The Response to Reviewers for Minor Revision

IEEE TRANSACTIONS ON MOBILE COMPUTING

Manuscript ID: TMC-2023-02-0157

RingSFL: An Adaptive Split Federated Learning Towards Taming Client Heterogeneity

Jinglong Shen, Student Member, IEEE, Nan Cheng, Member, IEEE, Xiucheng Wang, Student Member, IEEE, Feng Lyu, Member, IEEE, Wenchao Xu, Member, IEEE, Zhi Liu, Member, IEEE, Khalid Aldubaikhy, Member, IEEE, and Xuemin (Sherman) Shen, Fellow, IEEE

Dear Editor and Reviewers

Thank you for giving us the opportunity to submit a revised draft of the manuscript "RingSFL: An Adaptive Split Federated Learning Towards Taming Client Heterogeneity" for publication in IEEE Transactions on Mobile Computing. We appreciate the time and effort that editors and the reviewers dedicated to providing feedback on our manuscript and are grateful for the insightful comments and valuable improvements to our paper. We have incorporated most of the suggestions made by the reviewers, and have made the following major changes:

TODO

Please refer to the followings for a point-by-point response to the reviewers' comments and concerns. All page numbers refer to the revised manuscript.

Reviewer #1

The Reviewer's Comment #1.1

Response:

Reviewer #2

The Reviewer's Comment #2.1

Response:

The Reviewer's Comment #2.2

Response:

The Reviewer's Comment #2.3

Response:

The Reviewer's Comment #2.4

Response:

The Reviewer's Comment #2.5

Response:

The Reviewer's Comment #2.6

Response:

The Reviewer's Comment #2.7

Response:

Reviewer #3

The Reviewer's Comment #3.1

Previous comment 6: Multiple mini-batch: In federated learning, multiple mini-batch updates are performed before sending the updated model to the server to handle the communication bottleneck. However, in this scheme, to perform another mini-batch update, the full communication round among clients are required for the forward/backward propagation process.
–> Regarding this comment, the authors mention that communication between clients can be smaller compared to the communication between the clients and the server. This is reasonable. However, this is not my main point. Consider performing E local updates at each client using its local dataset. To achieve this, in conventional FL, each client just needs to perform E updates locally, without requiring any communications before the model aggregation process (Some stragglers may perform less than E local updates before aggregation). In RingSFL, however, even during the training process, this requires $2 * E * N$ full rounds of communications between all clients for backward/forward propagation. In other words, it requires $2 * E * N^2$ hops of D2D communications. I agree that one-hop communication time between devices could be relatively small, but can we assume that a full-round communication for all client is small? Given this, I am currently not convinced with the practicability of RingSFL. Is there a way to reduce this D2D communication burden caused by multiple local updates? Since handling the stragglers is one of the authors' main goal, I strongly believe that this issue should be clearly discussed in the revised manuscript.

Response:

Thanks for your constructive comments and for giving us the opportunity to explain this issue. In machine learning, forward propagation and back propagation is a serial process. Therefore, in RingSFL, if all clients are made to form the same

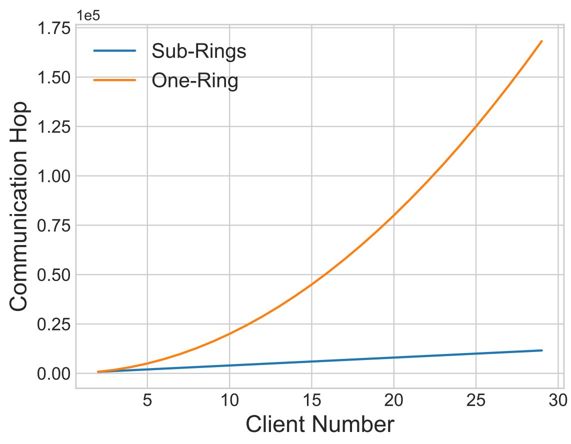

Figure R1: Effect of the number of clients on the number of communication hops in a communication round.

