HEAL: Resilient and Self-* Hub-based Learning

Mohamed Amine Legheraba, Stefan Galkiewicz, Maria Potop-Butucaru, Sébastien Tixeuil

HEAL: Resilient and Self-* Hub-based Learning

Mohamed Amine Legheraba[1], Stefan Galkiewicz[1], Maria Potop-Butucaru[1], and Sébastien Tixeuil[1,2]

[1] Sorbonne University, CNRS, LIP6, F-75005 Paris, France
[2] Institut Universitaire de France, Paris, France

Abstract. Decentralized learning enhances privacy, scalability, and fault tolerance by distributing data and computation across nodes. A popular approach is Federated learning, which relies on a central aggregator, yet faces challenges such as server vulnerabilities, scalability issues, privacy risks and most importantly, the single point of failure. Alternatively Gossip Learning and Epidemic Learning offer fully decentralization through peer-to-peer exchanges of model updates, ensuring robustness and privacy, at the price of slower model convergence. In this work, we introduce a novel decentralized learning framework called HEAL. HEAL is the first cross-layer decentralized learning framework that exploits an optimized self-organizing and self-healing underlying P2P overlay combining the strengths of Federated Learning, Gossip and Epidemic Learning. Leveraging the recently proposed Elevator algorithm, HEAL promotes dynamically chosen nodes to act as aggregators. Through simulations, we demonstrate that HEAL has similar performances to that of Federated Learning in crash-free settings, while being fully decentralized and fault-tolerant. In crash and churn prone environments HEAL outperforms Gossip and Epidemic Learning.

Keywords: Decentralized Learning · Federated Learning · Gossip Learning · Hub Learning · Machine Learning · Peer-to-peer networks · Hub sampling · Algorithms · Simulations.

1 Introduction

Decentralized learning is an approach to training machine learning models, where the data and computational tasks are distributed across multiple nodes or devices, rather than centralized in a single location. This paradigm improves privacy, scalability, and fault tolerance by enabling participants to collaboratively train models without sharing raw data. Each node processes its local data and exchanges model updates (e.g., gradients or parameters) with others, using a centralized coordinator or through peer-to-peer communication. Decentralized learning is particularly beneficial in scenarios where data is naturally distributed (such as in edge computing or IoT networks), or when data privacy or resource constraints make centralization impractical.

Related works. The idea of decentralized learning originates from distributed optimization and parallel computing. However, it was the advent of Federated Learning [15] that truly brought the concept into the spotlight. In Federated Learning all participants (nodes or devices) in a network train a machine learning model locally on their own data. Subsequently, each participant sends its model to a centralized aggregator, which aggregates the received models and returns the global model to the nodes. This process continues until a satisfactory accuracy is achieved. While Federated Learning provides an efficient alternative to traditional (centralized) machine learning, its dependence on a central server for aggregating models introduces several limitations. This centralization creates a *single point of failure*. That is, when the server crashes the generation of the global model becomes impossible. Furthermore, the single server is an easy target for adversarial attacks, such as model poisoning or inference attacks [19], which can compromise the integrity of the global model. Additionally, the server's role poses scalability challenges, as it must manage potentially vast numbers of updates from distributed participants, leading to communication and computation bottlenecks. More importantly, if the server is controlled by a single organization, it could introduce biases or favor certain participants' updates, exacerbating inequalities in model performance. This phenomenon leads to *leaning oligarchy*.

To address these limitations, it is essential to explore totally decentralized architectures which do not rely on a central coordinator. Gossip Learning [17] and Epidemic Learning[4] are the state of the art approaches for fully decentralized machine learning. However, the stochastic nature of these protocols can result in slower convergence compared to Federated Learning-based approaches [6]. It should be noted that in the case of gossip learning, recent research has been conducted to enhance its various aspects, such as improving its security against privacy attacks [3], increasing its efficiency by compressing the models sent [2]. More recent studies focused on examining the impact of poisoning attacks [18] on various gossip learning strategies.

Interestingly, none of the previous decentralized federated approaches had a cross-layer design philosophy.

