This project has received funding from the European's Union Horizon 2020 research innovation programme under Grant Agreement No. 957258

Architecture for Scalable, Self-*, human-centric, Intelligent, Secure, and Tactile next generation IoT

ASSIST-IoT Technical Report #13

Introducing Federated Learning into Internet of Things ecosystems – preliminary considerations

Karolina Bogacka, Katarzyna Wasielewska-Michniewska, Marcin Paprzycki, Maria Ganzha, Anastasiya Danilenka, Lambis Tassakos, Eduardo Garro

Submitted to the IEEE 8th World Forum on Internet of Things

Introducing Federated Learning into Internet of Things ecosystems – preliminary considerations

Karolina Bogacka
Systems Research Institute
Polish Academy of Sciences
Warsaw, Poland
0000-0002-7109-891X

Katarzyna Wasielewska-Michniewska
Systems Research Institute
Polish Academy of Sciences
Warsaw, Poland
0000-0002-3763-2373

Marcin Paprzycki
Systems Research Institute
Polish Academy of Sciences
Warsaw, Poland
0000-0002-8069-2152

Maria Ganzha, Anastasiya Danilenka
Faculty of Mathematics and Information Science
Warsaw University of Technology
Warsaw, Poland
0000-0001-7714-4844, 0000-0002-3080-0303

Lambis Tassakos
TwoTronic Gmbh
Meitingen, Germany
0000-0003-2511-9035

Eduardo Garro
Prodevelop
Valencia, Spain
0000-0002-8160-0125

***Abstract*—Federated learning (FL) was proposed to facilitate the training of models in a distributed environment. It supports the protection of (local) data privacy and uses local resources for model training. Until now, the majority of research has been devoted to "core issues", such as adaptation of machine learning algorithms to FL, data privacy protection, or dealing with the effects of uneven data distribution between clients. This contribution is anchored in a practical use case, where FL is to be actually deployed within an Internet of Things ecosystem. Hence, somewhat different issues that need to be considered, beyond popular considerations found in the literature, are identified. Moreover, an architecture that enables the building of flexible, and adaptable, FL solutions is introduced.**

***Index Terms*—applied federated learning, Internet of Things, federated learning topology**

I. Introduction

One of the critical (and practical) bottlenecks of the application of Machine Learning (ML) lies in the limited ability to collect, consistently label, and use large datasets. This is particularly the case for businesses that do not possess almost unlimited resources, as Google or Amazon do [1]. Moreover, while existing data may be large and labeled, it may be "split between stakeholders", who do not want to and/or cannot share their datasets [2]. For instance, this is the case for the medical data, which belongs to different hospitals/clinics. Moreover, there are ongoing controversies concerning the collection and storage of information [3]. Many ML developments, e.g. in mobile applications, rely on the models being periodically (re/up)trained on sensitive private data (e.g., browsing history, or geo-positioning). Hosting such data in a centralized location, even in adherence to strict legislation, still poses serious security risks, as can be seen through repeated data leaks [4]–[6].

It is also worth noting that the latest advancements in ML involve training very large models that require enormous computational resources [7]. This not only increases the cost but also the carbon footprint [8].

To overcome these, and other related, problems, Federated Learning (FL) has been proposed [9]. The name of the approach came from the use of a flexible *federation* of collaborating (often heterogeneous, edge) devices, known as clients, "synchronized" and "orchestrated" by a "central server". In FL, (i) clients train copies of the global model, using local data, and (ii) send updates to the server, which (iii) aggregates them, and (iv) updates the shared model [10], which (v) is sent back to the clients to continue the process, until a stopping criterion is met. Therefore, private data never leaves the clients [11]. While a lot of research is devoted to the FL process itself, it is mostly implemented and tested in a cloud. This means that important practical issues that, as we will argue, have to be resolved, are omitted [12], [13].

One should immediately realize that one of the future areas of application of FL is the Internet of Things. Among others, this is the result of a general trend to replace cloud-centric solutions with edge-cloud continuum-based approaches [14]. This is happening because storing data, and providing resources, in the data center is not sustainable for large-scale complex deployments, where latency can negatively impact performance. Hence, computing has to take place near (at) the edge of the network, physically close to sensors and/or users [15]. The resulting ecosystem represents the edge-cloud continuum and is the necessary direction for the evolution of Next-Generation Internet of Things deployments [13]. Here, among others, FL will deliver intelligence at the edge [10]. However, combining FL with IoT brings about its own issues: (i) heterogeneity of clients and networks can cause delays (latency variability), or the presence of "stragglers" (weaker/more

This work is part of ASSIST-IoT project that has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement 957258. Work of Maria Ganzha and Anastasiya Danilenka is funded in part by the Centre for Priority Research Area Artificial Intelligence and Robotics of Warsaw University of Technology within the Excellence Initiative: Research University (IDUB) programme.

busy clients); (ii) computing and/or storage resources on the (far) edge devices, as well as their battery life, tend to be very limited, which impedes the use of large models and poses restrictions on training time; and (iii) data used for the training can be highly redundant [12].

