Demo: FLaaS - Enabling Practical Federated Learning on Mobile Environments

Kleomenis Katevas[*]
Brave Software
London, UK
kkatevas@brave.com

Diego Perino
Telefonica Research
Barcelona, Spain
diego.perino@telefonica.com

Nicolas Kourtellis
Telefonica Research
Barcelona, Spain
nicolas.kourtellis@telefonica.com

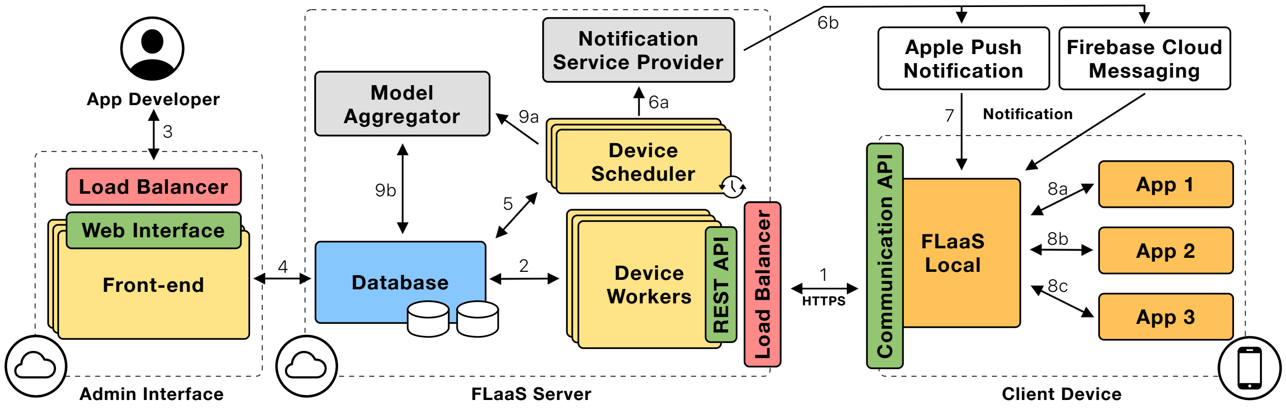


Figure 1: FLaaS architecture. A set of devices participate in FLaaS and periodically report their status (1) through the back-end's REST API, (2) to the DB for logging. App developers use the Admin Interface to (3) create new FLaaS projects, whose configuration is (4) stored in the DB. The Device Scheduler: (5) detects new FLaaS projects and queries for device statuses from the DB; (6a) sends an FL training request using its Notification Service provider to (6b) external services (e.g., APN or FCM). Each device's FLaaS Local (7) receives the training request and (8a, 8b, 8c) requests from collaborating 3rd-party apps to send either their local samples for training, or to train their own models. It then performs the required on-device ML training (if it received samples) or model aggregation/averaging, depending on ML training mode. When enough models are received by the Server (1), the Device Scheduler (9a, 9b) instructs the Model Aggregator to accumulate the received models, marks the FL round as complete, and continues with the next FL round until the project is complete (*i.e.*, reaching stopping criteria).

ACM Reference format:
Kleomenis Katevas, Diego Perino, and Nicolas Kourtellis. 2022. Demo: FLaaS - Enabling Practical Federated Learning on Mobile Environments. In *Proceedings of The 20th Annual International Conference on Mobile Systems, Applications and Services, Portland, OR, USA, June 25-July 1, 2022 (MobiSys '22),* 2 pages. https://doi.org/10.1145/3498361.3539693

1 MOTIVATION

Federated Learning (FL) [2] has emerged as a popular solution of Confidential Computing [3] to distributedly train a model on user devices, improving privacy and system scalability. Such privacy-preserving models can be used in wide range of applications, and especially in Telco networks [4]. However, there are no practical systems to easily enable FL training on mobile apps, and especially in an *as-a-service* fashion. In this demo, we implement and test FLaaS, our recently proposed end-to-end FL service [1]. FLaaS includes a client-side framework with app library and service, and a back-end server, to enable secure and easy to deploy intra- and inter-app FL model training on mobile environments.

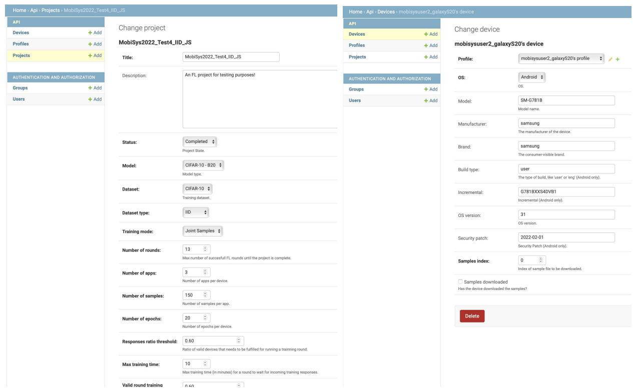
[*]Work performed while at Telefonica Research

1.1 Design Principles

In particular, FLaaS system has been design to be:
a) Easy to use by incorporating it in existing apps, and providing a simple user interface to configure the FL process.
b) On-device training capable by building on TensorFlow Lite to support different FLaaS ML model training modes: i) individual model per app, ii) joint model across collaborating apps.
c) Secure & private while training FL models, using i) secure communication channels and authentication between apps and FLaaS service, ii) secure on-device data storage.
d) Production ready: as it was tested in the wild on 140 real devices to deal with challenges such as scalability and robustness using load balancers, cloud-based notification service, etc.

2 FLAAS DESIGN & IMPLEMENTATION

Fig. 1 outlines the FLaaS architecture and steps required to perform an FL training process within FLaaS. The back-end incorporates the roles of Admin Interface, Server and Notification Service. It is hosted in the Heroku Cloud platform (heroku.com) and uses *Standard-1X Dynos* (512MB RAM), Linux-based shared containers to enable easy deployment, scaling, load balancing and scheduling of FLaaS modules.
a) Admin Interface: Front-end interface for app developers used to bootstrap, configure, execute and monitor FL projects. Some examples are shown in Fig. 2 for new FLaaS project creation and management of details for a Client Device.
b) Server: Scalable, cloud-hosted service in charge of orchestrating / monitoring FL processes on behalf of app developers. It consists of the following modules: 1) Database (a PostgreSQL v13.4 instance used for the Relational DB, and an Amazon S3 Data Store used as an Object


Home › Api › Projects › MobiSys2022_Test4_IID_JS

API
Devices + Add
Profiles + Add
Projects + Add

AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add

Change project

MobiSys2022_Test4_IID_JS

Title:	MobiSys2022_Test4_IID_JS
Description:	An FL project for testing purposes!
Status:	Completed Project State.
Model:	CIFAR-10 - B20 Model type.
Dataset:	CIFAR-10 Training dataset.
Dataset type:	IID
Training mode:	Joint Samples
Number of rounds:	13 Max number of succesfull FL rounds until the project is complete.
Number of apps:	3 Number of apps per device.
Number of samples:	150 Number of samples per app.
Number of epochs:	20 Number of epochs per device.
Responses ratio threshold:	0.60 Ratio of valid devices that needs to be fulfilled for running a trainning round.
Max training time:	10 Max training time (in minutes) for a round to wait for incoming training responses.
Valid round training	0.60

Home › Api › Devices › mobisysuser2_galaxyS20's device

API
Devices + Add
Profiles + Add
Projects + Add

AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add

Change device

mobisysuser2_galaxyS20's device

Profile:	mobisysuser2_galaxyS20's profile
OS:	Android OS.
Model:	SM-G781B Model name.
Manufacturer:	samsung The manufacturer of the device.
Brand:	samsung The consumer-visible brand.
Build type:	user The type of build, like 'user' or 'eng' (Android only).
Incremental:	G781BXXS4DVB1 Incremental (Android only).
OS version:	31 OS version.
Security patch:	2022-02-01 Security Patch (Android only).
Samples index:	0 Index of sample file to be downloaded.

☐ Samples downloaded
Has the device downloaded the samples?

Figure 2: User interface for (left) creation of new FLaaS project with various settings, and (right) new device details.

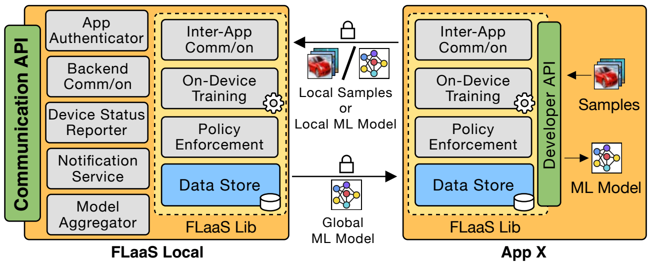


Figure 3: Inter and intra-app modules on Client Devices.