Our contribution In this paper we introduce a novel form of decentralized learning, HEAL, which combines the benefits of Federated Learning, Gossip and Epidemic Learning (summarized in Table 1). HEAL leverages the network overlay created by the newly introduced Elevator algorithm [13] that promotes in a totally decentralized and adaptive manner a prescribed number of participating nodes as hubs (nodes connected to the entire network). HEAL will use these hubs as aggregator nodes for the learning task. Unlike traditional federated learning, aggregator nodes are not pre-selected and hence HEAL becomes extremely resilient to the network dynamicity (nodes crashes and churn). That is, HEAL self-heals and self-adapts by promoting new hubs while continuing the learning process without significant losses. We challenged HEAL, Federated Learning, Gossip Learning and Epidemic Learning with various topologies in static and dynamic environments (crash and churn prone) on two tasks: a binary classification task Logistic Regression [8] on Spambase [7] and on multinomial classifica-

tion tasks LeNet5 [12] on MNIST [23]. HEAL outperforms Gossip and Epidemic Learning in both accuracy and convergence time. Moreover, HEAL is resilient to churn and crashes while Federated Learning cannot cope with these faults.

Decentralized Federated Learning	**Topology**	**Fault resilience**	**Churn resilience**	**Aggregation speed**
Federated Learning	Static (Star [15] and Multi-Star [9])	No	No	quick at the server
Gossip Learning	Static (Random Regular [6])	No	No	slow local
	Dynamic (with Newscast [17])	Yes	Yes	slow local
Epidemic Learning	Dynamic (Newscast [4])	Yes	Yes	slow local
	Dynamic (FedLay [10])	Yes (only one)	No	slow local
HEAL (this paper)	**Dynamic (Elevator [13])**	**Yes**	**Yes**	**quick at the hubs**

Table 1: Comparison of different decentralized learning algorithms.

Paper organization In Section 2 we propose an overview of decentralized learning strategies. Section 3 introduces the architecture of HEAL and the detailed description of the HEAL learning strategies. Section 4 presents our extensive evaluations. Section 5 concludes and proposes open research directions.

2 Overview of Decentralized Learning Techniques

The distinguishing factors among various decentralized learning algorithms in the literature are: (1) the algorithm employed to propagate and aggregate learning models within the network, and (2) the network topology on which the learning occurs. Naturally, these two concepts are interconnected, as certain propagation methods are better suited to specific topologies.

Decentralized propagation and aggregation of the learning models. There are various techniques for propagating the learning models within a network, each with its own set of advantages and disadvantages. The three most commonly discussed methods in the literature are *Federated Learning*, *Gossip Learning*, and *Epidemic Learning*. In Federated Learning [15], all models are aggregated at a central server, facilitating rapid convergence towards a global model. In Gossip Learning [17], each participant shares its model at a specified time interval with a randomly chosen neighbor in the network. In Epidemic Learning [4], each participant shares its model with all their neighbors in each cycle.

Network topology vs decentralized learning. In decentralized learning, the network topology significantly influences the performance of the learning task, as evidenced by various studies in the literature [21]. Different topologies impact the speed of information propagation and the system's resilience to node or link failures. At the extremes, we have the star topology, typically used with Federated Learning, and the random graph topology, often paired with Gossip Learning. However, many other topologies exist between these extremes. Additionally, it is important to distinguish between static topologies (predefined and unchangeable) and dynamic topologies (which evolve over time).

- Static topologies. Among static topologies, *star topology*, features a central server and clients connected to it. This setup is not entirely decentralized, as the aggregator server is selected at the beginning of the learning process, creating a single point of failure. The *multi-star topology* is a variation of the star topology, involving multiple servers. Typically, all stars are interconnected in a complete topology, with all other nodes connected to a predefined star. Next, we have the *complete topology*, where every participant communicates directly with all others. While this maximizes information propagation speed and ensures rapid convergence, it is impractical for large-scale networks. Another common topology is the *ring topology*, where each node is connected to two neighbors, forming a circular ring. This topology is straightforward to construct but does not scale well. Finally, in *random regular graphs* each participant is randomly connected to k other participants in the network, with k being a predefined parameter. This topology is highly robust against failures and churn, but the learning process convergence is slow. Other random topologies in the literature include small worlds [22] and power-law networks [1].
- Self-organised dynamic topologies. To establish a dynamic topology, two primary approaches are discussed in the literature: Distributed Hash Table (DHT)-based methods and decentralized peer sampling methods. Among the DHT-based methods, Chord [20] is a notable example. Additionally, Fedlay [10] is specifically designed for decentralized learning. For non-DHT methods, Newscast [11] is commonly used in Gossip Learning and Epidemic Learning.

