ABD-HFL: Byzantine-resistant Decentralized Hierarchical Federated Learning

Tengfei An, Maria Potop-Butucaru, Sébastien Tixeuil, Serge Fdida

ABD-HFL: Byzantine-resistant Decentralized Hierarchical Federated Learning

***Abstract*—Hierarchical federated learning (HFL) has attracted academic attention to improve the efficiency of federated learning (FL) in real-world applications, however, little research has been done to explore the structural advantages of HFL against Byzantine attacks and to investigate how to make HFL immune to top-level server Single Point of Failure (SPOF). To explore this field and improve the robustness of HFL, we propose a novel generalized paradigm ABD-HFL for asynchronous Byzantine-resistant decentralized hierarchical federated learning, a multi-tier structure without a central server for FL tasks with a large number of devices. Based on the layered structure, an innovative universal Byzantine resistance mechanism is designed in ABD-HFL, which enables it to apply a combination of multiple Byzantine robust techniques, making ABD-HFL more powerful than any single application of such techniques. Besides, ABD-HFL is a fully decentralized HFL, there is no central server, but rather multiple nodes at the top level agree on the global model where malicious model updates are excluded. A new concept of pipeline learning workflow is also introduced to study communication efficiency in ABD-HFL, which is based on asynchronous communication between various levels to train and propagate the global model. Our numerical evaluation validates the advantage of ABD-HFL in terms of robustness and communication efficiency.**

***Index Terms*—Hierarchical federated learning, distributed computing, Byzantine resistance, hierarchical networks, decentralized machine learning**

I. Introduction

Federated learning(FL) has emerged as a promising machine learning(ML) scheme since it was firstly introduced by Google [1], [2], [3]. According to the original FedAvg approach [4], FL task is solved by multiple global rounds, each round consisting of two steps. In the first step, participating devices(referred as clients) train a global model with their own dataset via stochastic gradient descent(SGD). In the second step, a server collects these local models and aggregates them into a new global model, which is sent back to clients for the next round of learning.

There are three different communication schemes in FL: synchronous, asynchronous, and semi-synchronous [5]. Under synchronous protocol, the above two steps in each round must be performed alternatively in sequence, while in asynchronous FL, these two steps of adjacent rounds can be executed at the same time. The semi-synchronous FL is a compromise between synchronous and asynchronous protocols.

The traditional FL learning paradigm can be viewed as a star topology which faces many challenges.

- *SPOF* Relying on a server to aggregate global models makes FL vulnerable to Single Point of Failure(SPOF).
- *Scalability* The central server is prone to become the bottleneck in terms of bandwidth and workload.
- *Efficiency* FL efficiency can be severely slowed down by stragglers in unreliable communication channels.
- *Security* Byzantine clients may risk FL by uploading malicious local model updates to the server.

A. Hierarchical Federated Learning(HFL)

In order to solve the problem of star topology, several new FL paradigms have been proposed, such as tree topology, gossip topology, mesh topology, etc. [5]. Among these paradigms, tree-structured hierarchical federated learning(HFL) has attracted plenty of attention in recent research. In [6], the authors describe HFL as an aggregation tree consisting of a global model aggregator and multiple partly model aggregators, overcoming the limitations of a single central aggregator.

Many HFL solutions assume a central server to form the global model, which still exposes the learning process to risks such as SPOF, Byzantine attacks, etc., shown in II-A. Although some authors [7] have mentioned it, there has been little research in academia on a fully decentralized hierarchical FL paradigm that prevents HFL from relying on any central server for global model aggregation.

In addition, the great potential advantages of HFL against Byzantine attacks have been overlooked when many researches focus on improving the efficiency of communication for HFL in the learning process, this is one of the main points to be discussed in this paper.

B. Summary of Contributions

In order to improve the above deficiency in HFL, we propose a novel Asynchronous Byzantine-resistant Decentralized Hierarchical FL framework(ABD-HFL). ABD-HFL is a *general* structure of *fully decentralized* HFL paradigm with an *innovative mechanism* that exploits the advantages of a hierarchical architecture in filtering malicious participants against Byzantine attacks, while maintaining the communication efficiency brought by the layered structure. Our contributions are summarized as follows:

We design a generic algorithm to protect HFL from Byzantine attacks through layer-by-layer filtering, which allows model aggregation at different levels using different types of approaches, such as Byzantine-robust aggregation techniques and distributed consensus mechanisms.

We further decentralize the HFL structure by eliminating a single central server from the tree structure, making the model

more robust in terms of SPOF and the risk of privacy leakage caused by attacking central server.

We also introduce the concept of *pipeline learning workflow*, based on what a formal pattern is developed to study communication efficiency by exploring potential pipelines between local model training and model aggregation procedures.

Our numerical simulations validate our theoretical innovations of ABD-HFL in Byzantine robustness, which has great potential for real-world applications, especially when the learning process involves high percentage of adversaries.

Paper Organization Section II summarizes the latest relevant research on HFL and Byzantine resistance methods in FL. Section III presents the algorithms and mechanisms of ABD-HFL, including pipeline learning workflows. The Byzantine-robust design in ABD-HFL are detailed in section IV, which is followed by the formal expression of ABD-HFL Byzantine tolerance capability with the proof of correctness. Section V shows the results of our simulation on real-world datasets.

II. Related Work

A. Hierarchical Federated Learning(HFL)

Compared with the traditional star topology FL, the main difference of HFL is the introduction of intermediate layers between clients and the central server. Recent research work on HFL has put efforts to explore mechanisms to improve FL in terms of communication efficiency and security.

Some authors have proposed novel computational paradigms based on HFL to ameliorate the problem of low communication efficiency, such as introducing an asynchronous federated optimization algorithm called FedAsync to counter the adverse effects by stragglers in the model aggregation phase [8]. In order to reduce unexpected waiting, SHFL [9] sets a time limit for the edge server to collect local models and send them to the cloud server for global aggregation. Async-HFL [10] and AHFL [11] are asynchronous HFL framework that uses a staleness-aware weight aggregation algorithm in model aggregation to handle stale new updates so that aggregation can proceed continuously without waiting. DFL [12] focuses on the client local training side in which a linear local-global model combiner scheme is adopted to keep clients training without waiting. Some authors have explored decentralized model aggregation, where edge servers aggregate models whenever they receive models from neighbors instead of waiting for stragglers or failed neighbors, such as DHA-FL [13]. HED-FL [14] takes a mechanism that requires multiple rounds of local model exchange between the bottom and middle layers before global aggregation is required from the middle level.

Another direction is to incorporate device-to-device(D2D) communications into HFL structures, which aims to improve communication efficiency by reducing the channel overload. MH-FL [7] belongs to such type which proposes a novel mechanism for edge-level local model aggregation that leverages D2D communication between clients of the same cluster to reach consensus on the aggregated local model before the local models are uploaded to the server. FL-EOCD scheme [15] and TT-HF [16] adopt the similar mechanism. However, the main drawback of this model is that the aggregation procedure is too complex to be implemented in reality.

Some authors have extended the research of HFL settings to delay-sensitive communication networks such as incoming 6G [17]. While some others have studied how to apply HFL structures to improve FL accuracy in the case of non-IID data distribution, such as the dynamic client clustering technology introduced in FL+HC [18]

There are also works to study HFL in Byzantine settings, such as in [19] the authors propose to group benign clients in a largest cluster while adversarial in other smaller groups based on cosine similarity of model updates, the server aggregates global model for each cluster separately.

However, most of these HFL solutions are based on the structure of wireless networks, assuming a central server to form the global model and intermediate local model aggregation through fixed nodes. This exposes the learning process in FHL to various risks similar to traditional FL, such as SPOF, Byzantine attacks, etc.

TABLE I
Byzantine Attack Types

Target	Attack Approaches	Explanation
Training datasets	Label flipping	Flip training sample labels
	Noise	Add noise to training samples
	Backdoor trigger	Inject a trigger into specific area of datasets
Model updates	Noise	Add random noise to model parameters
	Sign Flip(SF)	Flip the sign of the gradient
	A Little is Enough(ALE)	Add noise carefully to pretend to be benign model updates
	Inner Product Manipulation(IPM)	Manipulate inner product of true mean of model updates and output of aggregation schemes

B. Byzantine Attacks and Resistant Approaches in FL

As a distributed learning paradigm, FL faces severe threats from Byzantine attacks by malicious learning participants [20]. It has been proved that model aggregation in FL based on linear combination cannot tolerate even a single Byzantine participant in distributed implementation of current approaches based on SGD [21].

Some authors [22] classified malicious attacks in FL settings into two types, that is by manipulating training datasets or model parameters. Approaches of manipulating device datasets are also known as data poisoning attacks [23]. Manipulating model parameters is a way where the attacker contaminates the local model parameters, thereby adversely affecting the aggregation of global model. Malicious servers pose a greater threat to FL because the server may use the FL tasks to build malicious tasks, causing harmful consequences to FL target users [23]. Based on the work [24] which summarizes the main Byzantine attacks that threaten FL, we classify them according to the manipulation type, as shown in Table I.

TABLE II
TYPE OF BYZANTINE RESISTANCE APPROACHES

Type	Strategy	Approaches
Byzantine robust aggregation	Euclidean distance	Krum, Geomed, AutoGM
	Mean value	TrimmedMean, CC, Clustering
	Median	GeoMed, AutoGM, Median
	Cosine similarity	Clustering
	Clipping	CC
Consensus mechanism	Scalar consensus	PoW, PoS, PBFT, Committee
	Multidimensional consensus	(ϵ, p)-relaxed BVC, validated Byzantine asynchronous ϵ-agreement protocol

To counter the Byzantine attacks discussed above, various methods have been proposed, which can be divided into two type, as shown in Table II.

One approach is to build a Byzantine-robust aggregation schemes by exploring mathematical measurement of parameter vectors to detect and rule out malicious model updates in model aggregation. The authors of [24] summarize these schemes as measuring model parameter vectors by Euclidean-distance, geometric median, mean, cosine similarity, and clipping. Some famous proposals for such solutions include Krum [21], GeoMed [25], Median and Trimmed Mean [26] etc.

A shortcoming of these Byzantine-robust approaches is that they are normally discussed on traditional FL with a central server as a model aggregation, without considering a fully decentralized global model aggregation mechanism to reduce risks such as SPOF or attacks on the central server.

Another approach to explore Byzantine-resistant FL is to exploit the distributed optimization characteristics of FL, based on distributed consensus protocols. In this direction, there are several different consensus building technologies.

One is to explore the traditional scalar consensus protocols to filter malicious model updates, such as using PoW [27], PoS [28], PBFT [29], Committee based consensus [30] etc., this has been extensively studied in the blockchain-based FL paradigm [31], [32]. The basic paradigm of this technology is that before the aggregation procedure, each member of the consensus group use a scalar value(e.g. a score or vote) to express its opinion on a relevant model update, and then exchanges its value with all other members, so that after enough communication rounds, all members of the group reach an agreement on the scalar value which decides to accept or discard this local model update. In this way, malicious or illegal model updates are excluded from the aggregation stage.

Another technology is to investigate multidimensional consensus algorithms, such as multidimensional(or vector) agreement [33] [34]. In this paradigm, multiple aggregators form a consensus group, parameter vectors are collected by each aggregator and exchanged to all other aggregators, and group members agree on a safe area that excludes malicious model updates. Multidimensional agreements usually requires an exponential computations in each round. However, some more efficient multi-dimensional protocol algorithms have been proposed, such as (ϵ, p)-relaxed BVC [35], validated Byzantine asynchronous ϵ-agreement protocol [36], which reaches polynomial computational complexity.

