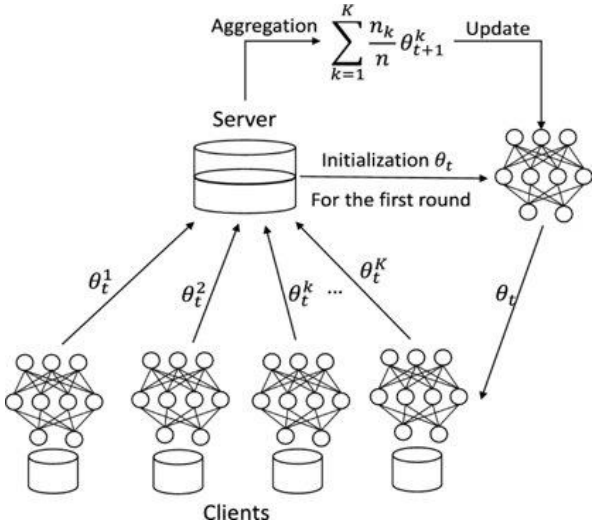
FL Client Selection

2022_Fall

x

Vanilla Federated Learning



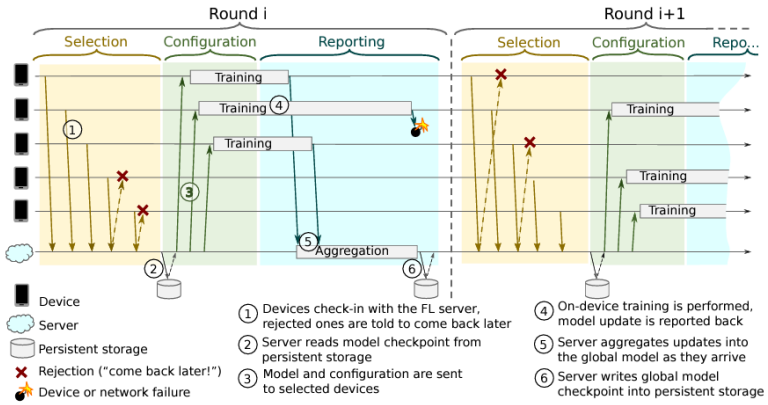


Figure 1: Federated Learning Protocol

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:

initialize w_0

for each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of m clients)

for each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)

