic indicates the cumulative amount of traffic during the experiment.

Figure 3 shows the JCT performance in three cases that differ in the number of gradient packets. Our evaluation demonstrates that AccelFL achieves a remarkable reduction of 30% in JCT in all cases when compared to the baseline scheme. This improvement can be attributed to AccelFL's ability to aggregate local gradients at line-rate speeds within the network with PDP, eliminating the need for aggregations in the FL server. Furthermore, by eliminating the need to exchange gradient packets with the FL server, AccelFL achieves an additional reduction in communication delay.

Table I presents the reduction in communication overhead with AccelFL in three cases that differ in the number of gradient packets. As shown in Table I, the communication overhead is reduced by 36.9% than the baseline scheme. This is because AccelFL can reduce the traffic path for gradient packets. In the baseline scheme, gradient packets have to travel an additional path compared to AccelFL to aggregate gradients in the FL server. However, AccelFL provides aggregation functionality in the switch, which can eliminate communication between the switch and the FL server. Therefore, the total amount of traffic of AccelFL is reduced.

V. Conclusion

We proposed AccelFL, an INA service with UPF functionality designed to accelerate FL in the mobile network. AccelFL switch provides UPF operations for regular data packets, including GTP handling and QoS enforcement. it also takes on an additional role when dealing with FL model update packets, acting as an aggregator that aggregates local gradient values and multicasts the resulting global packets to update local FL models. Our experiments have shown that AccelFL can reduce JCT and communication overhead by up to 30%.

In our future work, we will extend our source code to facilitate the implementation of AccelFL on the Tofino hardware programmable switch [2]. In addition, we will integrate the FL framework to evaluate the performance in a more practical environment.

Acknowledgment

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-01015, Development of Candidate Element Technology for Intelligent 6G Mobile Core Network), by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and the Ministry of Trade, Industry & Energy (MOTIE) of the Republic of Korea (No. 20212020800120), and by National Research Foundation (NRF) of Korea Grant funded by the Korean Government (MSIT) (No. 2021R1A4A3022102).

References

[1] P. Bosshart *et al.*, "P4: Programming Protocol-Independent Packet Processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87-95, July 2014.

[2] Intel Tofino. [Online] Available: https://www.intel.com/content/www/us/en/products/details/network-io/intelligent-fabric-processors/tofino.html [Accessed: 15-Oct-2023]

[3] R. MacDavid *et al.*, "A P4-baed 5G User Plane Function," in *Proc. ACM SOSR 2021*, October 2021.

[4] J. Heinanen *et al.*, "RFC 2698: A Two Rate Three Color Marker," Technical report, RFC Editor, USA, 1999. https://www.rfc-editor.org/rfc/rfc2698.html

[5] W. Liu *et al.*, "Programmable Data Plane Intelligence: Advances, Opportunities, and Challenges,"*IEEE Network*, early access.

[6] A. Sapio *et al.*, "Scaling Distributed Machine Learning with In-Network Aggregation," in *Proc. USENIX NSDI 2021*, April 2021.

[7] C. Lao *et al.*, "ATP: In-network Aggregation for Multi-tenant Learning," in *Proc. USENIX NSDI 2021*, April 2021.

[8] S. Banabilah *et al.*, "Federated learning review: Fundamentals, enabling technologies, and future applications," *ELSEVIER Information Processing & Management*, vol. 59, no. 6, pp. , November 2022.