Furthermore, some authors have proposed variants of consensus protocols in Byzantine distributed learning, such as averaging agreement [37] which builds a fully decentralized asynchronous learning paradigm. Kardam [38] and BYZSGD [39] are methods that use Lipschitzness of cost function to filter Byzantine nodes. MoNNA [40] belongs to the collaborative learning algorithms which use momentum of stochastic gradients to eliminate malicious model updates.

All of the above distributed proposals do not consider the network layer structure of real-world FL participant organizations, which can leverage the potential functionality of edge servers in HFL to help filter malicious model updates. This is also where the proposal to secure FL based on blockchain in the second approach falls short.

In summary, Byzantine robust aggregation techniques generally require low computational and communication overhead, but normally each type of method is particularly effective against some types of Byzantine attacks. For distributed consensus methods, they impose heavy communication costs on the system although they can be applied to more general Byzantine attacks.

III. ABD-HFL DESIGN

Inspired by leaf-only trees in LOT [41] and Rcanopus [42], ABD-HFL architecture is a collection of tree structures derived upwards from leaves. All nodes above the bottom level are leader nodes of adjacent lower level. To simply complex intersections between various factors, the following assumptions are made in ABD-HFL.

- *Assumption 1* The communication channel are assumed to be partial synchronous communication, for all nodes the message delivery time is arbitrary, finite but unbounded.
- *Assumption 2* For each level, there are always enough clusters to upload partial models to their upper level at any global training round.
- *Assumption 3* Nodes can join or leave the existing clusters, but no clusters will be split or combined.

A. ABD-HFL Network Architecture

In ABD-HFL, all participating end devices, except nodes from job publishers, will be considered equal and initially located at the bottom level. We use $\mathcal{L}$ to denote the set of

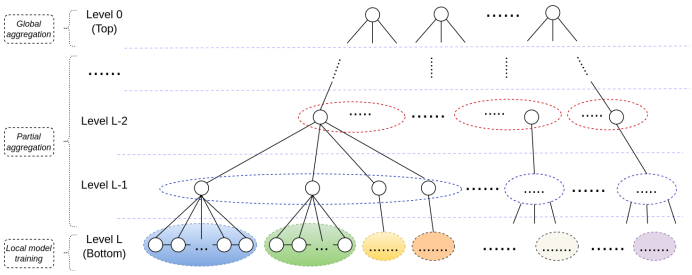

Fig. 1. Architecture of ABD-HFL.

levels, L_0 and L_L represent the top level and the bottom level respectively, given that there are $L+1$ levels of the trees in total. Note that $\mathcal{L} = \{\ell | 0 \leq \ell \leq L\}$ and $|\mathcal{L}| = L+1$. All participating nodes on the bottom level will form an arbitrary number of local learning clusters. The set of clusters at level L_ℓ is denoted as $\mathcal{C}_\ell$, $0 \leq \ell \leq L$, and the set of nodes of the i^{th} cluster at this level is denoted as $\mathcal{C}_{\ell,i}$ with $0 \leq i < |\mathcal{C}_\ell|$. The number of the clusters at level L_ℓ is C_ℓ, and the number of nodes at cluster $\mathcal{C}_{\ell,i}$ is $C_{\ell,i}$, obviously we have

$$C_\ell = |\mathcal{C}_\ell|, \quad C_{\ell,i} = |\mathcal{C}_{\ell,i}|$$

Starting from the bottom level, a leader $\mathcal{A}_{\ell,i}$ of the cluster $\mathcal{C}_{\ell,i}$ will be responsible for the collection and aggregation of local models uploaded by members of $\mathcal{C}_{\ell,i}$, as well as communication with adjacent upper/lower levels. Leader $\mathcal{A}_{\ell,i}$ will also upload the new partial aggregated model to its upper level $L_{\ell-1}$ if $\ell > 0$, and will disseminate models received from its upper level to its cluster members. The leader of each cluster in the L_ℓ level will form the upper level $L_{\ell-1}$. Similarly, nodes at level $L_{\ell-1}$ should be clustered together by geographical similarity to reduce the number of partial aggregated model transfers. This logic applies from the bottom to the top level L_0. At the top level, there is no central server, but are a set of nodes which form one cluster $\mathcal{C}_{0,0}$. The leader of cluster $\mathcal{C}_{\ell,i}$ at level L_ℓ is also a member of a j^{th} cluster $\mathcal{C}_{\ell-1,j}$ at its upper level $L_{\ell-1}$, as shown in Fig. 1. The notations of this paper is summarized in Appendix A.

The total number of levels $|\mathcal{L}|$ will be determined by the magnitude of the total number of participants. Besides, job publishers should contribute their own nodes which will be located at top level, and whose role is to participate in the aggregation of the global model. Through this way, the global model will be shared and stored by all nodes at top level, eliminating the risk of SPOF.

B. Flag level mechanism in ABD-HFL

Flag level In ABD-HFL, after uploading its local model updates, the local model trainer does not need to wait for a new global model which takes time to be formed, but only waits for a *flag partial model* from a *flag level* that is enough for the trainer to effectively start a new global round of training, which are defined as follows.

Definition 1: The *flag level* of ABD-HFL is the level at which the node propagates its *partial* aggregated model to its descendants for next global round of local model training, and this partial aggregated model is a *flag partial model*, or simply *flag model*.

At any round r, the global model aggregation process is being processed to produce global model $\boldsymbol{\theta}_{\mathbf{G}}^{(r)}$, while a new global training round is taking place in local trainers with the flag partial model $\boldsymbol{\theta}_{\mathbf{F}}^{(r)}$. Whenever global model aggregation is completed, $\boldsymbol{\theta}_{\mathbf{G}}^{(r)}$ will be passed to local trainers, and clients will adjusted their trained local model by a *correction factor* which is defined as follows. The selection of this partial model $\boldsymbol{\theta}_{\mathbf{F}}^{(r)}$ will be discussed in section III-D2.

Correction factor The global model $\boldsymbol{\theta}_{\mathbf{G}}^{(r)}$ will arrive after the device has processed the next round of training with flag model $\boldsymbol{\theta}_{\mathbf{F}}^{(r)}$ for some local iterations, making $\boldsymbol{\theta}_{\mathbf{G}}^{(r)}$ appears "stale" compared to the current local model. Flag models are aggregated from part of entire local updates trained on smaller datasets, compared to "outdated" global models, which are aggregated from the all effective local updates based on the entire dataset. Therefore, the current local model based on flag partial model may lead to over fitting to its own datasets. To avoid this potential risk, it is necessary to merge its current local model with the new arrived global model, so that the subsequent local training iterations will be performed on the updated local model.

Inspired by the Linear Local-Global Model Combiner proposed by [12], we adopt a similar linear approach to merge local model with arrived global model. Assume that device n within $\mathcal{C}_{L,i}$ receives a global model $\boldsymbol{\theta}_{\mathbf{G}}^{(r)}$ while n is training its local model $\boldsymbol{\theta}_n^{(r)}$, this local model is derived from flag partial model $\boldsymbol{\theta}_{\mathbf{F}}^{(r)}$, then the updated local model parameters $\boldsymbol{\theta'}_n^{(r)}$ is obtained by:

$$\boldsymbol{\theta'}_n^{(r)} = \alpha_{L,i}^{(r)} * \boldsymbol{\theta}_{\mathbf{G}}^{(r)} + (1 - \alpha_{L,i}^{(r)}) * \boldsymbol{\theta}_n^{(r)} \tag{1}$$

$\alpha_{L,i}^{(r)}$ is the *correction factor* for cluster $\mathcal{C}_{L,i}$ at round r, with $\alpha_{L,i}^{(r)} \in (0, 1]$. The value of correction factor is determined by the following factors

- *Global model latency* The latency of global model determines its staleness compared to the current local model. If the delay is large, the correction factor should be assigned a smaller value to penalize the global model for outdated information. If the delay is small, a larger value should be assigned to the correction factor to improve the generalization ability of the updated local model.
- *Relative datasets size of $\boldsymbol{\theta}_{\mathbf{F}}^{(r)}$ to $\boldsymbol{\theta}_{\mathbf{G}}^{(r)}$* Compared to the global datasets, the relative size of the total datasets on which the flag partial model $\boldsymbol{\theta}_{\mathbf{F}}^{(r)}$ is trained also affects the value of correction factor. The larger the relative size, the more representative of $\boldsymbol{\theta}_{\mathbf{F}}^{(r)}$ is in the aggregation of $\boldsymbol{\theta}_{\mathbf{G}}^{(r)}$, which implies the smaller the difference between $\boldsymbol{\theta}_{\mathbf{G}}^{(r)}$ and $\boldsymbol{\theta}_{\mathbf{F}}^{(r)}$. Therefore, the less information the staled $\boldsymbol{\theta}_{\mathbf{G}}^{(r)}$ could bring to current local model, the smaller the correction factor should be for member of cluster $\mathcal{C}_{L,i}$. And if the relative dataset size of $\boldsymbol{\theta}_{\mathbf{F}}^{(r)}$ is small, it means that $\boldsymbol{\theta}_{\mathbf{G}}^{(r)}$ can bring more new information to the local model, then the correction factor should be larger for members in $\mathcal{C}_{L,i}$.

C. ABD-HFL Algorithm

In ABD-HFL, the learning process consists of $\mathcal{R}$ global rounds. In each round, there are three procedures, local model training at the bottom level, partial model aggregation which ranges from bottom level to L_1, and global model aggregation that is executed at top level L_0, shown in Algorithm 1.

Local model training Algorithm 2 depicts the process of local model training in ABD-HFL. In global round r, $0 < r \leq \mathcal{R}$, all bottom-level devices locally start training their models

Algorithm 1 ABD-HFL Algorithm

Require: $\mathcal{L} = \{\ell | 0 \leq \ell \leq L\}$

Initialize $\boldsymbol{\theta}_G^{(0)}$ to all nodes.
for $\ell \in set\{L\}$ **do**
  LocalModelTraining
end for
for all $\ell \in set\{l : 1 \leq l \leq L\}$ **do**
  PartialModelAggregation
end for
for $\ell \in set\{0\}$ **do**
  GlobalModelAggregation
end for

Algorithm 2 LocalModelTraining

Require: $\{\mathcal{A}_{L,i}\}$, $\{\mathcal{C}_{L,i}\}$, $\mathcal{R}$, $\mathcal{T}$, $\left\{\alpha_{L,i}^{(r)}\right\}$, with $0 \leq r < \mathcal{R}$, $0 \leq i < |\mathcal{C}_L|$

$r \leftarrow 0$
while $r < \mathcal{R}$ **do**
  for all $\mathcal{C}_{L,i}$ at bottom level L **do**
    for all device $n \in \mathcal{C}_{L,i}$ **do**
      if $r = 0$ **then**
        $\boldsymbol{\theta}_n^{(r),0} \leftarrow \boldsymbol{\theta}_{\mathbf{G}}^{(0)}$
      else
        repeat
          Waits for messages
        until receives flag model $\boldsymbol{\theta}_{\mathbf{F}}^{(r)}$ from $\mathcal{A}_{L,i}$
        $\boldsymbol{\theta}_n^{(r),0} \leftarrow \boldsymbol{\theta}_{\mathbf{F}}^{(r)}$
      end if
      $t \leftarrow 0$
      while local iteration $t < \mathcal{T}$ **do**
        SGD updates as:
        $\boldsymbol{\theta}_n^{(r),t+1} \leftarrow \boldsymbol{\theta}_n^{(r),t} - \eta_n^{(r),t} \nabla \ell(\boldsymbol{\theta}_n^{(r),t}; \mathcal{D}_n)$
        if receives $\boldsymbol{\theta}_{\mathbf{G}}^{(r)}$ from $\mathcal{A}_{L,i}$ **then**
          $\boldsymbol{\theta}_n^{(r),t} \leftarrow \alpha_{L,i}^{(r)} \times \boldsymbol{\theta}_{\mathbf{G}}^{(r)} + (1 - \alpha_{L,i}^{(r)}) * \boldsymbol{\theta}_n^{(r),t}$
        end if
        $t \leftarrow t + 1$
      end while
      $\boldsymbol{\theta}_n^{(r)} \leftarrow \boldsymbol{\theta}_n^{(r),t+1}$
      Sends $\boldsymbol{\theta}_n^{(r)}$ to $\mathcal{A}_{L,i}$
    end for
  end for
  $r \leftarrow r + 1$
end while

with an initial flag model $\boldsymbol{\theta}_{\mathbf{F}}^{(r)}$ where r represents the number of round. Consider any bottom-level device n which belongs to cluster $\mathcal{C}_{L,i}$. Device n receives model $\boldsymbol{\theta}_{\mathbf{F}}^{(r)}$ from its cluster leader $\mathcal{A}_{L,i}$, then n follow Stochastic Gradient Decent(SGD) [43] to train the model and complete a predefined local training iterations $\mathcal{T}$. On receiving the global model $\boldsymbol{\theta}_{\mathbf{G}}^{(r+1)}$ during local training process, n will merge it with their own trained local models following Equation 1. After completing iterations $\mathcal{T}$, n uploads its trained model parameters $\boldsymbol{\theta}_n^{(r)}$ to its leader $\mathcal{A}_{L,i}$, and then waits to receive feedback from leader.