for each local epoch i from 1 to E **do**

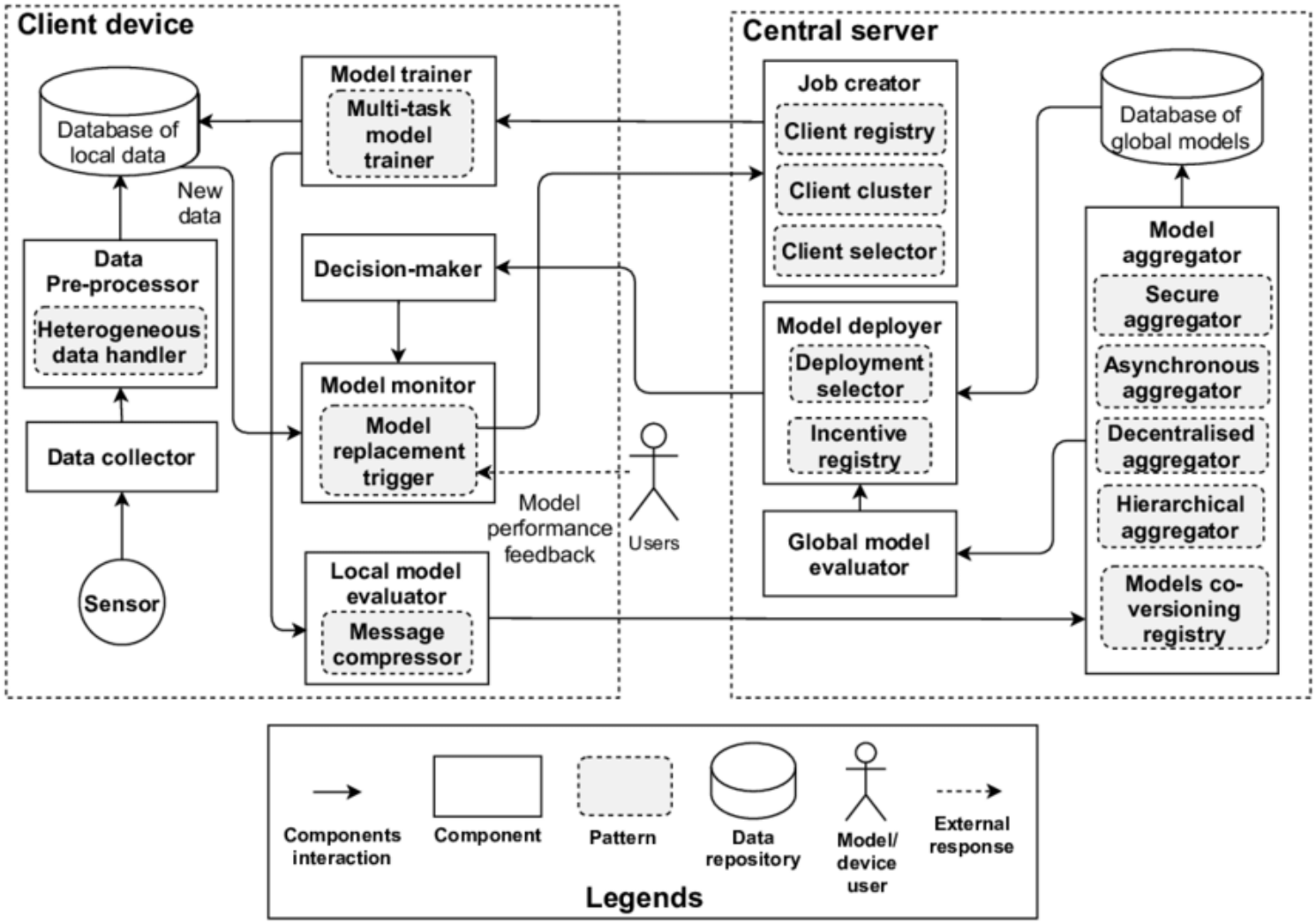
for batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return w to server

FLRA: A Reference Architecture for Federated Learning Systems

FLRA: A Reference Architecture for Federated Learning Systems, https://arxiv.org/abs/2106.11570



FLRA: A Reference Architecture for Federated Learning Systems

FLRA: A Reference Architecture for Federated Learning Systems, https://arxiv.org/abs/2106.11570

The central servers interacts with a massive number of client devices that are both system heterogeneous and statistically heterogeneous. The magnitude of client devices number is also several times larger than that of the distributed machine learning systems [18,24]. To increase the model and system performance, client devices can be selected every round with predefined criteria (e.g., resource, data, or performance) via **client selector** component.

Job creation	Mandatory	Job creator	Initialises training job and global model
	Optional	Client registry	Improves system's **maintainability** and **reliability** by maintaining client's information
		Client cluster	Tackles **statistical heterogeneity** & **system heterogeneity** by grouping clients with similar data distribution or resources before aggregation
		Client selector	Improves **model & system's performance** by selecting high performance client devices
Data collection & preprocessing	Mandatory	Data collector	Collects raw data through sensors or smart devices deployed
		Data preprocessor	Preprocesses raw data
	Optional	Heterogeneous Data Handler	Tackles **statistical heterogeneity** through data augmentation methods

FLRA: A Reference Architecture for Federated Learning Systems

FLRA: A Reference Architecture for Federated Learning Systems, https://arxiv.org/abs/2106.11570

Local model training.
Once the client receives the job from the central server, the model trainer component performs model training based on configured hyperparameters (number of epochs, learning rate, etc.). In the standard federated learning training process proposed by McMahan in [28], only model parameters (i.e., weight/gradient) are mentioned to be sent from the central server, whereas in this reference architecture, the models include not only the model parameters but also the hyperparameters.

Model evaluation.
The **local model evaluator** component measures the performance of the local model and uploads the model to the model aggregator on the central server if the performance requirement is met. In distributed machine learning systems, the performance evaluation on client devices is not conducted locally, and only the aggregated server model is evaluated. However, for federated learning systems, local model performance evaluation is required for system operations such as client selection, model co-versioning, contributions calculation, incentive provision, client clustering, etc.

Model training	Mandatory	Model trainer	Trains local model
		Local model evaluator	Evaluates local model performance after each local training round
		Model aggregator	Aggregates local models to produce new global model

FLRA: A Reference Architecture for Federated Learning Systems

FLRA: A Reference Architecture for Federated Learning Systems, https://arxiv.org/abs/2106.11570

this technique is particularly relevant when faced with nonIID data which can produce personalised model that may outperform the best possible shared global model [18]

The conventional design of a federated learning system that relies on a central server to orchestrate the learning process might lead to a single point of failure. **A decentralise aggregator** performs model exchanges and aggregation in decentralised manner to improve system reliability. The known uses of decentralised aggregator include BrainTorrent [31] and FedPGA [15]. Blockchain can be employed as a decentralised solution for federated learning systems.

Optional	Multi-task model trainer	Improves **model performance** (personalisation) by adopting multi-task training methods
	Message compressor	Improves **communication efficiency** through message size reduction to reduce bandwidth consumption
	Secure aggregator	Improves **data privacy** & **system security** through different secure multiparty computation protocols
	Asynchronous aggregator	Improves **system performance** by reducing aggregation pending time of late client updates
	Decentralised aggregator	Improves system **reliability** through the removal of single-point-of-failure
	Hierarchical aggregator	Improves **system performance** & tackle **statistical heterogeneity** & **system heterogeneity** by aggregating models from similar clients before global aggregation
	Model co-versioning registry	Improves system's **accountability** by recording the local models associated to each global models to track clients' performances

FLRA: A Reference Architecture for Federated Learning Systems

FLRA: A Reference Architecture for Federated Learning Systems, https://arxiv.org/abs/2106.11570

The **incentive registry** component maintains all the client devices' incentives based on their contributions and agreed rates to motivate clients to contribute to the training. Blockchain has been leveraged in FLChain [3] and DeepChain [36] to build a incentive registry.

Model deployment	Mandatory	Model deployer	Deploys completely-trained-models
		Decision maker	Decides model deployment
	Optional	Deployment selector	Improves **model performance** (personalisation) through suitable model users selection according to data or applications
		Incentive registry	Increases clients' **motivatability**
Model monitoring	Mandatory	Model monitor	Monitors model's data inference performance
	Optional	Model replacement trigger	Maintains **system & model performance** by replacing outdated models due to performance degrades

Client Selection

Oort: Efficient Federated Learning via Guided Participant Selection, https://www.usenix.org/conference/osdi21/presentation/lai
https://github.com/Kwangkee/FL/blob/main/FL%40ClientSelection.md#oort

As a result, data characteristics and device capabilities vary widely across clients. Yet, **existing efforts randomly select FL participants, which leads to poor model and system efficiency.** In this paper, we propose Oort to improve the performance of federated training and testing with guided participant selection.

With an aim to improve time-to-accuracy performance in model training, **Oort prioritizes the use of those clients who have both data that offers the greatest utility in improving model accuracy and the capability to run training quickly.**

Unfortunately, clients may not all be simultaneously available for FL training or testing [44]; they may have heterogeneous data distributions and system capabilities [19,38]; and including too many may lead to wasted work and suboptimal performance [19] (§2). **Consequently, a fundamental problem in practical FL is the *selection of a "good" subset of clients as participants*,** where each participant locally processes its own data, and only their results are collected and aggregated at a (logically) centralized coordinator.

Although random participant selection is easy to deploy, unfortunately,

- it results in poor performance of federated training because of large heterogeneity in device speed and/or data characteristics.
- **Worse, random participant selection can lead to biased testing sets and loss of confidence in results.**

Client Selection

Oort: Efficient Federated Learning via Guided Participant Selection, https://www.usenix.org/conference/osdi21/presentation/lai
https://github.com/Kwangkee/FL/blob/main/FL%40ClientSelection.md#oort

1. Job submission
2. Participant selection:
- the coordinator enquires the clients meeting eligibility properties (e.g., battery level), and forwards their characteristics (e.g., liveness) to Oort. Given the developer requirements (and execution feedbacks in case of training 2a),
- Oort selects participants based on the given criteria and notifies the coordinator of this participant selection(2b).

3. Execution
4. Aggregation

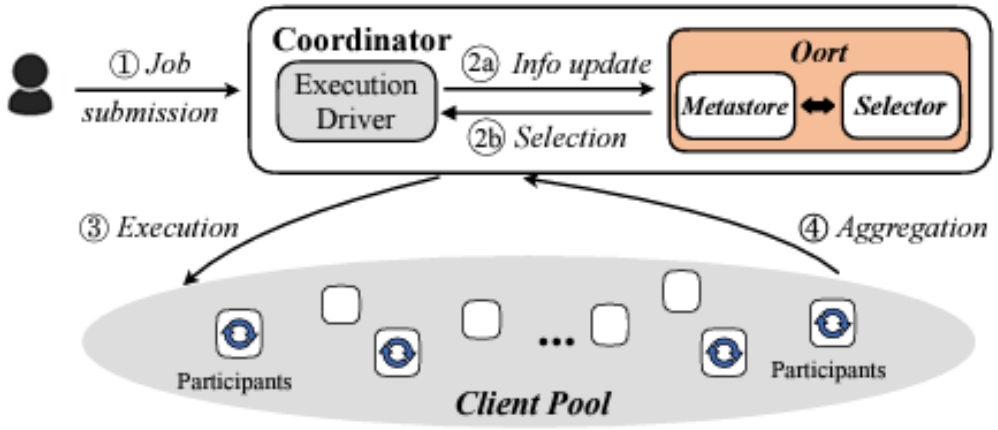


Figure 5: ***Oort architecture. The driver of the FL framework interacts with Oort using a client library.***

Client Selection

Oort: Efficient Federated Learning via Guided Participant Selection, https://www.usenix.org/conference/osdi21/presentation/lai
https://github.com/Kwangkee/FL/blob/main/FL%40ClientSelection.md#oort

주요 아이디어: loss-based statistical utility design
주요 아이디어: MAB (Multi-Armed Bandit) problem, exploration-exploitation

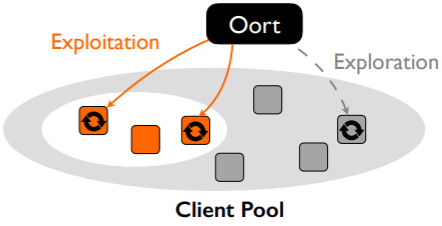
Challenge 1: Identify Heterogeneous Client Utility

- **Statistical utility**
  - Capture how the client data can help to improve the model
  - **Metric**: *aggregate training loss* of client data
    - Higher loss ⟶ higher stats utility (proof in paper)
- **Utility of a client** = $\frac{stats\_util\ (i)}{round\_duration\ (i)}$
  - i.e., *speed* of accumulating stats utility in ***round i***

Heterogeneity Scalability Dynamics Robustness 18

Challenge 2: Select High-Utility Clients at Scale

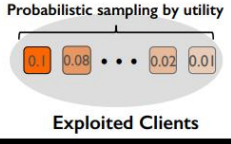
- **How to identify high-utility clients from millions of clients?**
  - *Spatiotemporal* variation: heterogeneous utility across clients over rounds
- **Exploration + Exploitation**
  - Explore not-tried clients
  - Exploit known *high-utility* clients



Challenge 3: Select High-Utility Clients Adaptively

- **How to account for stale utility since last participation?**
  - Utility changes due to dynamics

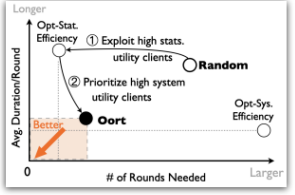
1. **Aging**: add uncertainty to utility ⟶ *Re-discover missed good clients*
   - current_utility = last_observed_utility + *observation_age*
2. **Probabilistic selection** by utility values
   - Prioritize high-utility clients
   - Robust to outliers and uncertainties



Heterogeneity Scalability Dynamics Robustness 21

More in Our Paper

- **How to respect privacy**
- **How to be robust to corrupted clients**
- **How to enforce diverse selection criteria**
  - Fairness, data distribution for FL testing



Heterogeneity Scalability Dynamics Robustness 22

Sample Selection

FedBalancer: Data and Pace Control for Efficient Federated Learning on Heterogeneous Clients (ACM MobiSys 2022), https://nmsl.kaist.ac.kr/projects/fedbalancer/
https://github.com/Kwangkee/FL/blob/main/FL%40ClientSelection.md#fedbalancer

Unlike centralized training that is usually based on carefully-organized data, FL deals with on-device data that are often unfiltered and imbalanced. As a result, conventional FL training protocol that treats all data equally leads to a waste of local computational resources and slows down the global learning process.

