Open Call 3

Fleshnet

Deliverable 3: Experiment Results and Final Report

Authors	Felix Freitag, UPC; Lu Wei, TTU; Chun-Hung Liu, MSU
Due Date	20220331
Submission Date	20220331
Keywords	federated learning, testbed, experimentally-driven research

Deliverable 3: Part I

Analysis, results, and wider impact

1 Abstract

The overall goal of the FLESHNET project is to deliver building blocks for adaptive decentralized federated learning experimentally validated in the realistic conditions of a wireless mesh network testbed. Both the EU and US partners will participate in the analysis of the experimental results. The design will be complemented by the US partners with the networking perspective, while the EU partner will complement the design with an adaptive and decentralized model.

2 Project Vision

The increase of the computing capacity of end user-owned devices and the appearance of lightweight machine learning frameworks have led to the situation that machine learning is nowadays available at user devices, while just a few years ago machine learning was exclusive to cloud providers. Federated learning has recently appeared as a promising machine learning approach which respects the privacy of the data. We envision novel federated learning applications in which any node can participate in the provision of data and in the training of machine learning models. For this to happen, federated learning building blocks must be extended and evaluated. These building blocks will enable the autonomy of the participants for taking self-determined decisions, facilitate the ownership of machine learning models and data, and enable decentralized governance of the data. This

scenario will be realistic in the near future as edge nodes of a federated learning network will receive an ever increasing amount of data coming from more and more nearby sensors.

3 Details on participants (both EU and US)

US:

Lu Wei is an assistant professor at the Department of Computer Science at Texas Tech University (TTU). He obtained a Ph.D. degree from Aalto University, Finland, in 2013. He held postdoctoral positions at University of Helsinki, Finland, from 2013 to 2015 and at Harvard University, USA, from 2015 to 2016.

The expertise of the TTU team is on the performance analysis in wireless networks. The expertise has been applied in FLESHNET to explain and understand the obtained experimental resultes with regards to the underlaying communication network, and for contributing suggestions and ideas on how to integrate network awareness in the design for adaptive behaviour of the federated learning components, and helping in the design of experiments for the validation.

Chun-Hung Liu is an assistant professor at the Department of Electrical and Computer Engineering of MSU. He obtained a Ph.D.degree from University of Texas at Austin, USA, in 2011.

The MSU team has a strong track record in the design and analysis of experiments in wireless networks, with the current focus being on machine learning based experiments for emerging networks including 5G/6G systems. The knowledge has been brought in to analyze the experimental results, to review the applied designs, protocols and algoritms and to make suggestions for the design of more adaptive federated learning architectures, along with propositions for the experimental settings.

EU:

Felix Freitag is an associate Professor at the Department of Computer Architecture of UPC. He obtained a PhD degree from the UPC in 1998. He coordinated the FP7 CLOMMUNITY project (2013-2015). His recent research focuses on community clouds and federated learning. Felix supervised several PhD theses. The full list of his publications is available at https://futur.upc.edu/FelixursErichFreitag. The PI of the UPC team is supported in the work for Fleshnet by members of the research group, among them Roc Meseguer and Leandro Navarro, associate professor and full professor, respectively, contributing to the research, and Pedro Vílchez, research support engineer with several years of professional experience, working on the maintenance of the computing infrastructure and software related to the testbed nodes. Leandro Navarro is full Professor at the Department of Computer Architecture at UPC and director of the distributed systems group at UPC. His research interests include distributed computing

Co-funded by the Horizon 2020
Framework Programme of the European Union

systems such as network and cloud infrastructures. Roc Meseguer is an associate Professor at the Department of Computer Architecture of UPC. His research interests include socio-economic decentralized systems and ambient intelligence.

UPC has lead the experimental evaluation within Fleshnet that is done in a testbed deployed within a wireless city mesh network called GuifiSants. New experiment executions done by UPC have been preceeded by analyzing among the project partners the results from previously obtained experimental data and by discussing new design to research the adaptive and decentralized federated learning architecture. UPC implemented the new designs and managed the experimental executions as well as the testbed nodes.

4 Results

In FLESHNET we studied the federated learning process in an heterogeneous environment in order to identify more adaptive designs for exploiting this heterogeneity. Federated learning uses a distributed computing infrastructure, therefore heterogeneity can be expected in several forms: 1) The quantity and quality of local data at each node may vary. For instance, local data may be acquired by sensors at or nearby the nodes, but the local circumstances of each node lead to a different number of training samples. 2) The computing capacity of the nodes can be different either by the proper hardware of the nodes or by concurrent executions of other applications at the node which reduce the available computing resources dedicated to the federated learning process. 3) For doing the exchange of the trained model some nodes may face limited network capacities, either due to permanent or dynamic network conditions.

We experimented with federated learning on distributed computing devices consisting of mini-PCs or Single-Board-Computers (SBCs), such as those found in home environments. This scenario represents user environments where computing devices run as home servers to manage several user-oriented services. In this situation the devices are not dedicated exclusively to a machine learning application. The main obtained results can be summarized as:
1) We introduce the deployment of federated learning in a wireless mesh network.
2) We develop the design for adaptive decentralized federated learning components.
3) We provide the evaluation of the federated learning process in distributed low-capacity devices connected to a wireless mesh network.

In the following we detail these results.

Scenario

Co-funded by the Horizon 2020
Framework Programme of the European Union

Within the GuifiSants wireless mesh network, we have computing nodes connected to several of the routers which are part of the network. We use these nodes as testbed for the experimentation. Figure 4.1 illustrates the deployed testbed.

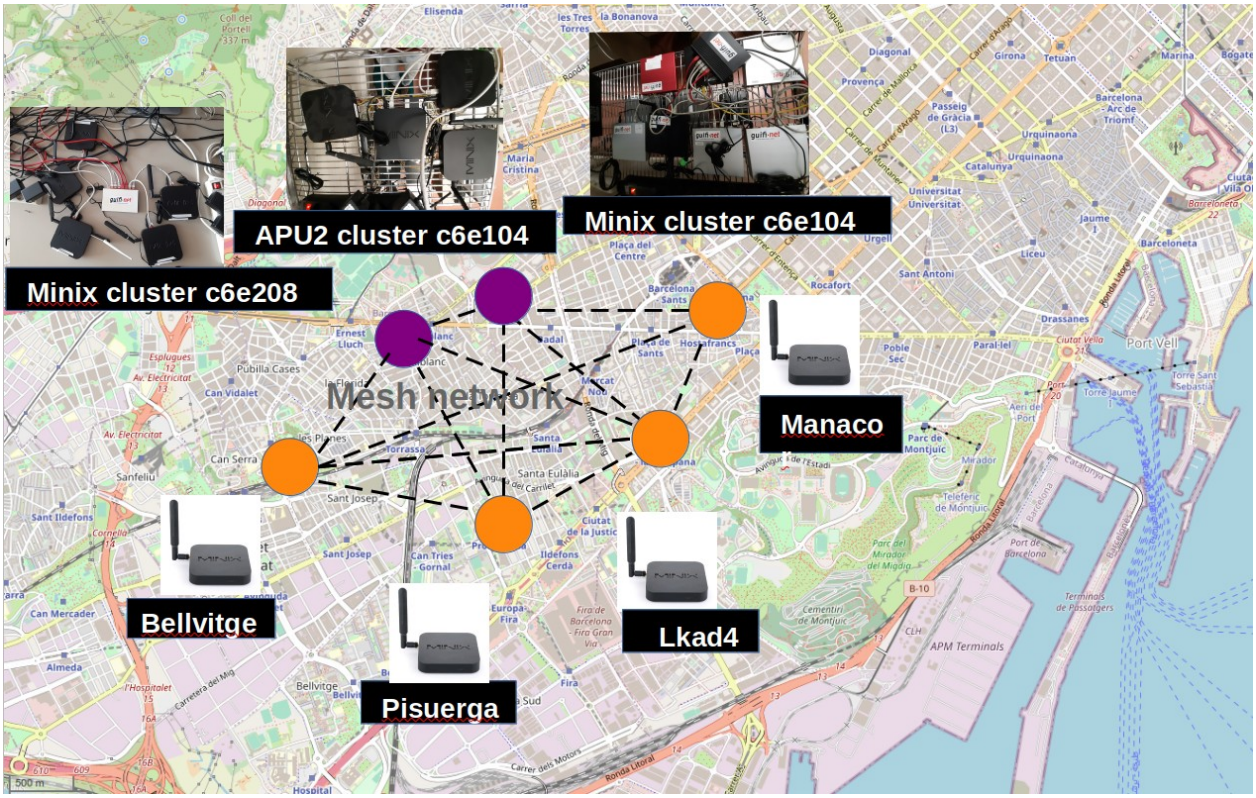

Figure 4.1 Testbed used for the Fleshnet experiments.

We conducted measurements to obtain the network characteristics of the testbed. Table 4.1 and Table 4.2 show the bandwidth between the testbed nodes. Table 4.3 shows the number of hops between the testbed nodes. Table 4.4 shows the round trip time (rtt) between the nodes. It can be seen that the network connectivity between the nodes is heterogeneous, which allows to experiment with situations of different bandwidth conditions.

	Bandwidth			
	Bellv	Ac6e104	Lkad4	Mana
Bellv	2146/2045	28.5/22.2	16.6/8.68	1.43/0.96
Ac6e104	19.0/32.1	4448/4871	18.3/40.9	1.19/1.74
Lkad4	11.5/10.0	43.1/50.8	7544/8387	0.59/0.74
Mana	0.46/1.12	0.84/0.30	0.53/0.35	7283/7377

Table 4.1 Throughput between some of the testbed nodes in Mbps and measured with iperf3.

Co-funded by the Horizon 2020 Framework Programme of the European Union

	Bandwidth		
	Mc6e104	Mc6e208	Pisu
Mc6e104	7519/8357	7.76/4.26	4.31/3.18
Mc6e208	3.11/4.74	7526/8129	3.06/3.75
Pisu	6.48/7.37	5.01/2.36	8057/7712

Table 4.2 Throughput between some of the testbed nodes in Mbps and measured with iperf3.

	Bandwidth						
	Bellv	Ac6e104	Lkad4	Mana	Mc6e104	Mc6e208	Pisu
Bellv	1	6	7	7	6	7	8
Ac6e104	6	1	4	5	2	4	6
Lkad4	8	3	1	7	4	3	4
Mana	7	5	6	1	5	6	7
Mc6e104	6	2	4	5	1	4	6
Mc6e208	8	3	3	7	4	1	4
Pisu	10	5	5	9	6	5	1

Table 4.3 Number of hops between testbed nodes measured with the traceroute tool.

	Bandwidth						
	Bellv	Ac6e104	Lkad4	Mana	Mc6e104	Mc6e208	Pisu
Bellv	0.231	5.984	13.529	118.117	8.100	290.031	33.287
Ac6e104	5.546	0.250	14.635	132.146	1.386	5.727	10.171
Lkad4	16.429	6.393	0.158	23.896	8.100	13.863	90.071
Mana	378.252	135.208	57.028	0.124	96.098	28.198	70.592
Mc6e104	5.421	0.851	8.363	57.217	0.109	10.168	59.717
Mc6e208	20.844	17.436	72.954	56.378	24.437	0.115	143.293
Pisu	41.214	12.548	51.744	69.809	46.654	21.932	0.120

Table 4.4 Average of round-trip delay time in ms between testbed nodes measured with the ping tool.

We consider a scenario where a server conducts federated learning rounds with a set of heterogeneous clients. This heterogeneity can be due different bandwidth, different client hardware, and different computing capacity usage according to the policy of each client node.

Specific software was installed on each testbed node including Docker to instantiate Docker containers for running the code for the federated learning experiments. Prometheus-Grafana support was installed to help in the testbed monitoring and experimental evaluations, besides other specific Linux command line tools for measuring the resouce consumption and other metrics. To ease the experimentation, we installed a local Docker

Co-funded by the Horizon 2020 Framework Programme of the European Union

registry and a Debian repository proxy within Guifi.net, which allowed to integrate in the testbed also these nodes that have limited or no Internet access. Thus, newly-built Docker images, which in an experiment were created for testing changes in protocols and algorithms, were pushed to the local Docker registry, and from there they were be pulled by any testbed node within Guifi.net.

Design of adaptive decentralized federated learning

Building blocks for decentralized adaptive federated learning

Federated learning uses a star topology, where a central node, i.e. the federated learning server, orchestrates the training with the registered clients. The role of the federated learning server at the end of a training round is to merge the local models received from the clients into a new global model. For doing the next round, this new model will be sent out to the clients.

Figure 4.2 shows the architectural overview of the building blocks for an adaptive decentralized federated learning network. We added a *dynamic configuration module* to the federated learning server. The dynamic configuration module determines for each round the learning parameters for each client. Therefore, the interaction between server and clients include individual learning parameters which are sent to the clients along with the new global model.

In this scenario with the dynamic configuration module the federated learning configuration for each round is centralized at the server. Then, measurements are taken at the server for each training round, without requiring additional measurements taken at the clients to be transmitted to the server. These measurement feed the dynamic configuration module which returns to the server for each new round new federated learning training parameters.

A key metric that the server calculate is the *workload_rythm*, which is a client-specific metric. The metric is defined as the number of samples by epochs trained divided by the time that passed between sending the global model to the client and receiving the local model back. Since the measurements are taken at the server and not at the client, this metric includes the time spend for the communication of the model, from the client to the server and back, and the time of the proper model training at the client.

We added decision capacity to the client, expressed by the *decentralized federated learning client* in Figure 4.2. Such a client will be able to overwrite the received training parameters with new values, computed locally on the base of local circumstances and conditions.

Co-funded by the Horizon 2020
Framework Programme of the European Union

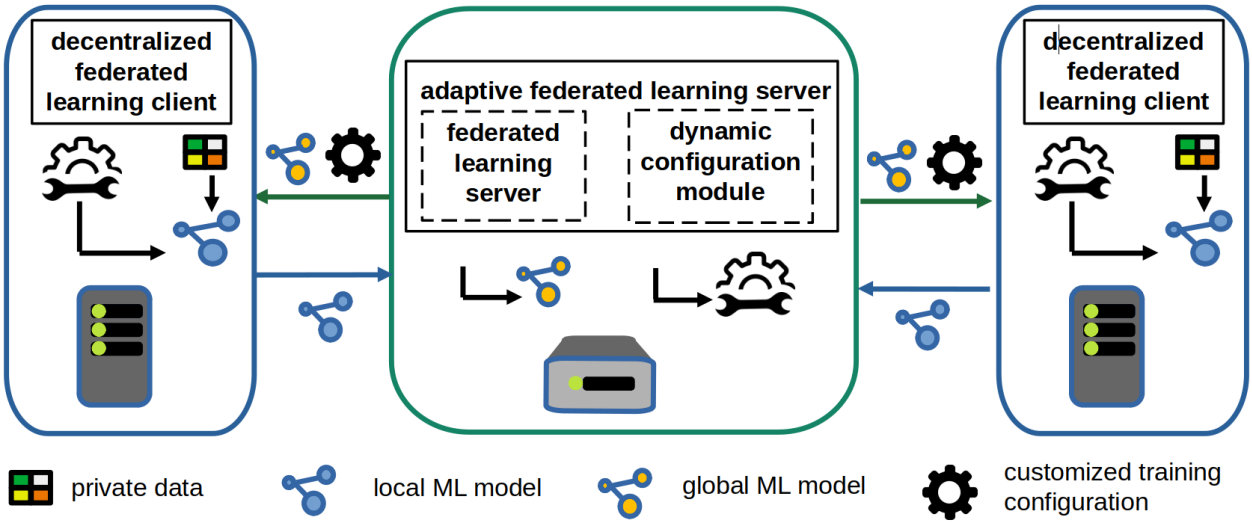

Figure 4.2 Architectural overview of building blocks.

The software implementation of the federated learning network consists of two major components, the code for the client and that of the server, publicly available at https://gitlab.com/dsg-upc/federated_learning. This code integrates the implementation of the dynamic configuration module. We also added measurement capacity and additional REST API endpoints to support the experimental evaluation in a distributed setting. Furthermore, we implemented the code to perform decentralized decisions in the federated learning client.

The communication between server and clients is conducted over the http protocol. The server implements the REST API summarized in Table 4.5. The clients implement the REST API summarized in Table 4.6.

Co-funded by the Horizon 2020 Framework Programme of the European Union

HTTP method	URL	Parameters	Description
POST	/client	client IP	enables the registration of a client at the server
POST	/training	training_type	enables to select between different machine learning models and datasets. Triggers the server to start a training round.
POST	/set_training_config_for_client	epochs, learning rate, batch size, # of training samples, # of testing samples, client IP	indicates the customized training configuration for a specific client. Used by the dynamic configuration module.
PUT	/model_params	local machine learning model, client IP, training_type, performance data	used by the federated learning client to push locally trained model to server. Piggybacks training performance data.
GET	/get_training_clients	N/A	obtains the list of registered clients with their performance metrics. Used by the dynamic configuration module.
DELETE	/client	client IP	deletes a client registered at the server

Table 4.5 Server REST API.

HTTP method	URL	Parameters	Description
POST	/training	global machine learning model, epochs, learning rate, batch size, # of training samples, # of testing samples, client IP, training_type	client receives new machine learning model and the customized training configuration from the server

Table 4.6 Client REST API.

Figure 4.3 shows the interaction between server and client where for the experimental control an external module initiates the required number of federated learning rounds.