Parial model aggregation Partial model aggregation procedure is executed across all intermediate levels, defined in Algorithm 3. Within the process, the communication flow starts from bottom level L_L to its upper level L_{L-1}, then goes upper along intermediate levels, such as from L_ℓ to $L_{\ell-1}$, $0 < \ell \leq L$.

Algorithm 3 PartialModelAggregation

Require: $\{\mathcal{A}_{\ell,i}\}$, $\{\mathcal{N}_{\ell,i,j} | \mathcal{N}_{\ell,i,j} \in \mathcal{C}_{\ell,i}\}$, $\ell_{\mathbf{F}}$, $\mathbf{BRA}$, $\mathbf{CBA}$, with $0 < \ell \leq L$, $0 \leq i < |\mathcal{C}_\ell|$

$r \leftarrow 0$
while $r < \mathcal{R}$ **do**
  for all $\mathcal{C}_{\ell,i}$ at level ℓ **do**
    for all $\mathcal{N}_{\ell,i,j} \in \mathcal{C}_{\ell,i}$ **do**
      if $\mathbf{BRA}$ is adopted at l **then**
        if $\mathcal{N}_{\ell,i,j} = \mathcal{A}_{\ell,i}$ **then**
          $\boldsymbol{\theta}_{\ell,i}^{(r)} \leftarrow$ AggMdlByCluster($\mathcal{C}_{\ell,i}, \mathbf{BRA}$)
          Broadcasts $\boldsymbol{\theta}_{\ell,i}^{(r)}$ to all members of $\mathcal{C}_{\ell,i}$
        else
          Sends $\boldsymbol{\theta}_{\ell,i,j}^{(r)}$ to $\mathcal{A}_{\ell,i}$ {$\boldsymbol{\theta}_{\ell,i,j}^{(r)}$ is the partial/local model of $\mathcal{N}_{\ell,i,j}$}
        end if
      else if $\mathbf{CBA}$ is adopted at l **then**
        $\boldsymbol{\theta}_{\ell,i}^{(r)} \leftarrow \mathbf{CBA}\left(\boldsymbol{\theta}_{\ell,i,k}^{(r)}; \left\{\boldsymbol{\theta}_{\ell,i,k}^{(r)}; |\mathcal{N}_{\ell,i,k} \in \mathcal{C}_{\ell,i}\right\}\right)$
      end if
    end for
    $\boldsymbol{\theta}_{\ell-1,p,q}^{(r)} \leftarrow \boldsymbol{\theta}_{\ell,i}^{(r)}$
    $\mathcal{A}_{\ell,i}$ stores $\boldsymbol{\theta}_{\ell-1,p,q}^{(r)}$ {$\mathcal{A}_{\ell,i} = \mathcal{N}_{\ell-1,p,q}$}
    if $\ell = \ell_{\mathbf{F}}$ **then**
      for all $\mathcal{N}_{\ell,i,j} \in \mathcal{C}_{\ell,i}$ **do**
        $\boldsymbol{\theta}_{\boldsymbol{F},\ell,i}^{(r+1)} \leftarrow \boldsymbol{\theta}_{\ell,i}^{(r)}$
        $\mathcal{N}_{\ell,i,j}$ broadcasts Flag Model as DisseminateModel ($\boldsymbol{\theta}_{\boldsymbol{F},\ell,i}^{(r+1)}$)
      end for
    end if
  end for
  $r \leftarrow r + 1$
end while

ABD-HFL provides users the mechanism to set different partial model aggregation approaches for different intermediate levels. In Algorithm 3, $\mathbf{BRA}$ stands for Byzantine-robust aggregation approaches, and $\mathbf{CBA}$ means Consensus mechanism based aggregation approaches, as shown in II-B.

Typically, assume at any intermediate level ℓ, in the setting of $\mathbf{BRA}$, leader $\mathcal{A}_{\ell,i}$ of $\mathcal{C}_{\ell,i}$ is responsible to receive partial models to form a *partial aggregated model* $\boldsymbol{\theta}_{\ell,i}^{(r)}$, as shown in Algorithm 4. After $\boldsymbol{\theta}_{\ell,i}^{(r)}$ is formed, $\mathcal{A}_{\ell,i}$ will also send a copy of it to all members of $\mathcal{C}_{\ell,i}$ for storage.

The minimum number of partial models that should be collected for aggregation is defined by a percentage of the cluster size, ϕ_ℓ. Then the corresponding number $C_{\ell,i} * \phi_\ell$ should be received by leaders will vary given that the cluster sizes differ

Algorithm 4 AggMdlByCluster

Require: $\mathcal{C}_{\ell,i}$, $\mathcal{A}_{\ell,i}$, ϕ_ℓ,$\mathbf{AG}$(Aggregation Approach)

$M_{\ell,i} \leftarrow 0$
repeat
  Waits for messages
  if $\mathcal{A}_{\ell,i}$ receives partial(or local) model $\boldsymbol{\theta}_{\ell,i,k}^{(r)}$ from n_k **and** $n_k \in \mathcal{C}_{\ell,i}$ **then**
    $M_{\ell,i} \leftarrow M_{\ell,i} + 1$
  end if
until $M_{\ell,i} \geq \phi_\ell \times C_{\ell,i}$ **or** Timeout
Aggregate partial model for $\mathcal{C}_{\ell,i}$ as $\boldsymbol{\theta}_{\ell,i}^{(r)} \leftarrow \mathbf{AG}\left(\boldsymbol{\theta}_{\ell,i,k}^{(r)}; \left\{\boldsymbol{\theta}_{\ell,i,k}^{(r)} | n_k \in \mathcal{C}_{\ell,i}\right\}\right)$
return $\boldsymbol{\theta}_{\ell,i}^{(r)}$

from each other. Assuming that the leader $\mathcal{A}_{\ell,i}$ belongs to the cluster $\mathcal{C}_{\ell-1,p}$ in upper level, once the *partial aggregation* is completed, it immediately uploads the aggregated model to its leader $\mathcal{A}_{\ell-1,p}$ of cluster $\mathcal{C}_{\ell-1,p}$. Similarly, when enough partial aggregated models have been collected by node $\mathcal{A}_{\ell-1,p}$, it will aggregate a new partial model $\boldsymbol{\theta}_{\ell-1,p}^{(r)}$ and upload it to its own cluster leader in level $L_{\ell-2}$. This process goes up to level L_1.

In the setting of $\mathbf{CBA}$ for ℓ, nodes in cluster $\mathcal{C}_{\ell,i}$ will reach an agreement on partial aggregated model $\boldsymbol{\theta}_{\ell,i}^{(r)}$ by the process of consensus mechanism. By this way, all nodes of $\mathcal{C}_{\ell,i}$ including leader $\mathcal{A}_{\ell,i}$ will get a copy of $\boldsymbol{\theta}_{\ell,i}^{(r)}$, then $\mathcal{A}_{\ell,i}$ can send $\boldsymbol{\theta}_{\ell,i}^{(r)}$ to its leader $\mathcal{A}_{\ell-1,p}$, which will be used in the process of partial model aggregation at level $L_{\ell-1}$.

Algorithm 5 DisseminateModel

Require: $\boldsymbol{\theta}_{\mathbf{M}}^{(r)}$

if disseminate $\boldsymbol{\theta}_{\mathbf{F}}^{(r)}$ **then**
  $\ell_s \leftarrow \ell_{\mathbf{F}}$
  $\boldsymbol{\theta}_{\mathbf{M}}^{(r)} \leftarrow \boldsymbol{\theta}_{\mathbf{F}}^{(r)}$
else if disseminate $\boldsymbol{\theta}_{\mathbf{G}}^{(r)}$ **then**
  $\ell_s \leftarrow 0$
  $\boldsymbol{\theta}_{\mathbf{M}}^{(r)} \leftarrow \boldsymbol{\theta}_{\mathbf{G}}^{(r)}$
end if
for all $\ell \in set\{l : \ell_s \leq l < L\}$ **do**
  for all $\mathcal{N}_{\ell,i,j} \in \mathcal{C}_{\ell,i}$ **do**
    if $\ell = \ell_{\mathbf{s}}$ **then**
      $\mathcal{A}_{\ell+1,k}$ Broadcasts $\boldsymbol{\theta}_{\mathbf{M}}^{(r)}$ to all members of $\mathcal{C}_{\ell+1,k}$ $\{\mathcal{A}_{\ell+1,k} = \mathcal{N}_{\ell,i,j}\}$
    else
      repeat
        Waits for messages
      until Receives $\boldsymbol{\theta}_{\mathbf{M}}^{(r)}$ from $\mathcal{A}_{\ell,i}$
      $\mathcal{A}_{\ell+1,k}$ Broadcasts $\boldsymbol{\theta}_{\mathbf{M}}^{(r)}$ to all members of $\mathcal{C}_{\ell+1,k}$
    end if
  end for
end for

When the partial model aggregation process is completed for cluster $\mathcal{C}_{\ell,i}$ of flag level $\ell_{\mathbf{F}}$, all cluster members will disseminate their specific flag model $\boldsymbol{\theta}_{\mathbf{F}}^{(r+1)}$ to all of their lower-level cluster members, which will disseminate the corresponding $\boldsymbol{\theta}_{\mathbf{F}}^{(r+1)}$ to lower levels in the same way, shown as in Algorithm 5. In this way, finally all devices at the bottom level will receive the flag model $\boldsymbol{\theta}_{\mathbf{F}}^{(r+1)}$ from their ancestors at flag level, then they can start the next round of local training.

Global model aggregation Global model is formed through communication flow from the last intermediate level L_1 to top level L_0. Similar to partial model aggregation, global model can be formed by Byzantine-robust approaches or consensus mechanism based approaches. The process is depicted in Algorithm 6.

For the setting of $\mathbf{BRA}$, a leader node $\mathcal{A}_{0,0}$ at the top level will collect partial models from other nodes, then it will aggregate all partial models into global model $\boldsymbol{\theta}_G^{(r+1)}$. The process is similar to partial model aggregation with $\mathbf{BRA}$. Once $\boldsymbol{\theta}_G^{(r+1)}$ is formed, $\mathcal{A}_{0,0}$ broadcasts it to all top-level nodes. On receiving $\boldsymbol{\theta}_G^{(r+1)}$, each top-level node will disseminate the new global model to its lower-level cluster members, which will broadcast $\boldsymbol{\theta}_G^{(r+1)}$ to its own lower level cluster members, this process goes levels downward until all devices at bottom level receive the new global model and used it for the following local iteration.

For the setting of $\mathbf{CBA}$, no leader node exists at the top level, all nodes are equal. They will follow the $\mathbf{CBA}$ consensus process to reach agreement on the new global model $\boldsymbol{\theta}_G^{(r+1)}$, and each node will get a copy of it. Then all nodes will disseminate $\boldsymbol{\theta}_G^{(r+1)}$ to the bottom-level nodes.

Algorithm 6 GlobalModelAggregation

Require: $\mathcal{A}_{0,i}$, $\mathbf{BRA}$, $\mathbf{CBA}$. $\{\mathcal{A}_{0,i}$ - leader of cluster $\mathcal{C}_{1,i}\}$

while $r < \mathcal{R}$ **do**
  for all node $\mathcal{N}_{0,0,i} \in \mathcal{C}_{0,0}$ **do**
    if $\mathbf{BRA}$ is adopted at the top level **then**
      if $\mathcal{A}_{0,0} = \mathcal{N}_{0,0,i}$ **then**
        $\boldsymbol{\theta}_{\mathbf{G}}^{(r+1)} \leftarrow$ AggMdlByCluster($\mathcal{C}_{0,0}, \mathbf{BRA}$)
        $\mathcal{A}_{0,0}$ broadcasts $\boldsymbol{\theta}_{\mathbf{G}}^{(r+1)}$ to all members of $\mathcal{C}_{0,0}$
      else
        Sends $\boldsymbol{\theta}_{0,0,i}^{(r)}$ to $\mathcal{A}_{0,0}$ $\{\boldsymbol{\theta}_{0,0,i}^{(r)}$ is the aggregated partial model of $\mathcal{C}_{1,i}\}$
      end if
    else if $\mathbf{CBA}$ is adopted at the top level **then**
      $\boldsymbol{\theta}_{\mathbf{G}}^{(r+1)} \leftarrow \mathbf{CBA}\left(\boldsymbol{\theta}_{0,0,k}^{(r)}; \left\{\boldsymbol{\theta}_{0,0,k}^{(r)}; |\mathcal{N}_{0,0,k} \in \mathcal{C}_{0,0}\right\}\right)$
    end if
    $\mathcal{N}_{0,0,i}$ stores Global Model $\boldsymbol{\theta}_{\mathbf{G}}^{(r+1)}$
    DisseminateModel ($\boldsymbol{\theta}_{\mathbf{G}}^{(r+1)}$)
  end for
  $r \leftarrow r + 1$
end while

After the formation of $\boldsymbol{\theta}_G^{(r+1)}$, all top-level nodes will disseminate $\boldsymbol{\theta}_G^{(r+1)}$ to devices at bottom level. This process is similar to the dissemination of flag model described above, but the process starts from the L_0 for $\boldsymbol{\theta}_G^{(r+1)}$.

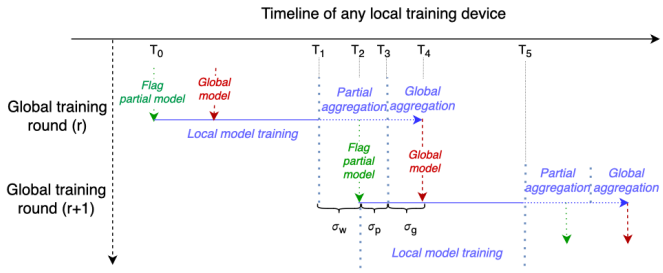

Fig. 2. ABD-HFL Pipeline Learning Workflow.

D. ABD-HFL Pipeline Learning Workflow

The ABD-HFL algorithm is a pipeline learning workflow, where the local model training procedure goes along with the global model aggregation process, making the whole process following an asynchronous communication pattern, as shown in Fig. 2. This effectively shortens the overall learning time by reducing the waiting time between successive global rounds.

There are two types of speed differences that can be exploited to gain concurrency between these tasks. The first is between local model training and model aggregation, and the second is between flag partial model and global model aggregation processes. Note that the second speed difference depends on the scheme chosen discussed in section IV-A.

1) Pipeline workflow efficiency: The efficiency of pipeline workflow can be studied and explored through a quantitative and qualitative analysis.

In any global learning round r, starting from the bottom level, within any cluster $\mathcal{C}_{L,i}$, assume that the leader $\mathcal{A}_{L,i}$ receives the first local model update from node $\mathcal{N}_{L,i,j}$. Then $\mathcal{A}_{L,i}$ waits to receive enough local models to start the aggregation process. The number of models needed is decided by the aggregation scheme described in Algorithm 3. We use τ_L to denote the duration from the time when the first local model is received by $\mathcal{A}_{L,i}$ until the time when $\mathcal{A}_{L,i}$ starts to aggregate its partial model. The duration of partial model aggregation to form $\boldsymbol{\theta}_{L,i}^{(r)}$ is denoted as τ'_L.

A similar partial aggregation process will occur from L_{L-1} to flag level $\ell_{\mathbf{F}}$, which means there will be δ of similar duration to τ_L and τ'_L, $\delta = L - \ell_{\mathbf{F}} + 1$. Assuming that node $\mathcal{N}_{\ell_{\mathbf{F}},j,p}$ at $\ell_{\mathbf{F}}$ is the ancestor of bottom-level node $\mathcal{A}_{L,i}$, whenever it gains flag model $\boldsymbol{\theta}_{\mathbf{F},j,p}^{(r+1)}$, it disseminates $\boldsymbol{\theta}_{\mathbf{F},j,p}^{(r+1)}$ to its descendants, including node $\mathcal{A}_{L,i}$. We ignore the time of model dissemination, then from the time $\mathcal{A}_{L,i}$ collects the first model to the time it receives the corresponding flag partial model, $\mathcal{A}_{L,i}$ and its members in $\mathcal{C}_{L,i}$ waits for a series of time duration:

$$\{\tau_L, \tau'_L, \cdots, \tau_{\ell_{\mathbf{F}}}, \tau'_{\ell_{\mathbf{F}}}\}$$

We use σ_w to represent this series of waiting time, then

$$\sigma_w = \sum_{i=\ell_{\mathbf{F}}}^{L} (\tau_i + \tau'_i)$$

While members of $\mathcal{C}_{L,i}$ starts the local training of global round $(r+1)$ with $\boldsymbol{\theta}_{\mathbf{F},j,p}^{(r+1)}$, the aggregation process from level $\ell_{\mathbf{F}}$ to L_1 is going on at the same time. The sum of the duration taken by these partial aggregation is similar to the duration of σ_w, we denote them as σ_p

$$\sigma_p = \sum_{i=1}^{\ell_{\mathbf{F}}-1} (\tau_i + \tau'_i)$$

Similarly, the global model aggregation at top level consists of two phase of model collection duration τ_g and aggregation duration τ'_g, we use σ_g denote the entire time

$$\sigma_g = \tau_g + \tau'_g$$

Thus, the total time from when the first local model is ready to when the global model arrives is

$$\sigma = \sigma_w + \sigma_p + \sigma_g \tag{2}$$

Within the duration σ_p and σ_g, members in $\mathcal{C}_{L,i}$ are training with $\boldsymbol{\theta}_{\mathbf{F},j,p}^{(r+1)}$ for the next global round, the only waiting time is σ_w.

Based on (2) we can derive the *efficiency indicator* as below, which indicates the computational resources that are effectively utilized by the pipeline workflow instead of being wasted on waiting.

$$\nu = \frac{\sigma_p + \sigma_g}{\sigma} \tag{3}$$

Note that the above efficiency indicator will vary from round to round, because each of the three items σ_w, σ_p, σ_g may be different in each global training round. And in each round, nodes in different cluster may also get different efficiency indicators. This mainly comes from the term σ_w, while the latter two items σ_p and σ_g are same for all nodes in the same round. The difference of σ_w comes from two part. First, for each node within a same cluster, the time to complete the local training iteration is different. Second, for nodes in different clusters, the speed between their ancestors is different at flag level $\ell_{\mathbf{F}}$. Due to these reasons, the formula (3) can be used as a qualitative analysis tool for ABD-HFL. The precise calculation for the effective overall efficiency indicator is a future work which is beyond the scope of this paper.

2) Flag level settings: Formula 3 implies that there is a trade-off between learning efficiency and waiting time in setting which intermediate level as flag level. If the flag level is closer to the top level, meaning that it is less dependence on correction factors defined by Formula 1, and that more useful work has been done before arrival of global model in the new round of training. But the disadvantage of this setting is that bottom-level devices have to wait longer for the arrival of flag partial model. While, if flag partial model is closer to the bottom level, a better communication efficiency can be gained. But in this case the flag partial model may differ significantly from the global model, which means the system relies more on correction factors to correct the staleness of global model. An anlysis of factors that affects the choice of the flag level is discussed in Appendix E.

TABLE III
AGGREGATION SCHEMES IN ABD-HFL

Scheme	Partial Aggregation Phase	Global Aggregation Phase
1	Byzantine-robust aggregation	Consensus mechanism
2	Consensus mechanism	Byzantine-robust aggregation
3	Byzantine-robust aggregation	Byzantine-robust aggregation
4	Consensus mechanism	Consensus mechanism

TABLE IV
APPLICABLE SCENARIOS FOR AGGREGATION SCHEMES IN ABD-HFL

Scheme	Participants Quantity	Robustness	Communication Cost
1	Masses	High	Intermediate
2	Intermediate	High	Intermediate
3	Masses	Intermediate	Low
4	Small	High	High

IV. BYZANTINE RESILIENCE MECHANISM IN ABD-HFL

A. Byzantine settings options

The flexible setting in ABD-HFL algorithm makes it possible to adopt a combination of different Byzantine resistant approaches from the two different domains discussed in II-B. In this way, task publishers can choose the best trade-off between robustness and efficiency according to their own criteria. Based on the two type of approaches, we summarize four possible Byzantine-robust combination for ABD-HFL, shown as in Table III.

Each of these four combination has its own potential applicable scenarios, we summarize the features of each of these combinations.

- *Scheme 1* This scheme deploys Byzantine-robust approaches for partial aggregation and consensus-based methods for global aggregation, which is suitable for FL with mass devices. With this setting, partial aggregation can filter out malicious models with lower overall communication cost, whereas on the top level a high quality of global model aggregation is implemented with a higher communication cost.
- *Scheme 2* This scheme fits FL with a relative small number of participants which is more sensible to malicious participants. At the expense of high communication in partial aggregation, consensus-based method can provide a relative high quality of partial models used for global model aggregation. With this guarantee, the top level can safely employ a lightweight Byzantine-robust aggregation for global model aggregation.
- *Scheme 3* With this configuration, Byzantine-robust aggregation are deployed through all levels, which ensures a relative faster aggregation speed than all other schemes, This scheme is suitable for a great amount of participants joining the FL task, and the extent of Byzantine behavior of participants should be controlled by job publisher.
- *Scheme 4* In this scheme consensus-based methods are adopted at all levels. Due to the heavy communication cost of distributed consensus mechanisms, this configuration will be accompanied by high communication cost. But the reward from heavy communication is better robustness on Byzantine attacks.

B. ABD-HFL Byzantine Tolerance Models

In this section we describe the Byzantine tolerance capability of ABD-HFL in terms of the proportion of malicious nodes. Job publishers can decide on the aggregation scheme shown in Table III for a specific FL task. We denote γ_1 as the percentage of maximum Byzantine tolerance of global aggregation at the top level, and denote γ_2 for partial aggregation in all intermediate levels.

First we develop the Equal Cluster Size Model((ECSM) as a basic model to study the property of ABD-HFL. ECSM is an ideal model where we consider all clusters on all levels except the top to be of equal size, i.e. each top node is the root of a complete m-ary tree. To simplify the notation, we use N_t to denote the number of nodes at the top in round r. Due to space limitation, all proofs of the following theorems and corollaries are given in the Appendix B.

In order to study the properties of ABD-HFL tree structure, we define a new tree concept based on m-arry tree.

Definition 2: A *p-ratio two-type complete m-ary tree* is a complete m-ary tree in which all nodes are either type-I or type-II nodes, the root is a type-I node, and the proportion of type-I in the child nodes of a type-I node is p, $0 \leq p \leq 1$, and the child nodes of a type-II node are all type-II.

With this definition, we can develop the properties related to the proportion of each type in each levels.

Theorem 1: For a p-ratio two-type complete m-ary tree of depth L, at the ℓ^{th} level, $0 \leq \ell < L$, there are $(pm)^\ell$ type-I nodes , and the proportion of type-I nodes at this level is p^ℓ.

We study the case of ABD-HFL structures with different Byzantine tolerance capability in the top level and subsequent levels. For convenience of description, we first define this property for ABD-HFL structure as below.

Definition 3: ABD-HFL with property γ_1-γ_2 is a ABD-HFL structure with N_t nodes at the top level, and the *maximum* proportion of Byzantine nodes tolerated in the *top level* is γ_1, the *maximum* proportion for *each cluster* of each subsequent level is γ_2.

Based on Definition 2, we define the concept of a p-ratio ABD-HFL for this analysis.

Definition 4: A *p-ratio ABD-HFL structure of L levels* is an ABD-HFL structure with N_t nodes at top level, and for some top-level nodes, each of them is the root of a p-ratio two-type complete m-ary tree of depth L, assuming that the nodes of type-I are honest nodes, and nodes of type-II are Byzantine nodes; while for other top-level nodes, each of them is the root of a complete m-ary tree of depth L, where all the nodes are Byzantine nodes(type-II).

In fact, the p-ratio ABD-HFL structure is a set of p-ratio two-type complete m-ary tree, with each node at the top level is the root of each such tree respectively.

Note that if all top-level nodes are Byzantine, then all nodes of a p-ratio ABD-HFL structure are Byzantine; if all top-level nodes are honest and $p = 1$, then all nodes are honest.

Firstly we develop an corollary about the number of nodes at each level of the structure.

Corollary 1: For an p-ratio ABD-HFL structure of L levels with N_t nodes at top level, there are a total of $N_t m^\ell$ nodes at level L_ℓ, given that $0 \leq \ell < L$.

Based on Corollary 1, we can develop an important property of ABD-HFL structure regarding the proportion of Byzantine nodes that can be tolerated in each level.

Theorem 2: For a p-ratio ABD-HFL structure of L levels with property γ_1-γ_2, for any level ℓ, $0 \leq \ell < L$, the maximum number of tolerated Byzantine nodes are $N_t m^\ell - (1-\gamma_1)N_t[(1-\gamma_2)m]^\ell$, and the maximum proportion of Byzantine nodes tolerated is $1-(1-\gamma_1)(1-\gamma_2)^\ell$.

Interestingly, if $\gamma_1 = \gamma_2$, according to Theorem 2, the maximum ratio of Byzantine nodes in each level becomes $1-(1-\gamma_1)^\ell$. Based on the above observations, we can further derive several interesting properties of p-ratio ABD-HFL structure with property γ_1-γ_2.

Corollary 2: For a p-ratio ABD-HFL structure with property γ_1-γ_2, a lower level tolerates a greater proportion of Byzantine nodes than its upper level.

Corollary 3: For a p-ratio ABD-HFL structure with property γ_1-γ_2, if the number of bottom-level nodes is fixed, the more levels there are, the greater the proportion of Byzantine nodes that the system can tolerate at bottom level.

Corollary 3 is useful to guide the build of ABD-HFL structure based on ECSM model. It implies that for any task, assuming the number of devices joining the learning task is fixed, increasing the total number of levels in ABD-HFL can make it tolerate higher proportion of Byzantine devices at the bottom level.

ECSM assumes equal cluster sizes. Extending ECSM to clusters of arbitrary size at any level, i.e., the Arbitrary Cluster Size Model(ACSM), is detailed in the Appendix C.

V. Numerical Evaluation

In this section, we present simulation results of an example ABD-HFL structure based on ECSM with real-world image classification task.

A. Experimental Setup

We choose the handwriting recognition data set MNIST [44], following Song et al. [45], we use DNN model for evaluation. The final test accuracy of the global model are compared between ABD-HFL and vanilla FL.

Following McMahan et al. [4], two sets of experiments are made, i.e. IID and non-IID data distribution. We set up two scenarios of data poisoning attacks, Type I and Type II. The former sets all training sample labels to 9, and the latter randomly set sample labels to values between 0 and 9.

The proportion of bottom-level devices with poisoned datasets are ranging from 0% to 65%, where 0% represents the situation where all nodes are honest.

For the Byzantine-resistant aggregation approaches, in ABD-HFL structure we follow scheme 1 mentioned in IV-A. The Byzantine-robust aggregation approaches are set for the partial aggregation. Two partial aggregation methods, Krum (MultiKrum) and Median, under IID and non-IID settings respectively, with the assumed proportion of malicious nodes in Krum's algorithm set to 25%. At the top level, a consensus mechanism inspired by Chen et al [28] is deployed for global model aggregation. Simulation settings are detailed in Appendix D.

Based on the above design, according to Definition 3, the MultiKrum algorithm mentioned above defines $\gamma_1 = 25\%$. We also assume that the consensus mechanism at the top level is able to filter out one Byzantine node from the four top nodes, which implies that $\gamma_2 = 1/4 = 25\%$. The total number of layers is 3, the bottom layer $L = 2$, according to Theorem 2, this ABD-HFL settings can tolerate up to 57.8125% of malicious participants at the bottom level, calculated as follows:

$$\begin{aligned} 1-(1-\gamma_1)(1-\gamma_2)^\ell &= 1-(1-25\%)*(1-25\%)^2 \\ &= 57.8125\% \end{aligned}$$

This inference has been verified by simulations shown below.

B. Simulation Results

The average test accuracy of five repeated runs to evaluate the final global model is shown in table V. Firstly, we observe that when the proportion of malicious nodes is 0% (all nodes are honest), ABD-HFL achieves almost the same prediction accuracy as vanilla FL model regardless of whether the data distribution is IID or non-IID, which verifies that ABD-HFL maintains the learning ability of the classical FL paradigm.

In the IID case of Type I attack, where both models deploy MultiKrum as partial model aggregation approach, with malicious nodes proportion increases from 5% to 57.8%, the accuracy of vanilla FL model drops sharply from approximately 90.3% to 10.1%, while for ABD-HFL model, it almost remains at the same level between 89% to 90%, which verifies the stability of ABD-HFL comparing to traditional centralized FL paradigm under such attack.

Note that even with malicious proportion of 65%, which is above the theoretical upper bound of 57.8%, ABD-HFL still achieves an accuracy of 73.8%, compared to 10.01% for vanilla FL. This sharp contrast does not appear in the IID cases of Type II attack, the reason is that Krum performs very well in this type of attack even with vanilla FL topology, and the similar performance of ABD-HFL to vanilla FL shows that it can keep the Byzantine-resistance capabilities of Krum mechanism deployed within its architecture without compromising it.

It is worth noting that in the case of non-IID simulation, the Median approach is deployed as aggregation in both models. Different from the IID cases where we can develop theoretical Byzantine tolerance boundary for Krum algorithm, it is difficult to give precise boundaries for Median aggregation, so the malicious proportion in non-IID is more of a representation than a precise quantitative characterization.

TABLE V
FINAL TESTING ACCURACY ON GLOBAL MODELS

Data Distribution	Attack Type	Model	Accuracy								
IID	Type I	ABD-HFL	89.9%	90.0%	90.0%	89.9%	90.0%	90.0%	89.9%	89.7%	73.8%
		Vanilla FL	90.0%	90.3%	90.1%	90.5%	89.9%	90.1%	10.1%	10.1%	10.1%
	Type II	ABD-HFL	90.1%	90.0%	90.1%	90.1%	90.3%	90.1%	89.6%	89.7%	88.8%
		Vanilla FL	90.1%	90.1%	90.2%	90.1%	90.1%	90.2%	89.8%	89.9%	89.6%
non-IID	Type I	ABD-HFL	89.5%	78.1%	76.5%	74.6%	68.7%	59.4%	62.7%	59.9%	48.9%
		Vanilla FL	90.0%	74.3%	72.2%	65.2%	57.7%	44.2%	14.7%	10.1%	10.1%
	Type II	ABD-HFL	89.7%	79.7%	75.8%	76.7%	73.6%	66.6%	68.9%	62.9%	47.0%
		Vanilla FL	89.7%	74.6%	76.1%	69.6%	68.1%	62.8%	46.3%	38.6%	26.8%
Malicious Proportion			0%	5.0%	10.0%	20.0%	30.0%	40.0%	50.0%	57.8%	65.0%

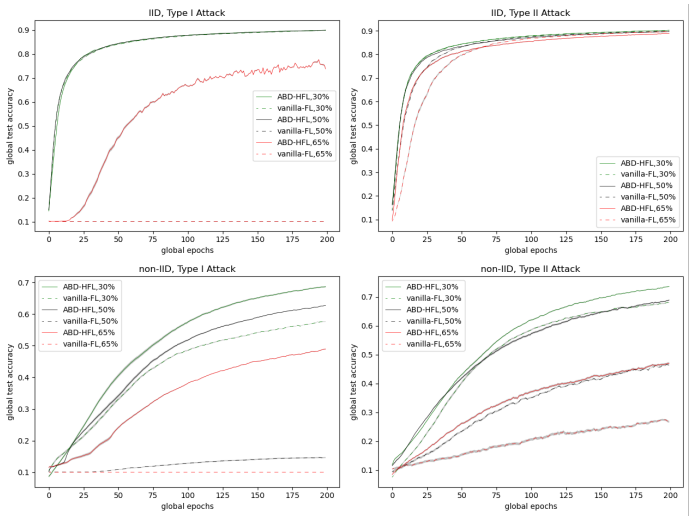

Fig. 3. Data-poisoning attack simulation results.

Nonetheless, the results of non-IID simulation more directly demonstrate the Byzantine fault tolerance advantage of ABD-HFL over centralized topologies. Although the accuracy of ABD-HFL decreases as the number of malicious participants increases, the decrease from approximately 78.1% to 48.9% and 79.7% to 47.0% is much smaller than that of vanilla FL model, which are from 74.3% to 10.1% and 74.6% to 26.8% respectively. The decrease in accuracy for both models comes partly from the Byzantine tolerance capability of Median algorithm and partly from the tougher challenges present in non-IID distribution compared to IID cases. Even so, ABD-HFL can keep a accuracy level above 60% when 50% participants are malicious, while vanilla FL can only achieve an accuracy of 14.7% to 46.3% depending on the attack type.

Figure 3 shows the relative convergence speed between ABD-HFL and vanilla FL in several data poisoning attack scenarios. Each curve consists of a line representing the average accuracy at each global step, with the gray shaded area standing for the confidence intervals based on five runs. We observe that both ABD-HFL or vanilla FL perform stably within five repeated running, except for the case with a malicious proportion of 65%, where both show larger but acceptable variation comparing to other cases.

As for the convergence speed, we can also observe that in all cases, ABD-HFL almost keeps pace with vanilla FL at low malicious proportions, such as below 50% in the IID case, while for the non-IID case, ABD-HFL always achieves much higher test accuracy than vanilla FL with a malicious proportion above 30%, which corresponds to the final accuracy shown in TableV.

VI. CONCLUSION AND OUTLOOK

In this work, we have studied the related technologies to extend traditional centralized FL to more efficient and Byzantine-robust, including various HFL proposals, Byzantine-robust aggregation approaches, and applying distributed consensus mechanism to secure FL. Based on these studies, we detect the risk of relying on a central server in most HFL models, as well as the blank domain where the huge potential functionality of layered architecture in HFL regarding to strengthen the Byzantine-resistance ability in FL has been ignored by recent research. As a result, we propose a novel ABD-HFL arhcitecture for building a decentralized FL platform with an innovative generic Byzantine-robust framework, and a pipeline learning workfow pattern to exploit the efficiency between local training and model aggregation. We also show the theoretical boundary of Byzantine tolerance ability in ABD-HFL, which are verified by numerical experiments.

In future work we hope to further explore optimization possibilities to make ABD-HFL more efficient and robust. To maximum the benefits of pipeline workflow, we need to conduct further research on factors that underlie asynchronous learning process, such as selection of flag level, defining an overall efficiency indicator, comparison of various aggregation schemes, etc.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *ArXiv*, vol. abs/1602.05629, 2016.

[2] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[3] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[5] J. Wu, S. Drew, F. Dong, Z. Zhu, and J. Zhou, "Topology-aware federated learning in edge computing: A comprehensive survey," *arXiv preprint arXiv:2302.02573*, 2023.

[6] O. Rana, T. Spyridopoulos, N. Hudson, M. Baughman, K. Chard, I. Foster, and A. Khan, "Hierarchical and decentralised federated learning," *arXiv preprint arXiv:2304.14982*, 2023.

[7] S. Hosseinalipour, S. S. Azam, C. G. Brinton, N. Michelusi, V. Aggarwal, D. J. Love, and H. Dai, "Multi-stage hybrid federated learning over large-scale d2d-enabled fog networks," *IEEE/ACM transactions on networking*, vol. 30, no. 4, pp. 1569–1584, 2022.

[8] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.

[9] M. G. Herabad, "Communication-efficient semi-synchronous hierarchical federated learning with balanced training in heterogeneous iot edge environments," *Internet of Things*, vol. 21, p. 100642, 2023.

[10] X. Yu, L. Cherkasova, H. Vardhan, Q. Zhao, E. Ekaireb, X. Zhang, A. Mazumdar, and T. Rosing, "Async-hfl: Efficient and robust asynchronous federated learning in hierarchical iot networks," in *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation*, 2023, pp. 236–248.

[11] P. Mitra and S. Ulukus, "Timely asynchronous hierarchical federated learning: Age of convergence," *arXiv preprint arXiv:2306.12400*, 2023.

[12] F. P.-C. Lin, S. Hosseinalipour, C. Brinton, and N. Michelusi, "Delay-aware hierarchical federated learning," *arXiv preprint arXiv:2303.12414*, 2023.

[13] W. H. Huff, R. Balakrishnan, H. Feng, M. Lee, P. Wang, C. Chen *et al.*, "Dha-fl: Enabling efficient and effective aiot via decentralized hierarchical asynchronous federated learning," in *MLSys 2023 Workshop on Resource-Constrained Learning in Wireless Networks*, 2023.

[14] F. De Rango, A. Guerrieri, P. Raimondo, and G. Spezzano, "Hed-fl: A hierarchical, energy efficient, and dynamic approach for edge federated learning," *Pervasive and Mobile Computing*, vol. 92, p. 101804, 2023.

[15] M. S. Al-Abiad, M. Obeed, M. J. Hossain, and A. Chaaban, "Decentralized aggregation for energy-efficient federated learning via overlapped clustering and d2d communications," *arXiv preprint arXiv:2206.02981*, 2022.

[16] F. P.-C. Lin, S. Hosseinalipour, S. S. Azam, C. G. Brinton, and N. Michelusi, "Semi-decentralized federated learning with cooperative d2d local model aggregations," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3851–3869, 2021.

[17] A. Ali and A. Arafa, "Delay sensitive hierarchical federated learning with stochastic local updates," *arXiv preprint arXiv:2302.04851*, 2023.

[18] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–9.

[19] F. Sattler, K.-R. Müller, T. Wiegand, and W. Samek, "On the byzantine robustness of clustered federated learning," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8861–8865.

[20] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *arXiv preprint arXiv:2003.02133*, 2020.

[21] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[22] J. Shi, W. Wan, S. Hu, J. Lu, and L. Y. Zhang, "Challenges and approaches for mitigating byzantine attacks in federated learning," in *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2022, pp. 139–146.

[23] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[24] S. Li, E. C.-H. Ngai, and T. Voigt, "An experimental study of byzantine-robust aggregation schemes in federated learning," *IEEE Transactions on Big Data*, 2023.

[25] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.

[26] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[27] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, "A blockchained federated learning framework for cognitive computing in industry 4.0 networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2964–2973, 2020.

[28] H. Chen, S. A. Asif, J. Park, C.-C. Shen, and M. Bennis, "Robust blockchained federated learning with model validation and proof-of-stake inspired consensus," *arXiv preprint arXiv:2101.03300*, 2021.

[29] Z. Yang, Y. Shi, Y. Zhou, Z. Wang, and K. Yang, "Trustworthy federated learning via blockchain," *IEEE Internet of Things Journal*, vol. 10, no. 1, pp. 92–109, 2022.

[30] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2020.

[31] Y. Qu, M. P. Uddin, C. Gan, Y. Xiang, L. Gao, and J. Yearwood, "Blockchain-enabled federated learning: A survey," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–35, 2022.

[32] A. Qammar, A. Karim, H. Ning, and J. Ding, "Securing federated learning with blockchain: a systematic literature review," *Artificial Intelligence Review*, vol. 56, no. 5, pp. 3951–3985, 2023.

[33] H. Mendes and M. Herlihy, "Multidimensional approximate agreement in byzantine asynchronous systems," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, 2013, pp. 391–400.

[34] N. H. Vaidya and V. K. Garg, "Byzantine vector consensus in complete graphs," in *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, 2013, pp. 65–73.

[35] M. Dotan, G. Stern, and A. Zohar, "Validated byzantine asynchronous multidimensional approximate agreement," *arXiv preprint arXiv:2211.02126*, 2022.

[36] H. Mendes, M. Herlihy, N. Vaidya, and V. K. Garg, "Multidimensional agreement in byzantine systems," *Distributed Computing*, vol. 28, no. 6, pp. 423–441, 2015.

[37] E. M. El-Mhamdi, S. Farhadkhani, R. Guerraoui, A. Guirguis, L.-N. Hoang, and S. Rouault, "Collaborative learning in the jungle (decentralized, byzantine, heterogeneous, asynchronous and nonconvex learning)," *Advances in Neural Information Processing Systems*, vol. 34, pp. 25 044–25 057, 2021.

[38] G. Damaskinos, R. Guerraoui, R. Patra, M. Taziki *et al.*, "Asynchronous byzantine machine learning (the case of sgd)," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1145–1154.

[39] E.-M. El-Mhamdi, R. Guerraoui, A. Guirguis, L. N. Hoang, and S. Rouault, "Genuinely distributed byzantine machine learning," in *Proceedings of the 39th Symposium on Principles of Distributed Computing*, 2020, pp. 355–364.

[40] S. Farhadkhani, R. Guerraoui, N. Gupta, L.-N. Hoang, R. Pinot, and J. Stephan, "Robust collaborative learning with linear gradient overhead," in *International Conference on Machine Learning*. PMLR, 2023, pp. 9761–9813.

[41] A. Allavena, Q. Wang, I. Ilyas, and S. Keshav, "Lot: A robust overlay for distributed range query processing," Technical report, University of Waterloo, Tech. Rep., 2006.

[42] S. Keshav, W. Golab, B. Wong, S. Rizvi, and S. Gorbunov, "Rcanopus: Making canopus resilient to failures and byzantine faults," *arXiv preprint arXiv:1810.09300*, 2018.

[43] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer, 2010, pp. 177–186.

[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[45] R. Song, L. Zhou, V. Lakshminarasimhan, A. Festag, and A. Knoll, "Federated learning framework coping with hierarchical heterogeneity in cooperative its," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 3502–3508.

APPENDIX A
NOTATIONS AND TERMINOLOGIES

TABLE VI
NOTATIONS AND TERMINOLOGIES

Notation	Description
$\mathcal{L}$	Set of levels in ABD-HFL structure
L_ℓ	The ℓ^{th} level
$\ell_{\mathbf{F}}$	Flag level
L_0, L_L	The top level, the bottom level
$\mathcal{R}$	Number of global rounds of federated learning
$\mathcal{T}$	Number of local training iterations
$\boldsymbol{\theta}_n^{(r)}$	Local model parameters of node n in round r
$\mathcal{C}_\ell$	Set of learning clusters at level L_ℓ
$\mathcal{C}_{\ell,i}$	Set of nodes in i^{th} cluster of level L_ℓ
$\mathcal{A}_{\ell,i}$	Leader node of set $\mathcal{C}_{\ell,i}$
$\mathcal{N}$	Set of all nodes
$\mathcal{N}_{\ell,i,j}$	The j^{th} node in cluster $\mathcal{C}_{\ell,i}$
$\boldsymbol{\theta}_{\mathbf{G}}^{(r)}$	Global model of round (r)
$\boldsymbol{\theta}_{\mathbf{F}}^{(r)}$	Partial flag model of round (r)
$\boldsymbol{\theta}_{\boldsymbol{F},\ell,i}^{(r+1)}$	Partial flag model aggregated by cluster $\mathcal{C}_{\ell,i}$ for round $(r+1)$
$\boldsymbol{\theta}_{\ell,i}^{(r)}$	Partial aggregated model of cluster $\mathcal{C}_{\ell,i}$ in round r
$\ell(\boldsymbol{\theta}_n^{(r),t};\mathcal{D}_n)$	Loss function, $\boldsymbol{\theta}_n^{(r),t}$ are parameters at round r, local iteration t
$\alpha_{L,i}^{(r)}$	Correction factor for cluster $\mathcal{C}_{L,i}$ at global round (r)
$\mathcal{D}_n$	Dataset of node n
C_ℓ	Total number of clusters of level L_ℓ
N_ℓ	Total number of nodes of level L_ℓ in global iteration round r
$C_{\ell,i}$	Total number of nodes in cluster $\mathcal{C}_{\ell,i}$
$\mathcal{B}_\ell$ ($\mathcal{H}_\ell$)	Set of Byzantine(honest) clusters at level L_ℓ
H_ℓ (B_ℓ)	Total number of honest nodes (Byzantine nodes) of level L_ℓ
$H_{\ell,i}$ ($B_{\ell,i}$)	Total number of honest nodes (Byzantine nodes) in cluster $\mathcal{C}_{\ell,i}$

APPENDIX B
PROOFS OF ECSM

Before proving the theorems and corollaries of ECSM model, we firstly put lemma which is used in the following proofs.

Lemma 1: For a complete m-ary tree of depth L, there are m^ℓ nodes in any ℓ^{th} level, given that $0 \leq \ell \leq L$.

Proof: We ignore the proof as it is straightforward. ■

Theorem 1: For a p-ratio two-type complete m-ary tree of depth L, at the ℓ^{th} level, $0 \leq \ell < L$, there are $(pm)^\ell$ type-I nodes , and the proportion of type-I nodes at this level is p^ℓ.

