Effect of topology spectral properties on convergence (All Reviews) Spectral properties of the topology do have an effect on the number of iterations to converge (lines 42–44 in our paper). But there is evidence that, in practice, this effect has been over-estimated by classic worst-case convergence bounds from [23,71,73]. We point the reviewers to [75], which partially explains the phenomenon and overviews theoretical results proving asymptotic topology-independence [58,81,5] and experimental evidence on image classification tasks ([75, Fig. 2], [62, Fig. 20], [58, Fig. 3]) and—we add—natural language processing tasks [58, Figs. 13-16]. Reference [47, Sec. 6.3] extends some of the conclusions in [75] to dynamic topologies and multiple local updates. Paper [5] (pointed in Review #2) proves convergence to be asymptotically topology-insensitive [5, Th.1] and shows the topology has a more significant effect on system throughput than on the number of iterations to converge [5, Figs. 1b and 3] with sparser topologies achieving higher accuracy *after the same time* [5,Table 5]. Motivated by these observations, we design topologies to maximize throughput rather than spectral gap (convergence is guaranteed by the choice of the consensus matrix). Our experiments show that this choice is correct, as the topologies selected by our algorithms achieve faster training than the STAR, which has optimal spectral properties, and MATCHA, which optimizes spectral properties given a communication budget (lines 299–301). As suggested in Review #1, we will explore how to further improve speedup by taking into account the spectral properties (lines 301–302). Our recent results show that, by optimizing the weights of the consensus matrix, MST training speedup in comparison to the STAR increases on average by an additional 20% (e.g. from 6x to 7.2x for Géant).

Why designing the topology if RING is nearly the fastest and sparsest one? (Review #2) RING is not always the fastest topology: MST and δ-MBST may have a smaller cycle time (and then larger throughput) than RING (Fig. 3), in particular when delays in the core network are the dominant component in Eq. (3). Whenever MST and δ-MBST have throughput close to RING, they achieve faster training, as they have better spectral properties. For example, MST achieves 1.35 training speed-up vs RING when training iNaturalist over Géant (not reported in Table 3). After these clarifications, the fact that the RING is often the best choice is indeed a perhaps surprising finding of our paper. Nevertheless, note that the RING we consider is not a generic ring (there are $(n-1)!$ rings in a complete n-node network), but the one selected by our algorithm with provable guarantees on the cycle time (Prop. 3.3 and 3.6).

Comparison with MATCHA (Review #2) The reviewer is right that MATCHA [99] selects more frequently the important links. Our description of MATCHA in the introduction may be too synthetic, but we confirm that we implemented the complete MATCHA solution including the optimization of matching selection probabilities according to (5) in [99] (see function `get_matching_activation_probabilities` in our code). For the communication budget we have selected the value $C_b = 0.5$, which is the typical value used in experiments in [99]. There is no real configuration criterion for C_b in [99], but [99, Fig. 3] suggests to select the smallest C_b that has the same spectral norm of vanilla-SGD—but less communication overhead. This criterion leads to pick for all our topologies, but "AWS North America," a value of $C_b \in [0.4, 0.6]$, with no significant change to the results in our paper—actually sometimes MATCHA performs worse than before. For "AWS North America" the criterion leads to $C_b = 0.2$. Table 1, first row, shows indeed that MATCHA is faster for $C_b = 0.2$, but still RING is 1.08 and 3.29 faster than MATCHA for 10 Gbps and 100 Mbps access links capacities, respectively. The table shows also that this criterion does not lead necessarily to the fastest training time for MATCHA. The alternative is to select C_b by running time-consuming training experiments, but in any case we have always observed RING to outperform MATCHA except on Géant (Table 1 below and Table 3 in our paper). Review #2 suggested to run MATCHA on our overlays. Note that MATCHA is supposed to find by itself how often to use each link and "achieve a win-win in this error-runtime trade-off for *any arbitrary network topology*" [99]. We run additional experiments with MATCHA over our topologies (for the ring we consider its undirected version as MATCHA uses bi-directional communications); however, MATCHA is still slower than RING (Table 1).

Impact of data partitioning on the model (Review #3) The different accuracy reached in the different networks is indeed due to minimizing the sum of the average losses as indicated in (1). There are two common settings in federated learning [53, Eq. 1], either optimizing the mean of local functions [39, Eq. 2] as considered in our and MATCHA's paper (arguably better for per-client fairness in non-IID settings), or optimizing the weighted sum of local functions as suggested by Review #3. We run experiments also in the second case and confirm that models' final accuracy is independent from the network. The other reviewer's comments are also spot on, we will improve the paper accordingly.

Table 1: RING's training speed-up vs MATCHA in AWS-North America network. MATCHA runs on top of underlay, RING, and δ-MBST with different values of communication budget C_b. 1 Gbps core links capacities. The star denotes the results with C_b set according to [99, Fig. 3]. Bold fonts denote the optimal setting for C_b.

Access links capacities	10 **Gbps**							100 **Mbps**						
Communication budget (C_b)	1.0	0.8	0.6	0.5	0.4	0.2	0.1	1.0	0.8	0.6	0.5	0.4	0.2	0.1
MATCHA over underlay	2.02	1.43	1.57	1.47	1.46	**1.08***	1.38	18.85	12.56	12.00	9.94	8.18	3.29*	**2.44**
MATCHA over δ-MBST	**1.10***	1.25	1.33	1.12	1.41	1.89	2.28	2.08*	2.26	1.56	1.45	1.31	**1.15**	1.15
MATCHA over RING	**1.00***	1.42	1.40	1.15	1.26	1.35	1.34	**1.00***	2.15	1.92	1.47	1.54	1.41	1.28