As noted, core research on Federated Learning is focused on machine learning (ML) and its intricacies. This can be seen also when one considers state-of-the-art of FL frameworks. For example, though TensorFlow Federated Framework (TFF) [16] offers a wide variety of stable ML models, it supports experimentation only in a simulated environment. In other words, TFF currently does *not* enable use of actual "edge devices". Another widely known FL platform is FATE [17]. Here, 6GB of RAM, and 100 GB of disk space, on the server as well as on the clients are expected. While this would work in a laboratory, it exceeds the capabilities of the majority of edge devices (at least of today). Among platforms, PaddleFL enables the implementation of decentralized architectures by default. However, due to the low number of current contributors, and the employment of PaddlePaddle, a lesser-known Deep Learning platform [18], PaddleFL may not be an optimal choice for future work. Flower (A Friendly Federated Learning Framework [19]) can be run on a diverse range of environments and devices, including Android, iOS, Raspberry Pi, and Nvidia Jetson. It is also compatible with popular ML frameworks like PyTorch and Keras. Finally, PySyft [20] allows the use of clients on the edge, using *pygrid*, which is a novel development. However, even the latest two platforms can be seen, primarily, as tools for studying the "nature of FL", rather than to be used to run FL in IoT ecosystems.

In this context, this work aims to (a) reflect on the nature of challenges that actual FL deployments in IoT have to address, (b) show how a reference architecture, proposed for Next-Generation IoT supports the deployment of Federated Learning, and (c) illustrate the flexibility of the proposed approach through its capability of setting systems with different FL topologies. Hence, the remaining parts of this work are organized as follows. In Section II a practical IoT-based scenario from ASSIST-IoT [1] project is described. Since FL will be actually deployed and experimented with in this use case, it will be used to summarize key requirements for "practical FL in IoT". In Section III we summarize pertinent state-of-the-art. Following, in Section IV, an architecture that fulfills the requirements of the use case and addresses issues materializing in IoT-based deployments is described. Next, in Section V, the ways in which the proposed architecture can be adapted and used are outlined. Finally, in Section VI, a summary of contributions, and directions of future work are provided.

II. Federated Learning use case in IoT deployment

The foundation of this contribution is provided by ASSIST-IoT project. There, a sample use case, in which FL is to be applied, is a part of the car damage recognition pilot.

The main goal of this scenario is to provide a fast and accurate inspection of car exterior damage, with minimal data transfer from edge devices to the cloud. Here, the task of car damage detection can be separated into three steps: (i) efficiently separating the vehicle from the background, (ii) vehicle part segmentation, and (iii) automatic defect detection. The results are to be used to support expert-delivered-evaluation, and to facilitate decisions involving insurance claims, as well as car return or leasing services.

As it can be seen in Fig. 1, the functional pipeline involves multiple professional scanners, equipped with high-quality cameras, based on the TwoTronic solution [2]. A high volume (more than 200) of scanned vehicles per day is expected. TwoTronic scanners, and "attached" medium-class computers, will serve as FL clients. The FL server will be located in an external data center in Nürnberg, Germany.

Fig. 1: Car damage recognition - scanner gate

Deploying a FL system is a complex task, depending not only on the availability of FL libraries and algorithms but also characteristics and limitations of a distributed system. Importantly, to be able to practically apply FL solution in this real-life use case, additional issues that are rarely addressed in literature, such as: (a) sudden user dropout, (b) weak network connection with potential interruptions, (c) geographical constraints (leading to unequal groups of clients), (d) data distribution (local distribution on the client differing from global distribution, with no additional public information that would enable problem mitigation through client grouping), and (e) system limitations, notably available RAM and number of cores, need to be considered. Lastly, (f) in environments with heterogeneous devices, interoperability may also become an issue.