Storage DB), 2) Device Schedulers (one-off Worker (Heroku Dyno) that is only instantiated periodically every T minutes, and handled by the *Heroku Scheduler service*, by starting a new Dyno, one per FL project), 3) Model Aggregator (Worker started by Scheduler when FL project's termination conditions are met), 4) Device Workers (each uses Django v3.2 under Python v3.8.12 and has a dedicated REST API based on Django REST framework v3.12.4), 5) Load Balancer (provided by Heroku service). Communication between Device Workers and Client Devices is encrypted using TLS v1.3 over HTTPS.

c) Notification Services: Mobile push notification services used for push comms to client devices through silent push notifications for new FL projects and rounds execution (e.g., Apple Push Notification (APNs) and Firebase Cloud Messaging (FCM)), provided by pushwoosh.com service.

d) Client Devices: In charge of computing the ML training tasks of a FLaaS FL project orchestrated by the Server. On such devices, there are two main modules: FLaaS Local and FLaaS Library. These modules facilitate the FLaaS on-device functions summarized in Fig. 3: App authentication, inter-app communication, access policy management, ML model training, data storing, and status reporting. FLaaS currently supports Android-based devices, Android 8.0+, since iOS does not provide (yet) in-app comm features necessary for joint, cross-app training. Third-party apps can integrate the FLaaS-Library with ~10 lines of code.

On-device ML training modes: FLaaS currently supports: (1) Individual model, where each app builds its own model on its samples, (2) Joint Samples (JS) model, where 2 or more collaborating apps share samples with FLaaS Local that builds a unified model, and (3) Joint Models (JM), where the collaborating apps train their own models on their data, and then share these models with FLaaS Local for averaging and reporting.

3 FLAAS EXPERIMENTATION

We performed a user study with 140+ real user devices and measured the overall time duration for an FL round in each of the two ML model training modes: Joint Samples vs. Joint Models, as well as time needed for different on-device functions. Preliminary results shown in Fig. 4

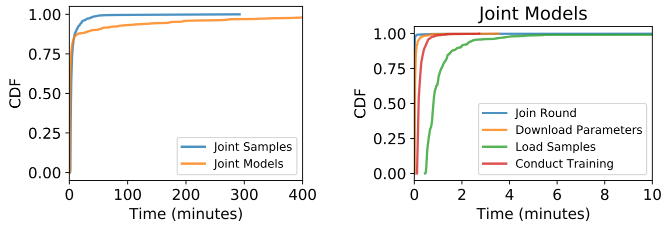


Figure 4: (left) Total duration of FL round, across projects using JS or JM mode. (right) Average time taken to perform different functions related to participation in an FL project using JM mode.

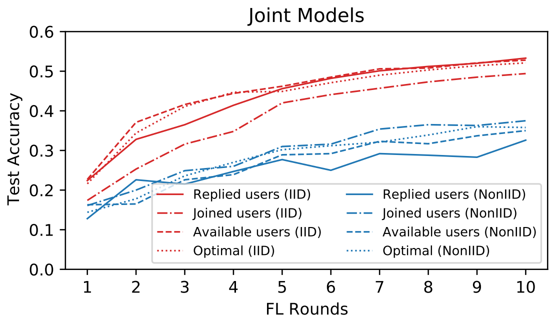


Figure 5: Test accuracy achieved per FL round, for Joint Models, using IID or NonIID data.

demonstrate that the majority of Client devices can return a useful model quite fast: the 80^{th} percentile duration of an FL round is 6.15 minutes for Joint Samples and 4.65 minutes for Joint Models. Currently, the most costly function in the on-device FL process is the loading of samples for training. Also, in Fig. 5 we show the ML utility (test accuracy) of the model built with Joint Models on real availability of users, again demonstrating comparative performance vs. ideal conditions when all registered users are considered available.

4 FLAAS DEMONSTRATION

This live demo includes the following steps:

Step 1: Creation of new FLaaS projects with all required configurations, using online front-end app developer interface.

Step 2: Deployment of project on several real mobile devices with different FLaaS-enabled mobile apps, to train FL models collaboratively.

Step 3: Monitoring of project's health and global FL model performance as in Fig. 5 on different experimental scenarios.

5 CODE RELEASE

FLaaS is open-sourced under MIT License to increase FL adoption, repeatability and further research on practical FL systems:

https://github.com/FLaaSResearch

ACKNOWLEDGMENT

This project received funding from EC under grant agreements No 830927 (Concordia), No 871370 (Pimcity) and No 871793 (Accordion).

REFERENCES

[1] Kourtellis, N., Katevas, K., and Perino, D. Flaas: Federated learning as a service. In *ACM DistributedML* (2020).

[2] Mo, F., Haddadi, H., Katevas, K., Marin, E., Perino, D., and Kourtellis, N. PPFL: Privacy-preserving federated learning with trusted execution environments. In *ACM MobiSys* (2021), pp. 94–108.

[3] Mo, F., Haddadi, H., Katevas, K., Marin, E., Perino, D., and Kourtellis, N. Ppfl: Enhancing privacy in federated learning with confidential computing. *ACM GetMobile: Mobile Comp. and Comm. 25*, 4 (Mar 2022), 35–38.

[4] Perino, D., Katevas, K., Lutu, A., Marin, E., and Kourtellis, N. Privacy-preserving ai for future networks. *Commun. ACM 65*, 4 (Mar 2022), 52–53.