An important aspect to consider in dynamic topologies is their resilience to failures. A specific type of failure is churn, where nodes in a peer-to-peer network enter and leave without any control. Another particular case of failures involves attacks targeting specific nodes, such as servers or central nodes. Centralized topologies are highly sensitive to these types of failures. In a static topology, there is no possibility to repair the topology in the event of a failure. In a DHT, some repairs are possible, but not always guaranteed face to high churn. Peer sampling algorithms are a good compromise to repair the topology when failures are detected and to be resilient to high churn.

HEAL, uses as underlying topology Elevator [13], a recently proposed decentralized peer-sampling algorithm. This algorithm enables nodes in a peer-to-peer

network to construct an overlay with h defined hubs, each hub being connected to all nodes in the network, with h being a parameter of the algorithm. Elevator is totally distributed, self-organizing and resilient to churn. In Table 1, we summarized the topologies used with various decentralized learning algorithms, along with the strengths and weaknesses of each approach.

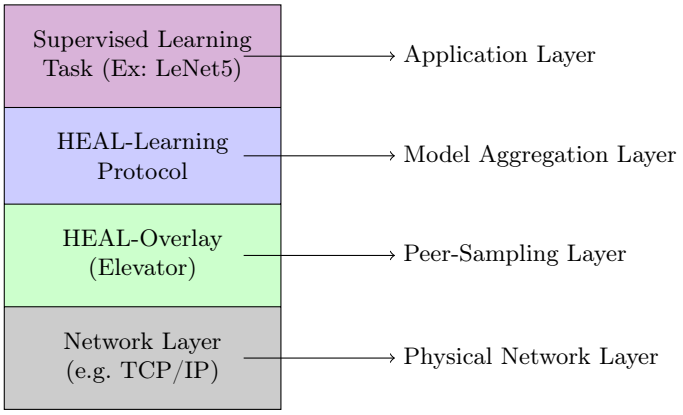


Fig. 1: HEAL architecture

3 HEAL architecture

Federated Learning is vulnerable due to its reliance on a central server, so our protocol must avoid having a single point of failure. Additionally, to ensure resilience to failures and churn, the topology should not be predefined but generated in a peer-to-peer manner. Conversely, to guarantee rapid model convergence, learning models should not be shared via gossip within the network but aggregated by a network node, which will then create the global model and distribute it back to the other network nodes. These two aspects may seem contradictory, but the HEAL-overlay (Elevator) protocol allows us to create a topology that satisfies both requirements.

HEAL architecture shown in Figure **??** is composed of two layers on top of the physical network. HEAL-overlay give by the Elevator protocol introduced in [13] and HEAL-learning protocol described in the sequel.

3.1 HEAL Overlay

In the following we briefly revisit how Elevator operates. For a more detailed explanation, readers can refer to the article that introduces the algorithm [13].

The Elevator protocol performs the following actions during each cycle: Each node in the peer-to-peer network retrieves the list of neighbors of their neighbors (i.e., the neighbors at a distance of two). The node then constructs an ordered list

Algorithm 1: HEAL Learning: The Hub algorithm

```
Data: duration to wait for model: delta_time
Data: list of all hubs (obtained by the HEAL overlay, Elevator protocol):
      hubs_list
nb_hubs ← hubs_list.size()
Loop
    models ← {}
    time ← time.now()
    backwards_list ← {}
    while time.now() < time + delta_time do
        peer_model, peer ← receive()
        models.append(peer_model)
        backwards_list.append(peer)
    average_model ← average(models)
    send(hubs_list, average_model)
    models_hubs ← {}
    models_hubs.append(average_model)
    nb_receive ← 0
    while nb_receive < nb_hubs do
        hub_model ← receive()
        nb_receive + +
        models_hubs.append(hub_model)
    global_model ← average(models_hubs)
    send(backwards_list, global_model)
```

Algorithm 2: HEAL Learning: The client algorithm

```
Data: duration to wait for model: delta_time
Data: The model, same for all node, weights or parameters initialized at
      random: model
Data: The local data of the node): data
Data: list of all hubs (obtained by HEAL overlay, Elevator protocol):
      hubs_list
Data: number of hub to send the model: number_hub_send
Loop
    model ← trainModel(model, data)
    hubs ← chooseRandom(hubs_list, number_hub_send)
    for hub ∈ hubs do
        send(hub, model)
    hubs_models ← list()
    // Receiving the global models from the hubs
    for hub ∈ hubs do
        model_hub ← receive()
        hubs_models.append(model_hub)
    model ← average(hubs_models)
```

of the most frequent peers (the frequency map) and contacts the c most frequent nodes (referred to as *preferred*). Each contacted node responds by sending the addresses from its backward list to the contacting node and adds the contacting node to its backward list. The contacting node's cache is then reset to an empty array. Subsequently, the node selects the h most frequent peers and c-h random peers from the backward lists of all preferred peers to populate its cache.

This protocol enables the rapid formation (in 4 cycles or fewer in practical settings) of a network topology with h defined hubs and a random distribution of the remaining incoming connections. The resulting network has a diameter of 2 and is highly resistant to both failures (including hub failures) and churn. In the event of all hubs failing, new hubs quickly emerge (typically with one cycle). These properties are particularly advantageous for decentralized learning, suggesting that we can implement a learning algorithm on this topology that achieves performance levels similar to Federated Learning while maintaining resilience properties as Gossip and Epidemic Learning.

3.2 HEAL Learning Protocol

Regarding the communication algorithm, we utilize the hubs within the network as aggregators, similar to how the central server aggregates models in Federated Learning. The key difference is that multiple hubs perform the aggregation, not just one, and these hubs emerge automatically. We leverage the presence of multiple hubs to distribute the aggregation workload, with each hub handling a portion of the network nodes and subsequently aggregating with each other. Each hub then returns the global model to its clients, and the protocol begins a new cycle. As with Federated Learning, the learning process continues over several cycles and concludes when the global model has converged.

In the event of one or more hubs failing during a protocol cycle, the remaining hubs can temporarily manage the nodes without a dedicated hub until a new hub emerges, which typically occurs within two cycles. If all hubs fail, the aggregation process halts but resumes as soon as new hubs appear, again within two cycles. It's important to note that Elevator also establishes random connections in the network, in addition to hub connections. In HEAL, we focus solely on connections to hubs (and between hubs) for model aggregation. These random connections could potentially be used to accelerate model convergence or mitigate malicious behavior, but this is beyond the scope of our current work. For now, we assume that all network nodes (and hubs) are honest, with plans to investigate malicious behavior in future research.

One intriguing feature of Elevator is the ability to select the number of hubs in the network through a parameter shared by all nodes. This is particularly valuable in HEAL, as it allows us to balance between having fewer hubs for higher convergence speed and more hubs for greater resilience to failures. Additionally, HEAL offers the flexibility to choose the number of hubs to which a client sends its model. A more detailed description of how Learning operates in HEAL follows.

Each node in the network executes the HEAL learning protocol, in addition to HEAL overlay construction via the Elevator protocol (that dynamically assigns "normal" (client) or "hub"(server) status to the participating nodes):

- If the node is a normal(client) node, it (1) selects a number of hubs(servers) at random, (2) performs a local training step, (3) sends the trained model to the hubs, and (4) waits for the global model.
- If the node is a hub, it (1) waits for a *delta* period to receive models from normal nodes, (2) aggregates these models by averaging their parameters, and (3) sends its aggregated model to all other hubs. (4) It then waits to receive models from other hubs and (5) aggregates all these models to obtain the global model. (6) Finally, the hub sends the global model back to the nodes.

Algorithms 1 and 2 present the detailed pseudo-code of HEAL Learning.

4 Evaluation results

We evaluated our algorithm using simulations on the Gossipy simulator [14]. We compared Hub Learning against Federated Learning [15], Gaia [9], Gossip Learning [17], Epidemic Learning [4], Epidemic Learning on a Chord topology [20], Epidemic Learning on a ring topology, and Fedlay [10]. For the static topologies (Federated Learning, Gossip Learning, Epidemic Learning, ring, Chord, and Gaia), we generated the topology using the Python library Networkx [5]. For the dynamic topologies (Fedlay and Hub Learning with Elevator), we generated the topology using the PeerSim simulator [16]. In Elevator (used by Hub Learning), the connections are directional. However, to compare them with other algorithms (which assume an undirected graph), we modified the underlying graph of the topology generated to make it undirected. All evaluations were conducted with a network of 100 nodes. For Elevator, we used 5 hubs, as we found this number to be a good balance between performance and resilience. For Gaia, we had 5 servers responsible for aggregation (to compare with the 5 hubs) and 19 nodes (or workers) attached to each server. Each algorithm was evaluated 5 times, and we present the average results obtained.

We assessed all protocols on two tasks: a binary classification task (Logistic Regression [8] on Spambase [7], with a learning rate of 0.1) and on multinomial classification tasks (LeNet5 [12] on MNIST [23], with a learning rate of 0.001). The weight decay (regularization parameter) was fixed at 0.01. Our algorithm was evaluated under various conditions: the failure of 20% of nodes, the failure of a hub, the failure of all 5 hubs, and during churn (where 10% of nodes disappear at each cycle and are replaced by new nodes).

All simulations were run on 16 vCPU, using 64G of memory, on a cluster composed of 10 servers, described in Table 2.

Machine	Memory	Processors	Cores
DELL PowerEdge XE8545	2 To	2 x AMD EPYC 7543	128 threads @ 2.80 GHz
DELL PowerEdge R750xa	2 To	2 x Intel Xeon Gold 6330	112 threads @ 2.00 GHz

Table 2: Description of the cluster

Crash-free, churn-free environment. For simulations without failures and churn, we ran all algorithms over 1000 cycles, on the 2 learning tasks (Spambase, MNIST). As shown in Figures 2a and 2b, Federated Learning performs best, which was expected. Surprisingly, the ring topology performs second best despite lower connectivity. Other topologies based on random graphs perform less well. HEAL, however, performs very well for both the Spambase and MNIST datasets.

In Tables 4, 5a and 5b we have compiled the results for all learning algorithms, for the three learning tasks, with the final accuracy, obtained after 1000 cycles, and the time to converge to a certain level of accuracy. Federated Learning, Gaia, and HEAL have very similar results, with a final accuracy of around 0.88 on Spambase, and 0.95 on MNIST. Gossip-based approaches achieve values of 0.85 and 0.91 on Spambase and MNIST. Convergence time is very fast for aggregator-based approaches: on Spambase, Federated Learning converges to 0.85 in 2 cycles, and HEAL takes 4 cycles. On the other hand, to converge on 0.90 accuracy, HEAL takes 339 cycles while Federated Learning takes 135 cycles, hinting at possible HEAL optimization, e.g. adjusting the learning rate. On MNIST, HEAL performs very well, even better than Federated Learning, and it converges to 0.95 accuracy in 76 cycles.

HEAL parameterized with number of hubs and number chosen hubs We ran simulations of HEAL, changing the number of hubs over 2000 cycles. On Figure 3a, we observe that increasing the number of hubs (and the number of hubs to which clients send their model) has almost no impact on accuracy, which is expected since hubs aggregate models. There is a slight drop in accuracy when the number of hubs is increased significantly, due to the fact that only non-hubs are learning, not hubs. Increasing the number of hubs to which we send our model, from 1 to $nb\_hubs/2$, slightly increases convergence speed. The results are summarized in Table 3.

Crashes-prone environment. We analyze the performance of the algorithms when the network suffers crashes (Table 6). To simulate a brutal failure we disconnected 20% of the nodes chosen uniformly at random, just after the start of the learning process, i.e., in this case, we have disconnected 20 nodes at cycle 10, and we compared HEAL with Chord, Gaia and Fedlay. HEAL is the algorithm with the best results, although Gaia remains very close, as seen in Figure 2c.