It is worth noting that, due to (geographical) distances between scanners and the FL server, located in Nürnberg, as well as the high speed and accuracy of prediction, necessary for this scenario, examining different Fl topologies may be in order. First, divergence from a centralized (client-server) schema to a decentralized one could protect the system from having a single point of failure. This could increase its reliability and resilience. Second, the introduction of additional

[1] https://assist-iot.eu/

[2] https://www.fahrzeugscanner.de/

aggregating clients into a centralized system, would mean that more information about an interrupted training is being preserved. Moreover, training could continue within lower levels of aggregating hierarchy. Additionally, a decrease in direct communication between scanners and the central server could mean faster training. The employment of non-standard topologies may also reduce sensitivity of the training process to interruptions and sudden client dropouts; by introducing additional communication channels.

III. Work related to Federated Learning topology

Taking into account potential importance of FL topology, let us summarize related state-of-the-art. Currently, the effect of topology between clients on FL systems is not fully understood, but hard to deny [21]. For sure, there is no "best topology", but rather it needs to be selected to match the characteristics of a specific use case. It has been observed that the centralized approach may not be appropriate, due to significant communication overhead and a single point of failure [22]. On the other hand, fully decentralized topologies can involve a significant cost of communication not related to client-to-server one [23]. It is worth mentioning that some works combine these approaches to improve convergence and scalability, for example by combining decentralized groups with a centralized update schema [24].

Some approaches experimented with star and ring architectures and their combination. The reason was to avoid the communication bottleneck of the former while gaining improved scalability and accuracy of the latter [25]. There, a star architecture with ring-based groups, supported by a self-balancing framework designed to mitigate the problem of a skewed global distribution, was evaluated.

Work presented in [26] uses a ring architecture with star-based groups, in a realistic use case with non-IID data with periodic variance. Overall, while a linear speedup with respect to the number of clients is reported, the need for periodic variance is a limiting factor.

Ring-based groups, without global communication, while further elaborating on the periodically variational distribution of the data samples, treated by semi-cyclic Stochastic Gradient Descent (SGD) is discussed in [27]. Here, it is observed that the use of ring-based groups may lead to slower training due to the higher number of rounds the process has to undergo for the model to gather information from all the nodes belonging to the group when compared with star-based groups.

Work reported in [24] investigates combinations of star and ring architectures and proposes two forms of the TornadoAggregate algorithm: one with a ring architecture with star-based groups, the other with a star architecture with ring-based groups. Interestingly, a substantial difference in results between the two TornadoAggregate versions is reported. The version with star architecture and ring-based groups, outperformed the ring architecture with star-based groups.

In [21] D-Cliques, a topology that aims at reducing gradient bias, by grouping clients in sparsely interconnected cliques, such that the label distribution in the clique would be representative of the global distribution, is presented. This approach led to the convergence speed similar to that of a fully-connected topology with a 98% reduction in the total number of edges, and 96% reduction in the total number of messages.

A contrasting approach to data skewness mitigation, in the form of a hierarchical FL system with Federated Gradient Descent being conducted on the user-edge layer and Federated Averaging between edges and the cloud, is presented in [28]. The resulting architecture is designed with an IoT environment in mind, with the potentially less efficient connections between edge and the server supporting less frequent communication.

Work described in [29] uses segmentation to allow for large model training on far edge. The proposed approach relies on a combination of model segmentation level synchronization mechanisms, which divides the model into a set of not overlapping subsets, and a decentralized design reminiscent of the gossip protocol, with each worker randomly transferring the model segment to a few other workers. Model redundancy had to be included in order to ensure convergence. Discussed prototype acknowledges the problem of workers suddenly exiting and returning. This work has been further extended in [23], forming a bandwidth-aware solution by greedily choosing a client with sufficient bandwidth to avoid delays. The convergence guarantees were provided, with the training time being reduced up to 18 times, compared to that of baselines with no accuracy degradation.

Another approach to decentralized FL (DAFCL) can be found in [30]. In DAFCL, all clients are connected through an undirected graph. Each of them is supposed to train the model based on its local data, and exchange the results with its neighbors, through a symmetric doubly stochastic matrix. To avoid a single point of failure, the average model estimation is tracked using First Order Dynamic Average Consensus (FODAC). This architecture shows promising results. Nevertheless, to use it in Next Generation IoT environments further work on communication efficiency, and increasing resilience to sudden catastrophic events, such as user dropout, would be necessary.

In summary, research related to FL topology introduces a multitude of approaches to the problem. from the perspective of this contribution, it "does not matter" which topology should be used or is the best in a given scenario. The question is: how to make sure that any needed topology can be instantiated in Next Generation IoT Ecosystems. Proposing a pathway to answering this question is the goal of the remaining parts of this contribution.