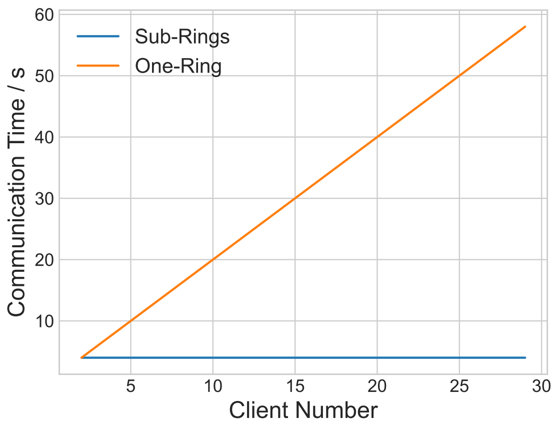

Figure R2: Effect of the number of clients on the communication time in a communication round.

ring, it is possible that the training efficiency of the system will be affected due to the accumulated delay of multi-hop communication.

This issue can be addressed by dividing the large ring consisting of all clients into multiple subrings consisting of partial clients, thus reducing the number of hops in each subring. Specifically, the server can group the clients in advance so that each group of clients forms a subring, and subsequently train in each subring using the method described in the manuscript, and when all subrings have completed local training, all clients send their local models to the same server for aggregation. At this point, the system contains multiple subrings and each subring contains a suitable number of clients. Since the number of clients in each subring is relatively small, it can be ensured that the latency of multi-hop communication will not have an excessive impact on the training efficiency of the system.

This is an issue of communication topology construction, which at the time of writing the manuscript we had placed in the future work section, and on which we have recently made some progress.

We use a concrete example to illustrate the gain in communication that results from being divided into multiple subrings. Consider a RingSFL system containing N clients, each of which performs E local updates in each communication round. If we fix each group to contain only two clients when grouping the clients, each subring in the system consists of two clients, totaling $N/2$ subrings. Then in each communication round, each client needs to perform a total of $2 * E * 2 = 4 * E$ hops of communication, and each subring needs to perform $2 * E * 2^2 = 8 * E$ hops of communication, and the total number of communication hops the whole system needs to perform is

$$H_{Sub-Rings} = 2 * E * 2^2 * (N/2) = 4 * E * N \tag{1}$$

And when all the clients are formed into the same ring, the total number of communication hops to be performed by the whole system is

$$H_{One-Ring} = 2 * E * N^2 \tag{2}$$

Since $N \geq 2$, we have $H_{Sub-Rings} \leq H_{One-Ring}$. From the Figure R1, we can see that dividing into multiple sub-rings can effectively reduce the number of communication hops of the system.

From the perspective of communication time consumption. Assuming that the average delay of one-hop communication is Q, and since the clients are in parallel with each other and the different subrings are also in parallel with each other, the communication time for the system to complete a communication round is

$$T_{Sub-Rings} = (2 * E * 2) * Q = 4 * E * Q \tag{3}$$

And if all clients form the same ring, the required communication time is

$$T_{One-Ring} = 2 * E * N * Q \tag{4}$$

Since $N \geq 2$, we have $T_{Sub-Rings} \leq T_{One-Ring}$. As can be seen from the Figure R2, after dividing into multiple sub-rings, we can ensure that the communication time consumption does not become unacceptable due to the excessive number of clients involved in the training.

The Reviewer's Comment #3.2

Previous comment 7: Finally, I am not very convinced with the ring-shaped architecture itself for handling the straggler issue. For example, when a specific client becomes outage due to the battery issue as described in the introduction, it results in significant bottleneck since every forward/backward signals cannot pass that client. In contrast, in conventional FL with a single server, it is acceptable to have some stragglers in each round because one can ignore them.
–> Regarding this comment, I understand the authors' comment. If the computation/communication capabilities are known, and if there are no clients dropping out, there is no issue at all. However, if a specific client drops out, this causes a critical issue in RingSFL while conventional FL can simply ignore that client. I believe that this issue should be also clearly discussed in the revised paper.

Response: