Many thanks to the reviewers for their deep, thoughtful reviews and constructive suggestions.

Main Contributions: We would firstly explain the theoretical contribution of our work. Our algorithm is the first to provide a provably communication-efficient and distributional robust FL algorithm, and both the design and the analysis of the algorithm is technically non-trivial.

(**R4**): Convergence analysis. To solve the minimax problem:$\min_{\boldsymbol{w}} \max_{\boldsymbol{\lambda}} F(\boldsymbol{w}, \boldsymbol{\lambda})$, our goal is to find the saddle point $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*)$ such that $\forall \boldsymbol{w}, \boldsymbol{\lambda}$, $F(\boldsymbol{w}^*, \boldsymbol{\lambda}) \leq F(\boldsymbol{w}^*, \boldsymbol{\lambda}^*) \leq F(\boldsymbol{w}, \boldsymbol{\lambda}^*)$. We use the primal-dual gap: $\max_{\boldsymbol{\lambda}} F(\hat{\boldsymbol{w}}, \boldsymbol{\lambda}) - \min_{\boldsymbol{w}} F(\boldsymbol{w}, \hat{\boldsymbol{\lambda}})$ as convergence measure, a classic convergence measure used in convex-concave minimax optimization. It measures how much our solution $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\lambda}})$ is far from the saddle point $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*)$ (if our solution is exactly the saddle point, then the primal-dual gap is zero). Our theoretical novelties are: (1) Our problem setting is novel. We study the minimax optimization problem where primal variable is trained locally, averaged periodically, and the dual variable can only be updated periodically. (2) To mitigate the issue that dual variable cannot get updated every iteration, we propose a novel gradient sampling method (as pointed by **R2** as a "clever algorithm" which we appreciate). By this gradient sampling update scheme, we have a unbiased approximation on the history gradients of dual variable. (3) We give the convergence analysis on convex-linear and nonconvex-linear settings. Nonconvex-concave (linear) is one of the hardest cases in minimax optimization, and in this work we even have more difficulties, due to the irregular updating scheme compared to a single machine case. In addition, we also give the analysis for regularized setting: strongly-convex-strongly-concave, and PL-strongly-concave cases. We totally agree with (**R4**) that we can not have both strong convexity and bounded gradients assumptions at the same time **unless** the problem is defined over a compact set. In our analysis we assume that parameter domain is bounded (Assumption 3), thus having both assumptions is sound. Currently we cannot remove bounded gradient assumption, because we need to characterize the variance of history gradient estimation of $\boldsymbol{\lambda}$. (**R3**) commented that "server choosing clients" mode cannot solve low participation issue. The solution can be that server sends request to more clients and waits until K clients responded. The convergence analysis will be trickier though.

For the questions about convergence rate, we admit that to achieve communication efficiency, we sacrifice slightly on convergence rate (**R4**). However, that does NOT mean our algorithm is slower (**R4**). The communication cost is the major bottleneck that slows down the distributed training. As shown in the experiments, our algorithm converges faster than vanilla Agnostic Federated Learning (AFL). Compared to gradient descent (**R1**), in [27] they get $O(1/\sqrt{T})$ rate, slightly better than us, but not communication efficient. As for the choice of τ (**R3**), it is a very good question. We do optimize on η, γ and τ to get the best convergence rate. The convergence rate is presented as polynomial of τ in the Appendix proofs. (**R4**) also had a comment that the same τ for all clients are not nice since clients will have different amounts of data. We respectfully disagree with this opinion, since each client will only compute gradient on a small fixed mini-batch of data, so it does not matter how many data they have in total.

For the privacy issue, (**R2**) proposed a very good question: since server asks client to evaluate its loss at $\tilde{\boldsymbol{w}}$ with data point $\xi = (\boldsymbol{x}, y)$, and return the loss $f_i(\tilde{\boldsymbol{w}}, \xi) = a$, does it mean server can infer the data point information equipped with $\tilde{\boldsymbol{w}}$ and $f_i(\tilde{\boldsymbol{w}}, \xi)$? Let us consider the simple linear regression case. If server wants to infer data point $\xi = (\boldsymbol{x}, y)$, it needs to solve the following problem: $\|\tilde{\boldsymbol{w}}^\top \boldsymbol{x} - y\|^2 = a$ to find out $\boldsymbol{x}$ and y. This problem does not admit a unique solution, or namely, server does not have enough information to determine $\boldsymbol{x}$ and y at the same time. However, we should admit, there is still chance that information could be leaked due to some model inversion attack, but it is not just the problem in our work, but the problem in whole FL community. Thus the suggestions from (**R2**) about adding noise or model compression will be good future directions.

For experiment part, (**R1**) pointed out that our dataset partition is not practical and we should follow AFL's setup. We would argue that we exactly followed them: (1) We use the Fashion MNIST dataset, and adult dataset (in Appendix), the same as them; (2) We partition the dataset by giving one client only one class data, the same as them; (3) We use the full 10 classes Fashion MNIST dataset but AFL only use part of dataset, so our experiments are even more convincing; and (4) We compare the model trained in average domain (FedAvg) and trained by distributionally robust domain, the same as what they did with Fashion MNIST dataset, yet one difference is that we do not train the model fully on local data, but it is already widely observed (even in AFL paper) local model will have very poor out-of-distribution performance, let alone the worst distribution performance. The other question is that in Fig.2, the DRFA has the same performance with conventional FL (**R4**). This is because Fig.2 is showing the average distribution performance, hence, it is reasonable that in average our algorithm performs similar to FedAvg. We should mention that the main point is that our method is better than FedAvg in worst distribution performance, as shown in Fig.3. (**R4**) commented that our work may not be as efficient as asynchronous FL. We are confused in which aspect our work is worse, can asynchronous FL achieve more distributional robustness? or it can solve the DRO with less communication cost? If so, it would be great if providing related works and we are very happy to compare with them in subsequent version.

For some minor questions, Alg.1 L4, 12, 13 should be $\bar{\boldsymbol{w}}$ instead of $\boldsymbol{w}$, and in L12, 13 local models should not have bars. L166, it should be minimizer of loss functions not gradients. L187 we missed square sign. L218,we mean single machine case. We would also thank **R1** for different FL protocol papers, and we will discuss them.