IV. Federated Learning in IoT – proposed architecture

Let us now introduce the proposed architectural approach to Federated Learning in IoT ecosystems. Since support for different topologies has been shown to be important in large-scale real-life deployments, the possibility of easily implementing them is crucial. Moreover, the proposed architecture should

be resistant to sudden user dropout, network connection with interruptions or uneven grouping of clients.

The proposed FL architecture is developed according to the Reference Architecture (RA) introduced in the ASSIST-IoT project, and motivated by real-life scenarios, coming from three industrial pilots. This RA is based on the concept of encapsulation, in which is instantiated in the form of enablers. Interested readers should consult [31] for necessary details. Note that the fact that the proposed FL architecture is compatible with ASSIST-IoT RA principles allows the use of additional enablers that can extend its capabilities, e.g., with a semantic toolset to enable interoperability, or self-* functionalities such as automated configuration (e.g., to control the state of topology and adjust its configuration) [32]. These aspects are, however, outside of the scope of this contribution.

As it can be seen in Fig. 2, the proposed FL architecture is formed by four enablers: *FL Orchestrator*, *FL Repository*, *FL Training Collector*, and *FL Local Operations*. The *FL Orchestrator* is the enabler responsible for the configuration propagation to other enablers, workflow management, and control over the FL life cycle. It also acts as the entrance gate for human interactions. Moreover, *FL Orchestrator* may control the FL training process, and constraints related to e.g., the minimum number of clients, or minimum system requirements. On the other hand, the *FL Repository* is a supplementary enabler for storing models, algorithms, and any data needed in the FL process. Last, the *FL Training Collector* and *FL Local Operations* act as FL servers and clients, respectively. They are used in the constructed system as communicating components, remaining in constant contact according to the gRPC protocol, by utilizing functionalities implemented as a part of the Flower library [33]. In other words, the *FL Training Collector* possesses the capabilities of a FL centralized server, while the *FL Local Operations* (located on edge clients) has the abilities of an FL client, with the main focus placed on local model training and dataset loading. Let us now describe the *FL Training Collector* and the *FL Local Operations* in more detail.

A. FL Training Collector

FL Training Collector mainly serves the role of a server node. Uploading configuration (e.g. from the *FL Repository*) initiates the training process. The configuration data can include, among others, the type of aggregation algorithm used for FL, the minimal number of clients necessary in order to start training, the minimum number of clients necessary for training each round, the fraction of clients to be sampled for training or evaluation, a set timeout for the responses coming from clients, the number of clients to choose for training with blacklisting and some additional values used for later testing. The behaviour exhibited by the *FL Training Collector* before and after each training, as well as evaluation round, is defined in the form of a Strategy class, in accordance with the requirements of the Flower library. This class is used by the Flower server to group clients, selected from available client interfaces, with the appropriate weights to be later sent by the server and to define the mechanisms used to aggregate results from the clients and evaluate current model performance. Due to its periodic nature (methods are called in the defined order, before and after every round), this class is also used for gathering metrics and saving current model weights, for later analysis. The metrics, which are gathered after each training round, consist of aggregated evaluation loss, global evaluation loss, and global accuracy. They are collected in order to facilitate monitoring of the training process. Later, they are locally stored in the enabler in the form of a serialized object inside a pickle file [34].

B. FL Local Operations

An instance of *FL Local Operations*, the analogue for the FL client, is created similarly to the *FL Training Collector*. In order to start the training, it needs to be provided with a training configuration, and the address of the *FL Training Collector* instance, which it should be connected to.

FL Local Operations enabler is responsible for loading and preprocessing the right subset of local data, and setting up the local model. It not only executes but also enhances the behaviour of an FL client in the form of classes extending the flower.client.Client class, by implementing methods of initiating, fitting the model, and evaluating the model performance. The evaluation accuracy and loss of the current model are computed on the local test set. The values of these metrics, in their original form, as well as an average (in the case of clustered architecture – weighted average, in an attempt to increase the precision of the visualization, for unstable client groupings) over the metric values from all *FL Local Operations* is used to assess the efficiency of the training process. Similarly to *FL Training Collector*, these statistics are regularly stored as pickle files [34]. *FL Local Operations* may also include mechanisms related to privacy, such as data encryption or differential privacy [4], [5].

C. FL training process

Let us now describe the FL training process that is to take place in the case of basic, centralized, topology.