Proof: The proof can be accomplished through a level by level induction.

We firstly prove it for the top level, where $\ell = 0$. At the top level, there is only one root node which is type-I, which means that the number of type-I nodes is 1, thus the proportion of type-I nodes is 1. As

$$(p * m)^0 = 1$$
$$p^0 = 1$$

Which proves the correctness of the theorem for the top level.

We then prove it for the the first level. All nodes at L_1 is the children of the root of top level, so the number of type-I nodes at the first level is $1 \times p \times m = pm$, and the number of type-II nodes is $1 \times (1-p) \times m = (1-p)m$. According to the Lemma 1 there are m nodes at level one, so the ratio of type-I nodes at level one is

$$\frac{pm}{m} = p$$

which proves the correctness for the first level.

Suppose the theorem is true at the ℓ^{th} level, where there are m^ℓ nodes in total, and there are $(pm)^\ell$ nodes of type-I. Remember that each node at that level has m children, so at ℓ^{th} level a type-I node has pm nodes of type-I as its children. So the total number of type-I nodes among all the child nodes at ℓ^{th} level is:

$$(pm) \times (pm)^\ell = (pm)^{\ell+1}$$

which is also the total number of type-I nodes at $(\ell+1)^{th}$ level.

Similarly, according to Lemma 1 there are $m^{\ell+1}$ nodes at $(\ell+1)^{th}$ level, thus the ratio of type-I nodes at $(\ell+1)^{th}$ level is

$$\frac{(pm)^{\ell+1}}{m^{\ell+1}} = p^{\ell+1}$$

which completes the proof. ■

Corollary 1: For an p-ratio ABD-HFL structure of L levels with N_t nodes at top level, there are a total of $N_t m^\ell$ nodes at level L_ℓ, given that $0 \leq \ell < L$.

Proof: Firstly we focus on any top node. Since the node is the root of a complete m-ary tree, according to Lemma 1, level L_ℓ has m^ℓ nodes.

There are N_t nodes at top level, so in level L_ℓ, there are a total of $N_t * m^\ell$ nodes, leading to the proof. ■

Theorem 2: For a p-ratio ABD-HFL structure of L levels with property γ_1-γ_2, for any level ℓ, $0 \leq \ell < L$, the maximum number of tolerated Byzantine nodes are $N_t m^\ell - (1-\gamma_1)N_t[(1-\gamma_2)m]^\ell$, and the maximum proportion of Byzantine nodes tolerated is $1-(1-\gamma_1)(1-\gamma_2)^\ell$.

Proof: For convenience, we use B_ℓ and H_ℓ to denote the number of Byzantine nodes and honest nodes at level ℓ, use P_ℓ to denote the proportion of Byzantine nodes in L_ℓ, and use $B_{\mathcal{C}_{\ell,i}}$, $H_{\mathcal{C}_{\ell,i}}$ to denote the number of Byzantine nodes and honest nodes in cluster $\mathcal{C}_{\ell,i}$. We prove it from the top level where $\ell = 0$. According to definition 3, there are N_t nodes at the top level, and the maximum Byzantine tolerance ratio is γ_1, so the number of Byzantine nodes tolerated at top level satisfies:

$$B_0 \leq \gamma_1 N_t$$

Notice that

$$\begin{aligned}\gamma_1 N_t &= N_t m^0 - (1-\gamma_1)N_t[(1-\gamma_2)m]^0 \\ &= N_t - (1-\gamma_1)N_t\end{aligned}$$

As a result, the proportion of Byzantine nodes satisfies:

$$\begin{aligned}P_0 &= \frac{B_0}{N_t} \\ &\leq \frac{\gamma_1 N_t}{N_t} \\ &\leq \gamma_1\end{aligned}$$

Notice that

$$\begin{aligned}\gamma_1 &= 1-(1-\gamma_1)(1-\gamma_2)^0 \\ &= 1-(1-\gamma_1)\end{aligned}$$

Which proves the correctness of the assertion for the top level L_0.

From this we can infer that the number of honest nodes required at top level satisfies:

$$\begin{aligned}H_0 &\geq N_t - \gamma_1 * N_t \\ &\geq N_t(1-\gamma_1)\end{aligned}$$

Then we consider for the first level below the top level, where $\ell = 1$. By definition 4, each top node is the root of a p-ratio two-type complete m-ary tree, and is also the leader node of each cluster of level one. According to the definition 3, for each cluster at level one, the maximum ratio of Byzantine nodes tolerated is γ_2 , which implies that the minimum ratio of honest nodes in each cluster is $(1-\gamma_2)$. By the definition of 2, let $p \geq 1-\gamma_2$, then by theorem 1 the number of honest nodes at any cluster $\mathcal{C}_{1,i}$ of this level will be

$$\begin{aligned}H_{\mathcal{C}_{1,i}} &= (pm)^1 \\ &\geq (1-\gamma_2)m\end{aligned}$$

implying the Byzantine nodes in each cluster at this level can be tolerated is

$$\begin{aligned}B_{\mathcal{C}_{1,i}} &= m - H_{\mathcal{C}_{1,i}} \\ &\leq m-(1-\gamma_2)m \\ &\leq \gamma_2 m\end{aligned}$$

Then considering all clusters at this level, we get the total number of honest nodes at this level satisfies:

$$\begin{aligned} H_1 &= H_0 * H_{\mathcal{C}_{1,i}} \\ &\geq (1-\gamma_1)N_t * (1-\gamma_2)m \\ &\geq (1-\gamma_1)N_t * (1-\gamma_2)m \end{aligned}$$

According to corollary 1 there are $N_t m$ nodes at L_1, that is $N_1 = N_t m$, and by Lemma 1, there are m nodes for each complete m-ary tree, then we get

$$\begin{aligned} B_1 &= N_t m - H_1 \\ &\leq N_t m - N_t(1-\gamma_1)(1-\gamma_2)m \end{aligned}$$

Hence the proportion of Byzantine nodes at L_1 satisfies

$$\begin{aligned} P_1 &= \frac{B_1}{N_1} \\ &\leq \frac{N_t m - N_t(1-\gamma_1)(1-\gamma_2)m}{N_t m} \\ &\leq 1-(1-\gamma_1)(1-\gamma_2) \end{aligned}$$

which completes the prove for level L_1.

Assuming that the theorem holds for level L_ℓ, we show that it also holds for level $L_{\ell+1}$. From corollary 1 we know that there are $N_t m^\ell$ nodes at L_ℓ. Based on the assumption, the number of Byzantine nodes tolerated at L_ℓ satisfies:

$$B_\ell = N_t m^\ell - (1-\gamma_1)N_t[(1-\gamma_2)m]^\ell$$

Then we get

$$\begin{aligned} H_\ell &= N_t m^\ell - B_\ell \\ &= N_t m^\ell - [N_t m^\ell - (1-\gamma_1)N_t[(1-\gamma_2)m]^\ell] \\ &= (1-\gamma_1)N_t[(1-\gamma_2)m]^\ell \end{aligned}$$

Remember that the minimum proportion of honest nodes is $(1-\gamma_2)$, let $p \geq (1-\gamma_2)$, then

$$\begin{aligned} H_{\ell+1} &= H_\ell * p * m \\ &\geq H_\ell * (1-\gamma_2) * m \\ &\geq (1-\gamma_1)N_t[(1-\gamma_2)m]^\ell * (1-\gamma_2) * m \\ &\geq (1-\gamma_1)N_t[(1-\gamma_2)m]^{\ell+1} \end{aligned}$$

There are a total of $N_t m^{\ell+1}$ nodes in level $L_{\ell+1}$, so

$$\begin{aligned} B_{\ell+1} &= N_t m^{\ell+1} - H_{\ell+1} \\ &\leq N_t m^{\ell+1} - (1-\gamma_1)N_t[(1-\gamma_2)m]^{\ell+1} \end{aligned}$$

As a result, the proportion of Byzantine nodes satisfies

$$\begin{aligned} P_{\ell+1} &= \frac{B_{\ell+1}}{N_{\ell+1}} \\ &\leq \frac{N_t m^{\ell+1} - (1-\gamma_1)N_t[(1-\gamma_2)m]^{\ell+1}}{N_t m^{\ell+1}} \\ &\leq 1-(1-\gamma_1)(1-\gamma_2)^{\ell+1} \end{aligned}$$

which completes the proof. ■

Corollary 2: For a p-ratio ABD-HFL structure with property γ_1-γ_2, a lower level tolerate a greater proportion of Byzantine nodes than its upper level.

Proof: Consider such an ABD-HFL of depth L, for any level L_ℓ and its adjacent lower level L'_ℓ with $0 \leq \ell \leq \ell'$Ł. As

$$\begin{gathered} \ell < \ell' \\ (1-\gamma_2) < 1 \\ \ell - 1 \geq 1 \\ \ell' - 1 \geq 1 \end{gathered}$$

then

$$(1-\gamma_2)^{(\ell-1)} > (1-\gamma_2)^{(\ell'-1)}$$

then

$$(1-\gamma_1)(1-\gamma_2)^{(\ell-1)} > (1-\gamma_1)(1-\gamma_2)^{(\ell'-1)}$$

As a result,

$$1-(1-\gamma_1)(1-\gamma_2)^{(\ell-1)} < 1-(1-\gamma_1)(1-\gamma_2)^{(\ell'-1)}$$

That is

$$P_\ell < P'_\ell$$

which completes the proof. ■

Corollary 3: For a p-ratio ABD-HFL structure with property γ_1-γ_2, if the number of bottom-level nodes is fixed, the more levels there are, the greater the proportion of Byzantine nodes that the system can tolerate at bottom level.

Proof: Comparing two ABD-HFL system $\mathcal{S}_1, \mathcal{S}_2$ with same property γ_1-γ_2, $\mathcal{S}_1$ and $\mathcal{S}_2$ have the same number of participants at bottom level, but the number of levels are different. Here we denote L_1 as the bottom level is level for $\mathcal{S}_1$, and denote L_2 for as the bottom level is level for $\mathcal{S}_2$, assuming $L_1 < L_2$ From theorem 2,

$$P_{1,L_1} = 1-(1-\gamma_1)(1-\gamma_2)^{(L_1-1)}$$
$$P_{2,L_2} = 1-(1-\gamma_1)(1-\gamma_2)^{(L_2-1)} P_{L_1}^{(S_1)} = 1-(1-\gamma_1)(1-\gamma_2)^{(L_1-1)}$$
$$P_{L_2}^{(S_2)} = 1-(1-\gamma_1)(1-\gamma_2)^{(L_2-1)}$$

As

$$L_1 < L_2$$
$$(1-\gamma_2) < 1$$
$$L_1 - 1 \geq 1$$
$$L_2 - 1 \geq 1$$

then

$$1-(1-\gamma_1)(1-\gamma_2)^{(L_1-1)} < 1-(1-\gamma_1)(1-\gamma_2)^{(L_2-1)}$$

That is

$$P_{L_1}^{(S_1)} < P_{L_2}^{(S_2)}$$

which proves the assertion. ■

APPENDIX C
ARBITRARY CLUSTER SIZE MODEL

In practical applications with ABD-HFL structure, the size of each cluster in each level may differ from each other. We further extend ECSM model into an Arbitrary Cluster Size Model(ACSM), where the cluster size can not only vary between levels, but also vary in the same level. We develop the property of ACSM model regarding to the Byzantine tolerance ability in each level. For convenience of analysis, we firstly give some important definitions.

Definition 5: If the proportion of malicious model updates collected from its cluster is greater than the maximum proportion of Byzantine tolerance, the cluster is called a *Byzantine cluster*, otherwise it is called a *honest cluster*.