To this end, we propose FedBalancer, a systematic FL framework that **actively selects clients' training samples.** Our sample selection strategy prioritizes more "informative" data while respecting privacy and computational capabilities of clients. To better utilize the sample selection to speed up global training, we further introduce an adaptive deadline control scheme that predicts the optimal deadline for each round with varying client training data.

For model developers who prototype a mobile AI with **FL without a proxy dataset,** achieving faster convergence on thousands to millions of devices is desired to efficiently test multiple model architectures and hyperparameters [33]. Service providers who frequently update a model with continual learning with FL require to minimize the user overhead with better time-to-accuracy performance [39].

A key objective in FL is to optimize time-to-accuracy performance. FL tasks typically require hundreds to thousands of rounds to converge [13, 38], and clients participating at a round undergo substantial computational and network overhead [21]. Deploying FL across thousands to millions of devices should be done efficiently, quickly reaching the model convergence while not sacrificing the model accuracy. This becomes more important when FL has to be done multiple times, as often the case when model developers prototype a new model with FL without a proxy dataset or periodically update a deployed model to new domain via continual learning or online learning with FL.

Sample Selection

FedBalancer: Data and Pace Control for Efficient Federated Learning on Heterogeneous Clients (ACM MobiSys 2022), https://nmsl.kaist.ac.kr/projects/fedbalancer/

The sample selection of FedBalancer prioritizes more "informative" samples of clients to efficiently utilize their computational effort. **This allows low-end devices to contribute to the global training within the round deadline by focusing on smaller but more important training samples.** To achieve high time-to-accuracy performance, the sample selection is designed to operate without additional forward or backward pass for sample utility measurement at FL rounds. Lastly, FedBalancer can coexist and collaborate with orthogonal FL approaches to further improve performance.

The loss threshold ratio (ltr) enables FedBalancer to start training with all samples and gradually remove already-learned samples. FedBalancer initialize ltr as 0.0 and gradually increases the value by loss threshold step size (lss) as shown in Algorithm 3. Note that the deadline ratio ($ddlr$), which controls the deadline of each round (described in Section 3.3), is also controlled with ltr.

FedBalancer gradually increase loss threshold to remove already-learned samples
- **Round 가 진행될수록, Loss threshold 는 gradually increase**

The intuition of sampling a portion of data from UTi is to avoid catastrophic forgetting [35, 78] of the model on already-learned sample
- We sample $L \cdot p$ samples from OTi and $L \cdot (1 - p)$ samples from UTi where L indicates the number of selected samples and p is a parameter in an interval of [0.5, 1.0]
- L, the length of selected samples, is determined based on the hardware speed of a client
- p is a FedBalancer parameter between 0.5 ≤ p ≤ 1.0.
- -> p 가 클수록, catastrophic forgetting 을 좀 더 걱정한다는 의미.

Client Selection

Yae Jee Cho, https://github.com/Kwangkee/FL/blob/main/FL@CarnegieMellon.md#yae-jee-cho

Towards Understanding Biased Client Selection in Federated Learning,
https://proceedings.mlr.press/v151/jee-cho22a.html
In our work, we present the convergence analysis of federated learning with biased client selection and quantify how the bias affects convergence speed. **We show that biasing client selection towards clients with higher local loss yields faster error convergence.**

To Federate or Not To Federate: Incentivizing Client Participation in Federated Learning,
https://arxiv.org/abs/2205.14840
Figure 2: Aggregating weight qk(w) for any clientk versus the emprical incentive gap Fk(w) − Fk(wb k). The weight qk(w) is small for clients that already have a very large incentive (global much better than local) or no incentive at all (local much better than global), and is highest for clients that are moderately incentivized (global similar to local).