1) An instance of *FL Training Collector* receives a training configuration from the *FL Orchestrator*.
2) *FL Training Collector* waits for a minimal number of clients, as specified by the configuration.
3) Required number of *FL Local Operations* instances receive their training configuration, from the *FL Orchestrator*, similar in content to that supplied to the *FL Training Collector*, but also including identifying information about the *FL Training Collector* participating in the process.
4) Activated instances of *FL Local Operations* establish a connection with the *FL Training Collector*.
5) *FL Training Collector* samples *FL Local Operations* and provides them with model weights and, possibly, additional configuration, which triggers the training process on *FL Local Operations*.

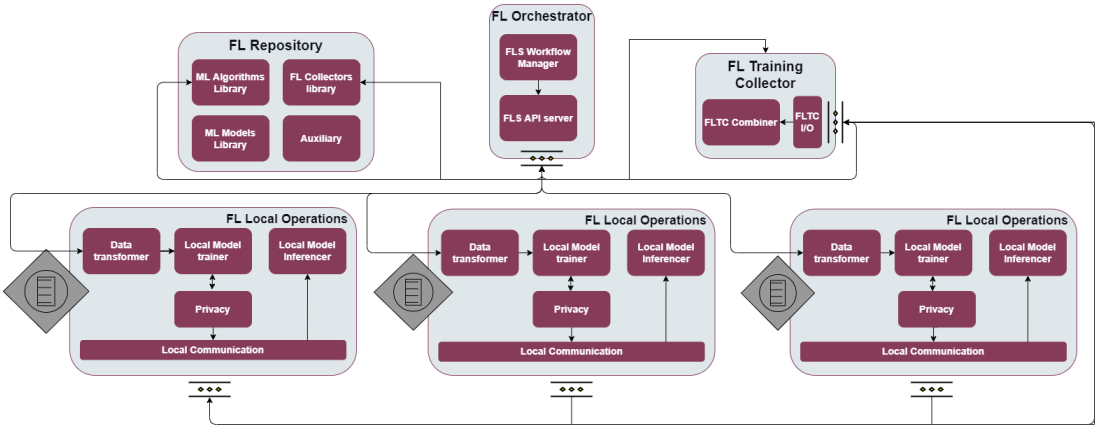


Fig. 2: Proposed FL in IoT architecture

6) *FL Local Operations* instances train the model (in parallel) and return the weights along with any metrics they were requested to gather.
7) Next, the weights are aggregated according to a strategy supplied by the *FL Training Collector*. The data, along with any computed metrics, is communicated (as required) before and after model evaluation (after each round).

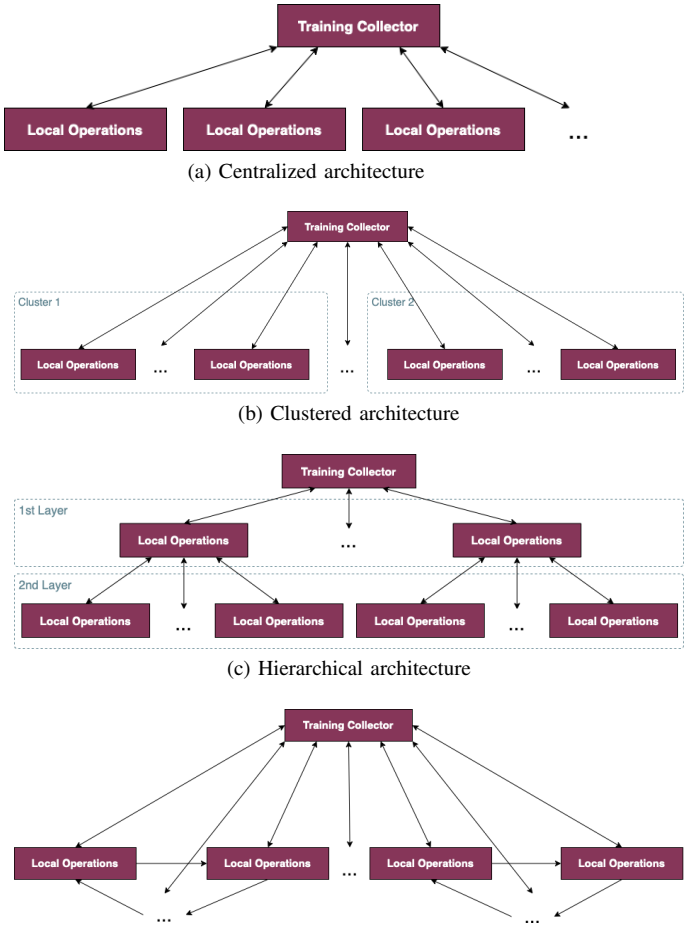
This approach, formulated for the basic centralized architecture, can be then modified in order to support other topologies.

V. Other FL topologies, applicability and usability

By performing slight modifications to the basic architecture, it is possible to instantiate other topologies proposed in the literature. In particular, four topologies have been implemented and initially tried: centralized architecture, clustered architecture, hierarchical architecture, and star architecture with ring-based groups. They are illustrated in Fig. 3. It should be noted that the aim of this work was to establish that the proposed architectural approach, based on enablers originating from the ASSIST-IoT RA can be used to easily set up "any" FL topology. Thus, this is what was implemented and tested. The usage of these topologies for the car maintenance use case, described above, will be explored in the near future.

The basic centralized architecture was implemented following the description presented above. The possibility of using different "parameters" of the FL process, as represented in the setup, including client numbers, model architecture, approach to model averaging, data collected by the *FL Local Operations* and *FL Training Collector* has been tested.

As for the clustered architecture, the implemented version, first, accepts a set number of clusters and then uses the Iterative Federated Clustering Algorithm (IFCA) [35] to dynamically determine the adherence of a given client to a cluster at the beginning of each round. Next, in the aggregation stage, the cluster models are updated, based only on the data from the clients that belong to them at the moment. When faced with IID data, the clients are determined to belong to a



(a) Centralized architecture

(b) Clustered architecture

(c) Hierarchical architecture

(d) Star architecture with ring-based groups

Fig. 3: ASSIST-IoT FL alternative architectures for IoT environments

single cluster, which leads the architecture to behave similarly to the centralized one. For the non-IID data, the clustered architecture leads to the development of a number of models, each tailored exactly to a given cluster of clients, instead of a single global solution. The implemented architecture was tested using CIFAR-10 (for IID data) and German Traffic Sign Recognition Benchmark dataset (for non-IID data) and the results matched these found in the literature [36], [37].

The hierarchical topology necessitates the creation of an additional component [28]. In this work it is implemented as a special case of *FL Local Operations* called *1st Layer Local Operations*. This necessitates that the version of *FL Local Operations* acts as a basic FL client called *2nd Layer Local Operations*. This additional enabler serves as FL server to *2nd Layer Local Operations* and as FL client to the *FL Training Collector*, aggregating the updates from the *2nd Layer Local Operations* for a set number of local rounds, and afterwards propagating them to the global *FL Training Collector* for aggregation. Again, the instantiated, hierarchical, topology was tested using the CIFAR-10 and German Traffic Sign Recognition Benchmark dataset and obtained results matched these reported in the literature [36], [37].

In yet another experiment, the star topology with ring-based groups introduced decentralized elements, based on the Tornadoes architecture [24]. Here, the training process starts with the *FL Training Collector* sending the initial model to all available *FL Local Operations*. Then, the *FL Local Operations* uses every local round to train the model on its local data to pass it to the next instance belonging to its ring-based group, and accept an incoming model from the previous instance, for further training. After a given number of local rounds a global aggregation (performed by the *FL Training Collector* occurs. As in previous cases, the constructed topology was tested (on the CIFAR-10 and German Traffic Sign Recognition Benchmark datasets) and obtained results match these found in [36], [37].

Finally, Fig. 4 presents a solution envisioned for the use case described in Section II using enablers from the proposed architecture. Here, we use a centralized topology where *FL Local Operations* are run on clients (cameras). *FL Orchestrator*, *FL Training Collector* and *FL Repository* are located in the cloud. This environment is going to be somewhat more "stable", because there is a predefined number of clients in the business environment. The main goal is to distribute the processing, instead of sending all the images to the cloud and processing it centrally. Here, although the centralized topology seems to be a good choice for initial implementation, it is clear that a more complex topology will ultimately be needed. One of the reasons is that in extended deployment groups of scanners (one or more) may belong to different stakeholders. Therefore, a hierarchical topology would be a natural choice. Nonetheless, it has been already established (above) that such topology is easy to deliver using the existing set of enablers.

On the diagram, besides FL enablers, additional enablers designed and implemented within ASSIST-IoT (following ASSIST-IoT RA) are included addressing: cybersecurity (specifically authentication and authorization), *Long Term Storage* enabler (that can provide local storage of images for FL clients), and *Tactile Dashboard* (for visualizations needed in the system). These elements can provide all additional functions needed in the ecosystem.

VI. Concluding remarks

Even though there is a lot of research in the field of FL, most of it is devoted to FL processes, algorithms, or specific aspects such as data security. Here, we try to address issues related to the deployment of FL system in a real-life use case in an IoT ecosystem. This requires the choice of an appropriate architecture. In this context, the ASSIST-IoT RA was extended to deliver a set of enablers that allow easy configuration of FL system with machine learning parameters, as well as any required topology. Moreover, additional enablers, created for the RA allow turning the FL process into a complete, robust solution.