Note that when this definition is applied at top level, each node is considered as a cluster itself and is the leader of this single-node cluster. Based on the above definition, we give the responding definition for clusters.

Definition 6: The leader of a Byzantine cluster is called a *Byzantine leader*, and the leader of a honest cluster is called a *honest leader*.

We define ABD-HFL to consist of Byzantine clusters and honest clusters, and other type of cluster are not possible. In any level, there may be multiple Byzantine clusters and honest clusters. We use $\mathcal{B}_\ell$ and $\mathcal{H}_\ell$ to denote the set of Byzantine cluster and honest clusters respectively. Obviously, we have

$$\mathcal{C}_\ell = \mathcal{B}_\ell \cup \mathcal{H}_\ell$$
$$C_\ell = |\mathcal{B}_\ell| + |\mathcal{H}_\ell|$$

We define B_ℓ and H_ℓ as the number of Byzantine nodes and honest nodes in level L_ℓ respectively, and denote the maximum proportion of Byzantine nodes tolerated in any level ℓ as P_ℓ, and the minimum proportion of honest nodes as Q_ℓ.

Similar to appendix B, we define $P_{\ell,i}$ as the maximum proportion of Byzantine nodes tolerated in any cluster $\mathcal{C}_{\ell,i}$, and $Q_{\ell,i}$ as the corresponding minimum proportion of honest nodes. Note that for a Byzantine cluster, it is possible that all cluster members are Byzantine nodes, so we have

$$0 \leq P_{\ell,i} \leq 1, i \in \mathcal{B}_\ell$$

While, for honest clusters there is a bound value for the proportion of Byzantine nodes; at the top level, the upper bound is γ_1, and for other levels it is γ_2, hence

$$0 \leq P_0 \leq \gamma_1 \tag{4}$$

$$0 \leq P_{\ell,i} \leq \gamma_2, i \in \mathcal{H}_\ell; \ell \geq 1 \tag{5}$$

We have

$$P_\ell + Q_\ell = 1 \tag{6}$$

$$P_{\ell,i} + Q_{\ell,i} = 1 \tag{7}$$

Then combining (4-7) we get

$$1 - \gamma_1 \leq Q_0 \leq 1 \tag{8}$$

$$1 - \gamma_2 \leq Q_{\ell,i} \leq 1, i \in \mathcal{H}_\ell; \ell \geq 1 \tag{9}$$

Then we can express H_ℓ and B_ℓ as below

$$B_\ell = P_\ell * N_\ell$$
$$H_\ell = Q_\ell * N_\ell$$

which is equal to

$$B_\ell = \sum_{i \in \mathcal{B}_\ell} P_{\ell,i} * C_{\ell,i} \tag{10}$$

$$H_\ell = \sum_{i \in \mathcal{H}_\ell} Q_{\ell,i} * C_{\ell,i} \tag{11}$$

We can express P_ℓ and Q_ℓ as follows:

$$P_\ell = \frac{B_\ell}{N_\ell} \tag{12}$$

$$Q_\ell = \frac{H_\ell}{N_\ell} \tag{13}$$

As with theorem 2, we can develop a property for any ABD-HFL regarding to the maximum proportion of Byzantine nodes tolerated in each level. Firstly we need to define an important quantity that represents the relative size of all honest clusters to the total size of the level as below.

Definition 7: The *relative reliable number* of level ℓ in ABD-HFL is the ratio between the total number of nodes in *honest clusters* to the total number of node in this level.

We use ψ_ℓ to denote the relative reliable number of level ℓ, by the definition

$$\psi_\ell = \frac{\sum_{i \in \mathcal{H}_\ell} C_{\ell,i}}{\sum_{i=1}^{C_\ell} C_{\ell,i}} \tag{14}$$

With this, we give the following property.

Theorem 3: For an ABD-HFL of L+1 levels with property γ_1-γ_2, for any level ℓ, $0 \leq \ell \leq L$, the maximum proportion of Byzantine nodes tolerated is inversely proportional to relative reliable number of the level.

Proof: We start the proof from the top level. For a valid ABD-HFL structure, at top level there is only one cluster which is regarded as an honest cluster. By (12), we get

$$Q_0 = \frac{H_0}{N_0} \tag{15}$$

and (14), we have

$$\psi_0 = \frac{\sum_{i \in \mathcal{H}_0} C_{0,i}}{\sum_{i=1}^{C_0} C_{0,i}} \tag{16}$$

As in top level, $|C_{0,i}| = 1$ for any $i \in \mathcal{H}_0$, so

$$H_0 = \sum_{i \in \mathcal{H}_0} C_{0,i}$$
$$N_0 = \sum_{i=1}^{C_0} C_{0,i}$$

Then from (15),(16) we get

$$Q_0 = \psi_0 \tag{17}$$

Combining (17),(7) we get

$$P_0 = 1 - \psi_0$$

which proves that the theorem is true for the top level.

We proceed to prove for any level L_ℓ with $\ell > 0$. According to (9),(11),(13) and definition 7 we get

$$\begin{aligned} Q_\ell &= \frac{H_\ell}{N_\ell} \\ &= \frac{\sum_{i \in H_\ell} Q_{\ell,i} * C_{\ell,i}}{\sum_{i=1}^{C_\ell} C_{\ell,i}} \\ &\geq (1-\gamma_2) \frac{\sum_{i \in H_\ell} C_{\ell,i}}{\sum_{i=1}^{C_\ell} C_{\ell,i}} \\ &\geq (1-\gamma_2)\psi_\ell \end{aligned}$$

Then, from (6) we can get

$$\begin{aligned} P_\ell &= 1 - Q_\ell \\ &\leq 1 - (1-\gamma_2)\psi_\ell \end{aligned}$$

which shows that P_ℓ is inversely proportional to the relative reliable number ψ_ℓ, compeleting the proof. ■

APPENDIX D
NUMERICAL EVALUATION SETTINGS

A. Basic Settings

In all simulations, the learning consists of a total of 200 global training epochs, and 5 local training iterations.

The setting of data distribution is as below:

- *IID data distribution* In IID cases, training samples for each label are shuffled and then distributed equally to all clients, with each client having approximate 937 samples covering sample labels from 0 to 9.
- *non-IID distribution* In non-IID cases, the size of training datasets is evenly assigned to each client, each client has only 2 sample labels, which is regarded as an extreme non-IID cases. A special design is set in the code to ensure that honest participants as a whole cover all ten labels, so that the final accuracy is affected by the sample size of all honest nodes, rather than the lack of some sample labels.

In ECSM model, the cluster sizes are equal at all levels except the top level. We construct a ABD-HFL model of three-level network with a cluster size of 4, and 4 nodes at the top level. As a result, there are 64 clients at the bottom level performing the local training tasks. In the simulation, the leader of each cluster is assigned virtually, and clients are ordered by client id from 0 to 63. The vanilla FL is set with a central server as aggregation for all 64 clients. Table VII lists the detail of simulation settings.

Specially, for ABD-HFL, at the top level, only one of all four top-level nodes is considered malicious, given the maximum tolerance of 33% Byzantine participants in a distributed environment. Note that in the data poisoning attack, a malicious node manipulates training data instead of model updates. According to this definition, in ABD-HFL, even if a malicious node is elected as its cluster leader, it will "honestly" aggregate local models from its cluster members and uploads the partial aggregated model to its leader at upper level. So in this simultion the proportion of malicious nodes refers to the bottom layer in ABD-HFL, and considering that all leader nodes are initially elected from the bottom layer, the number of bottom nodes is equal to the total number of nodes.

TABLE VII
NUMERICAL EVALUATION SETTINGS

Items	Parameters	Settings	Notes
Dataset		MNIST	
Learning settings	Global rounds	200	
	Local iterations	5	
ABD-HFL settings	Model	ECSM	
	Number of levels	3	
	Cluster size	4	Number of nodes in each cluster
	Number of top-level nodes	4	
	Number of bottom-level clients	64	
Vanilla FL	Number of clients	64	
Data distribution	Number of samples for each client	947	

TABLE VIII
DELAY CASES OF DIFFERENT BYZANTINE SCHEME

Delay Case	Advises for L_k
big τ'-big τ_g	depends on other factors
small τ'-small τ_g	close to top level
small τ'-big τ_g	close to top level
big τ'-small τ_g	depends on other factors

B. Voting based consensus mechanism

At the top level of ABD-HFL, each node votes on the partial aggregated model received from other top-level nodes. Upvoting or downvoting a model is decided after testing a node's testing dataset.

The partial models that receive the fewest number of positive votes are considered malicious, and are excluded from the final global model aggregation. As an explanation, given the limitation of the testing dataset size (10000), instead of distributing them evenly to all clients, we assign them evenly to 4 top-level nodes, so that the top level nodes can make meaningful votes.

APPENDIX E
DISCUSS ON FLAG LEVEL SETTINGS

Several important factors should be considered on choosing the best flag level, shown as following.

- *The number of levels in ABD-HFL* The depth of the ABD-HFL determines the available choice of flag level L_k. We start from the simplest case, $L = 1$, where there is only two levels, the top and the bottom level, and the descendants of each node of top level are directly the local model training participants. In this case, the only possible for flag level is at top level, that is $L_k = 0$. The top-level nodes will collect all local model updates and form the partial aggregated model, which will then be immediately returned to its descendants for next round of training. At the same time, the top-level nodes starts the process of global model aggregation. For ABD-HFL with more than 2 levels, $L > 1$, there are L possible choices for flag level, $L_k \in \{0, 1, \cdots, L-1\}$. Then the balance of the trade-offs should be well considered in such a choice.
- *The aggregation schemes* The aggregation scheme significantly affects the choice of L_k. As shown in section IV-A, the scheme determines the delay in aggregation processes. From this perspective, four possible compositions of delays are shown in Table VIII. Note that we use τ' to denote the delay caused by the partial aggregation process, τ_g to denote the delay caused by the global aggregation. In the case of big τ'-big τ_g, there is a long delay in both partial aggregation and global aggregation, and the choice of L_k depends on other factors and needs to balance the trade-off between the efficiency indicator and cost of correction factor. While, in small τ'-small τ_g, the delays of both are short. In this case L_k should be located near the top level because the cost of correction factor becomes the main consideration due to the small absolute value of the gain of efficiency improvement. In the case of big τ'-small τ_g, the delay caused by partial aggregation is longer, while the delay caused by global aggregation is shorter, in this case L_k should be decided together with all other factors. For the case of small τ'-big τ_g, it is recommended that L_k should be located near the top level if no other factors strongly oppose it. Since the partial aggregation delays are small, we can achieve relatively large efficiency gains and keep the cost of correction factor as low as possible.
- *Bandwidth difference of each level* The difference in bandwidth at each level affects the propagation of the model. Although it is ignored in the model in section III-D1, in reality they may become a major factor to be considered, especially when the AI model is large and has limited bandwidth.
- *Quality of the federated learning task* The quality of FL tasks also plays a key role in the selection of L_k. One factor that has a profound impact on this is the use of correction factor. While it is assumed that the correction factor is sufficient

to compensate for the staleness of the new arrived global model compared to the flag partial model being trained, it is hardly comparable to the effect of training directly with global model. For FL tasks that are highly sensitive to user data, or tasks that will have serious consequences if quality cannot not guaranteed, such as cases in medical diagnoses, efficiency consideration should be reduced when designing L_k, or the global model should be used directly to replace the flag partial model. While there are situations where the speed of FL is critical, such as in real-time decision-making business, efficiency indicator becomes more important, and cost of correction factor or even sacrificing some of training quality is acceptable to the task publishers.