HEAL under churn environment and hub-targeted attacks. We further analyzed the performance of HEAL under network churn conditions. To simulate churn,

h	s	Accuracy	0.85	0.90	0.95
1	1	0.9667	21	37	90
2	1	0.9531	11	22	127
3	1	0.9607	11	19	64
7	1	0.9638	14	26	67
9	1	0.9653	16	29	69
13	1	0.9664	18	31	81
17	1	0.9665	20	33	88
21	1	0.9674	20	35	86
25	1	0.9657	25	35	87

(a) With s = 1

h	s	Accuracy	0.85	0.90	0.95
5	2	0.9589	7	15	52
7	3	0.9641	6	10	33
9	4	0.9647	6	11	36
13	6	0.9628	5	10	34
17	8	0.9601	5	10	37
21	10	0.9594	5	11	45
25	12	0.9607	5	10	47

(b) With s = $\frac{h}{2}$

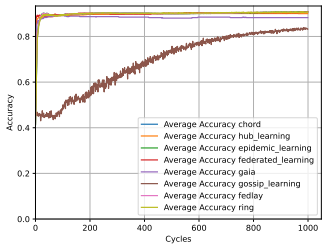
Table 3: HEAL final accuracy (on cycle n°2000) and number of rounds to obtain this level of accuracy, for a network of 100 nodes, a number of hubs h and a varying number of hubs s designated for each node, without failures, on the MNIST dataset.

we disconnected 10% of the nodes at each cycle and replaced them with an equal number of new nodes, each connected to 20 nodes uniformly at random, between cycles 50 and 150. We also analyzed the performance of the main learning algorithms after a targeted attack on the hubs during the execution of the simulation. We tested two scenarios, one where we disconnected one of the 5 hubs, and another where we disconnected all 5 hubs at the same time. In both case the failure happened in the round 10. In Figure 3b, we have compared the execution of HEAL without failures, and with different failure scenarios (crash of 20 nodes, crash of one hub, crash of all 5 hubs, churn). As can be seen, there is no significant impact when 20 nodes or hubs crash. Indeed, thanks to the Elevator overlay, even when all the hubs are shutdown, 5 new nodes are elected very quickly as hubs, and the training job continues as if no catastrophic event had happened. During churn, model accuracy falls slightly, but rises again very quickly once churn is over, back to the level without failures.

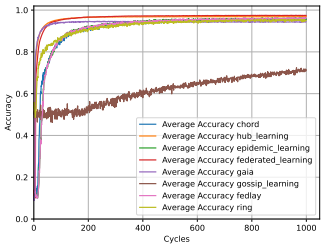
5 Conclusion

In this paper we introduced HEAL protocol for decentralized learning that combines the convergence speed of Federated Learning with the resilience to churn and failures of Gossip and Epidemic Learning.

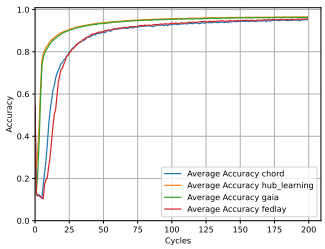
Our simulation results (summarized in Table 4, 5 and 6) show that, on the MNIST dataset, HEAL (with 5 hubs) achieves an accuracy 136% more than Gossip Learning, 106% more than Epidemic Learning and 99% of the accuracy of the baseline Federated Learning. HEAL achieves an accuracy of 0.95 in 76 cycles, which is one cycle slower than Gaia, and much faster than random graph methods, which achieve this value in 5 times as many cycles. By setting HEAL with 7 hubs and the number of hubs to which each node sends its model at 3,



(a) For the Spambase dataset, no failures

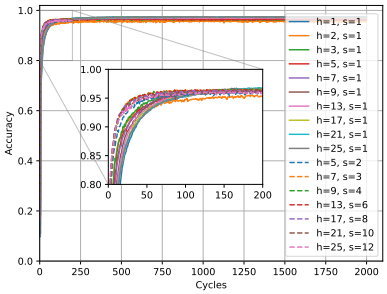


(b) For the MNIST dataset, no failures

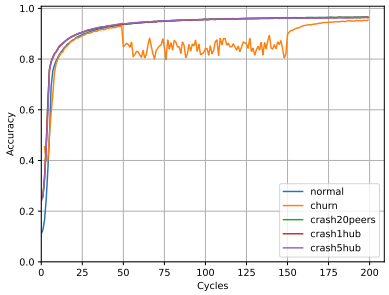


(c) For the MNIST dataset, when 20% of the nodes fail at round 10.

Fig. 2: Accuracy of various communication protocols, with a network of 100 nodes, during 1000 cycles. HEAL overlay has 5 hubs, each node sends its model to one hub.



(a) HEAL with different numbers of hubs (h), from 1 to 25, and different numbers of hubs to which nodes send their model (s), 1 or $\frac{h}{2}$, no failures, 2000 cycles



(b) HEAL accuracy for all contexts (no failures, crash of 20 peers, crash of 1 hub, crash of all hubs, churn), with 5 hubs, each node sent its model to one hub, 200 cycles.

Fig. 3: Accuracy of HEAL for the MNIST dataset, with 100 nodes

Method	**Spambase**	**MNIST (LeNet)**
Federated Learning	0.9087	0.9742
Gaia	0.8826	0.9442
Gossip Learning	0.8322	0.7098
Epidemic Learning	0.9548	0.9111
Ring	0.9076	0.9499
Chord	0.9063	0.9542
FedLay	0.9056	0.9639
HEAL	0.9001	0.9687

Table 4: Final accuracy by communication method and dataset used. HEAL overlay with 5 hubs, each node sends its model to one hub (fault and churn free scenario).

Method	**0.85**	**0.90**
Federated Learning	2	135
Gaia (5 servers)	6	N/A
Gossip Learning	N/A	N/A
Epidemic Learning	12	25
Ring	13	141
Chord	12	27
FedLay	12	26
HEAL (5 hubs, each node sent it model to one hub)	4	339

(a) On the Spambase dataset.

Method	**0.85**	**0.90**	**0.95**
Federated Learning	22	37	91
Gaia (5 servers)	13	28	75
Gossip Learning	N/A	N/A	N/A
Epidemic Learning	90	137	372
Ring	74	157	569
Chord	98	159	451
FedLay	89	136	422
HEAL (5 hubs, each node sent it model to one hub)	15	27	76
HEAL (7 hubs, each node sent it model to three hubs)	5	10	33

(b) On the MNIST dataset.

Table 5: Number of rounds required to achieve different accuracy levels across datasets, for a network of 100 nodes, without failures. N/A indicates that the protocol didn't achieve the accuracy level at the end of the simulation (1000 cycles).

Method	Context of crash	**Accuracy**
Gaia (5 servers)	crash of 20% of the nodes at cycle n°10	0.9637
Chord	crash of 20% of the nodes at cycle n°10	0.9520
FedLay	crash of 20% of the nodes at cycle n°10	0.9572
HEAL (5 hubs, each node sent it model to one hub)	crash of 20% of the nodes at cycle n°10	0.9658
HEAL (5 hubs, each node sent it model to one hub)	crash of 1 hub	0.9638
HEAL (5 hubs, each node sent it model to one hub)	crash of 5 hub	0.9629
HEAL (5 hubs, each node sent it model to one hub)	churn of 10% between cycle 50 and cycle 150	0.9528

Table 6: Final accuracy (on cycle n°200) by propagation method, after a context of failures, on the MNIST dataset.

it is possible to reduce it to 33 cycles, which is 2.3 times faster than Gaia (the second best result). Our protocol continues to operate in the presence of faults, and in each fault scenario, the final accuracy is at most equal to 98% of the accuracy in a fault-free context.

HEAL paves the way for a new approach to decentralized learning, featuring a cross-layer approach. Our future work will focus on adapting HEAL to heterogeneous environments, enhancing its robustness against various attacks (e.g. poisoning attacks, model attacks, etc).