References

[1] Q. Xia, W. Ye, Z. Tao, J. Wu, and Q. Li, "A survey of federated learning for edge computing: Research problems and solutions," *High-Confidence Computing*, vol. 1, no. 1, p. 100008, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S266729522100009X

[2] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A big data - ai integration perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1328–1347, 2021.

[3] B. Murdoch, "Privacy and artificial intelligence: Challenges for protecting health information in a new era," *BMC Medical Ethics*, vol. 22, no. 1, 2021.

[4] H. Zheng, H. Hu, and Z. Han, "Preserving user privacy for machine learning: Local differential privacy or federated machine learning?" *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 5–14, 2020.

[5] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X20329848

[6] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[7] Q. Hu, P. Sun, S. Yan, Y. Wen, and T. Zhang, "Characterization and prediction of deep learning workloads in large-scale GPU datacenters," *CoRR*, vol. abs/2109.01313, 2021. [Online]. Available: https://arxiv.org/abs/2109.01313

[8] X. Qiu, T. Parcollet, J. Fernández-Marqués, P. P. B. de Gusmão, D. J. Beutel, T. Topal, A. Mathur, and N. D. Lane, "A first look into the carbon footprint of federated learning," *CoRR*, vol. abs/2102.07627, 2021. [Online]. Available: https://arxiv.org/abs/2102.07627

[9] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016. [Online]. Available: http://arxiv.org/abs/1602.05629

[10] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. Vincent Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021.

[11] S. Kumar, R. Schlegel, E. Rosnes, and A. G. i. Amat, "Coding for straggler mitigation in federated learning," 2021.

[12] A. Tak and S. Cherkaoui, "Federated edge learning: Design issues and challenges," *IEEE Network*, vol. 35, no. 2, pp. 252–258, 2021.

[13] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," *CoRR*, vol. abs/2009.13012, 2020. [Online]. Available: https://arxiv.org/abs/2009.13012

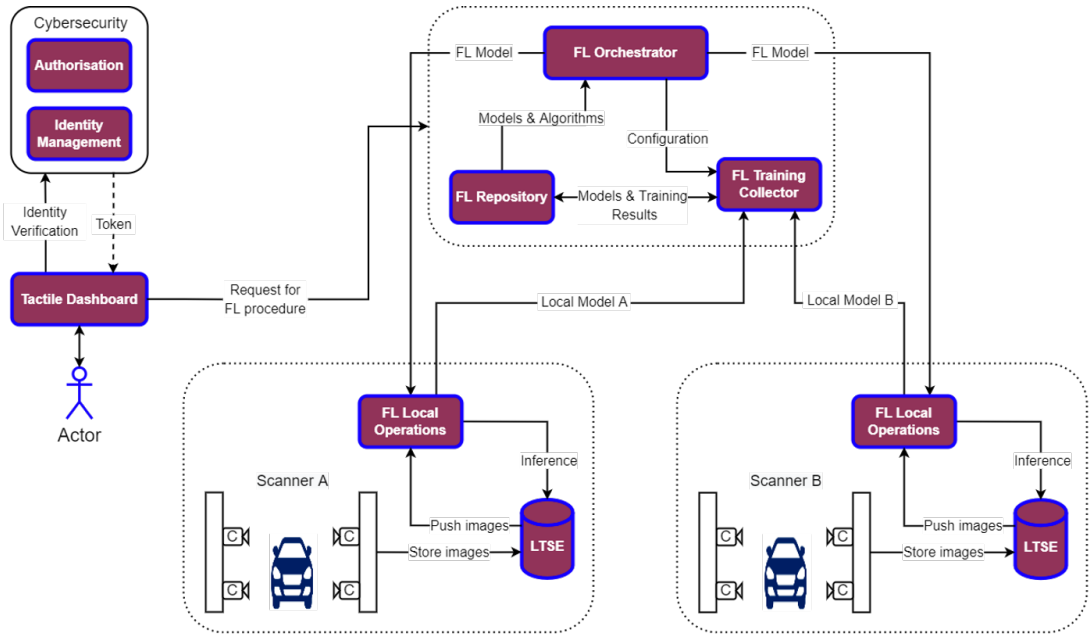


Fig. 4: FL architecture for the car damage use case

[14] J. Cao, Q. Zhang, and W. Shi, *Challenges and Opportunities in Edge Computing*. Cham: Springer International Publishing, 2018, pp. 59–70.

[15] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms," *ACM computing surveys*, vol. 52, no. 5, pp. 1–37, 2019.

[16] *TensorFlow Federated: Machine Learning on Decentralized Data*, accessed in 2022. [Online]. Available: https://www.tensorflow.org/federated?hl=en

[17] *An Industrial Grade Federated Learning Framework*, accessed in 2022. [Online]. Available: https://fate.readthedocs.io/en/latest/

[18] I. Kholod, E. Yanaki, D. Fomichev, E. Shalugin, E. Novikova, E. Filippov, and M. Nordlund, "Open-source federated learning frameworks for iot: A comparative review and analysis," *Sensors*, vol. 21, p. 167, 12 2020.

[19] D. J. Beutel, T. Topal, N. D. Lane, A. Mathur, T. Parcollet, and X. Qiu, *Flower: A Friendly Federated Learning Framework reference manual*. [Online]. Available: https://flower.dev/docs/

[20] OpenMined, *PySyft*, accessed in 2022. [Online]. Available: https://blog.openmined.org/tag/pysyft/

[21] A. Bellet, A. Kermarrec, and E. Lavoie, "D-cliques: Compensating noniidness in decentralized federated learning with topology," *CoRR*, vol. abs/2104.07365, 2021. [Online]. Available: https://arxiv.org/abs/2104.07365

[22] L. Chou, Z. Liu, Z. Wang, and A. Shrivastava, "Efficient and less centralized federated learning," *CoRR*, vol. abs/2106.06627, 2021. [Online]. Available: https://arxiv.org/abs/2106.06627

[23] J. Jiang, L. Hu, C. Hu, J. Liu, and Z. Wang, "Bacombo—bandwidth-aware decentralized federated learning," *Electronics*, vol. 9, no. 3, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/3/440

[24] J. Lee, J. Oh, S. Lim, S. Yun, and J. Lee, "Tornadoaggregate: Accurate and scalable federated learning via the ring-based architecture," *CoRR*, vol. abs/2012.03214, 2020. [Online]. Available: https://arxiv.org/abs/2012.03214

[25] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-balancing federated learning with global imbalanced data in mobile systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 59–71, 2021.

[26] Y. Ding, C. Niu, Y. Yan, Z. Zheng, F. Wu, G. Chen, S. Tang, and R. Jia, "Distributed optimization over block-cyclic data," *CoRR*, vol. abs/2002.07454, 2020. [Online]. Available: https://arxiv.org/abs/2002.07454

[27] H. Eichner, T. Koren, H. B. McMahan, N. Srebro, and K. Talwar, "Semi-cyclic stochastic gradient descent," *CoRR*, vol. abs/1904.10120, 2019. [Online]. Available: http://arxiv.org/abs/1904.10120

[28] N. Mhaisen, A. A. Abdellatif, A. Mohamed, A. Erbad, and M. Guizani, "Optimal user-edge assignment in hierarchical federated learning based on statistical properties and network topology constraints," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 55–66, 2022.

[29] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," *CoRR*, vol. abs/1908.07782, 2019. [Online]. Available: http://arxiv.org/abs/1908.07782

[30] Z. Chen, D. Li, J. Zhu, and S. Zhang, "Dacfl: Dynamic average consensus based federated learning in decentralized topology," 2021.

[31] A. Fornés-Leal, I. Lacalle, C. E. Palau, P. Szmeja, M. Ganzha, M. Paprzycki, E. Garro, and F. Blanquer, "Assist-iot: A reference architecture for next generation internet of things," in *Proceedings of The 21st International Conference on Intelligent Software Methodologies, Tools, and Techniques, IN PRESS*, 2022.

[32] K. Nalinaksh, P. Lewandowski, M. Ganzha, M. Paprzycki, W. Pawłowski, and K. Wasielewska-Michniewska, "Implementing autonomic internet of things ecosystems – practical considerations," in *Parallel Computing Technologies*, V. Malyshkin, Ed. Cham: Springer International Publishing, 2021, pp. 420–433.

[33] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, and N. D. Lane, "Flower: A friendly federated learning research framework," *CoRR*, vol. abs/2007.14390, 2020. [Online]. Available: https://arxiv.org/abs/2007.14390

[34] G. Van Rossum, *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[35] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," 2020. [Online]. Available: https://arxiv.org/abs/2006.04088

[36] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," 2014. [Online]. Available: https://arxiv.org/abs/1412.6806

[37] M. Thoma, "Analysis and optimization of convolutional neural network architectures," 2017. [Online]. Available: https://arxiv.org/abs/1707.09725