References

1. Barabási, A.L., Bonabeau, E.: Scale-free networks. Scientific american **288**(5), 50–9 (2003)
2. Danner, G., Hegedűs, I., Jelasity, M.: Decentralized machine learning using compressed push-pull averaging. In: Proceedings of the 1st International Workshop on Distributed Infrastructure for Common Good. pp. 31–36 (2020)
3. Danner, G., Jelasity, M.: Fully distributed privacy preserving mini-batch gradient descent learning. In: Distributed Applications and Interoperable Systems: 15th IFIP WG 6.1 International Conference, DAIS 2015, Held as Part of the 10th International Federated Conference on Distributed Computing Techniques, DisCoTec 2015, Grenoble, France, June 2-4, 2015, Proceedings 15. pp. 30–44. Springer (2015)
4. De Vos, M., Farhadkhani, S., Guerraoui, R., Kermarrec, A.M., Pires, R., Sharma, R.: Epidemic learning: Boosting decentralized learning with randomized communication. Advances in Neural Information Processing Systems **36** (2024)
5. Hagberg, A., Swart, P.J., Schult, D.A.: Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Laboratory (LANL), Los Alamos, NM (United States) (2008)

6. Hegedűs, I., Danner, G., Jelasity, M.: Decentralized learning works: An empirical comparison of gossip learning and federated learning. Journal of Parallel and Distributed Computing **148**, 109–124 (2021)
7. Hopkins Mark, Reeber Erik, F.G., Jaap, S.: Spambase. UCI Machine Learning Repository (1999), DOI: https://doi.org/10.24432/C53G6X
8. Hosmer Jr, D.W., Lemeshow, S., Sturdivant, R.X.: Applied logistic regression. John Wiley & Sons (2013)
9. Hsieh, K., Harlap, A., Vijaykumar, N., Konomis, D., Ganger, G.R., Gibbons, P.B., Mutlu, O.: Gaia:{Geo-Distributed} machine learning approaching {LAN} speeds. In: 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17). pp. 629–647 (2017)
10. Hua, Y., Pang, J., Zhang, X., Liu, Y., Shi, X., Wang, B., Liu, Y., Qian, C.: Towards practical overlay networks for decentralized federated learning. arXiv preprint arXiv:2409.05331 (2024)
11. Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.M., Van Steen, M.: Gossip-based peer sampling. ACM Transactions on Computer Systems (TOCS) **25**(3), 8–es (2007)
12. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation **1**(4), 541–551 (1989)
13. Legheraba, M.A., Potop-Butucaru, M., Tixeuil, S.: Elevator: Self-* and persistent hub sampling service in unstructured peer-to-peer networks. arXiv preprint arXiv:2406.07946 (2024)
14. Makgyver: Gossipy (2024), https://github.com/makgyver/gossipy, python module for simulating gossip learning and decentralized federated learning. Accessed: 2024-11-21
15. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)
16. Montresor, A., Jelasity, M.: PeerSim: A scalable P2P simulator. In: Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09). pp. 99–100. Seattle, WA (Sep 2009)
17. Ormándi, R., Hegedűs, I., Jelasity, M.: Gossip learning with linear models on fully distributed data. Concurrency and Computation: Practice and Experience **25**(4), 556–571 (2013)
18. Pham, A., Potop-Butucaru, M., Tixeuil, S., Fdida, S.: Data poisoning attacks in gossip learning. In: International Conference on Advanced Information Networking and Applications. pp. 213–224. Springer (2024)
19. Rodríguez-Barroso, N., Jiménez-López, D., Luzón, M.V., Herrera, F., Martínez-Cámara, E.: Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges. Information Fusion **90**, 148–173 (2023)
20. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM computer communication review **31**(4), 149–160 (2001)
21. Vogels, T., Hendrikx, H., Jaggi, M.: Beyond spectral gap: The role of the topology in decentralized learning. Advances in Neural Information Processing Systems **35**, 15039–15050 (2022)
22. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world'networks. nature **393**(6684), 440–442 (1998)
23. Yann LeCun, C.C., Burges, C.J.: Mnist database of handwritten digits (2010), https://yann.lecun.com/exdb/mnist/, accessed: 2024-12-13