Co-funded by the Horizon 2020 Framework Programme of the European Union

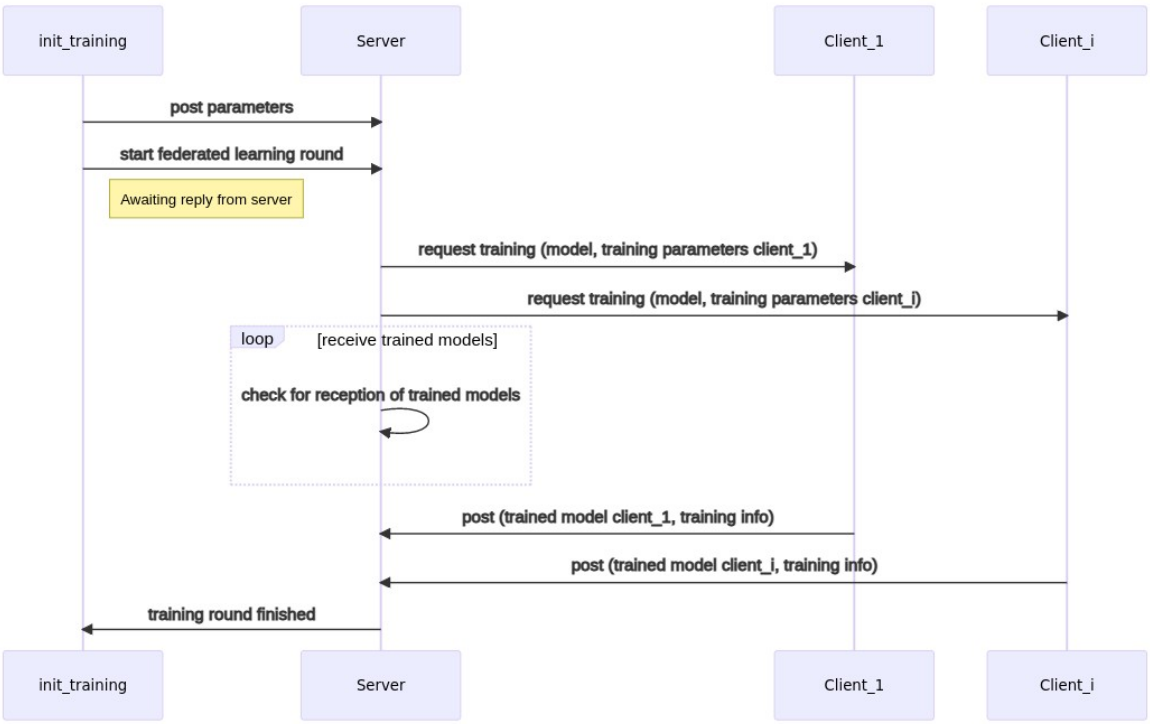

Figure 4.3 Sequence diagram for the main interaction between components.

We designed an algorithm in the dynamic configuration module component of the server that is able to adapt the workload assigned to clients depending on the measured client performance. The dynamic configuration module is invoked every time a federated learning round has finished and determines the individual training parameters for each client for the next round. The objective is to exploit the idle time of fast clients and to assign a higher training effort. Therefore, for determining new parameters, the training performance of the clients is taken into account.

Figure 4.4 shows for the case of two clients the details of the times which are spent when the server does a federated learning round. At the server site the training time for both clients is measured.

This training time for a client i, t_train_client_i, is determined by the time spend for the transmission of the model (t_rx_client_i and t_tx_client_i) and the time to train the model locally (t_compute_client_i). In a heterogeneous environment where clients run on different hardware and where the bandwidth from each client to the server is different, for each client these times will be different while the performing the same local training configuration. For instance, in Fig. 4.4 it can be seen that client_2 has a better bandwidth and the model transmission time is less than in client_1, and also the computing time of client_2 is less.

Co-funded by the Horizon 2020
Framework Programme of the European Union

From the training time t_train_client_i measured at the server, the server computes the *workload_rhythm*. The workload rhythm is defined by:
$workload\_rythm_{cl(i)} = epochs_{cl(i)} * training\_images_{cl(i)} / t_{train\_client(i)}$

The algorithm identifies which is the worst client. It does that by analyzing the mean workload rhythm of each client in the last 3 rounds. Once determined, it iterate through all other (faster) clients and assigns a suitable workload with which they can trained in the next round within the training time of the slowest client.

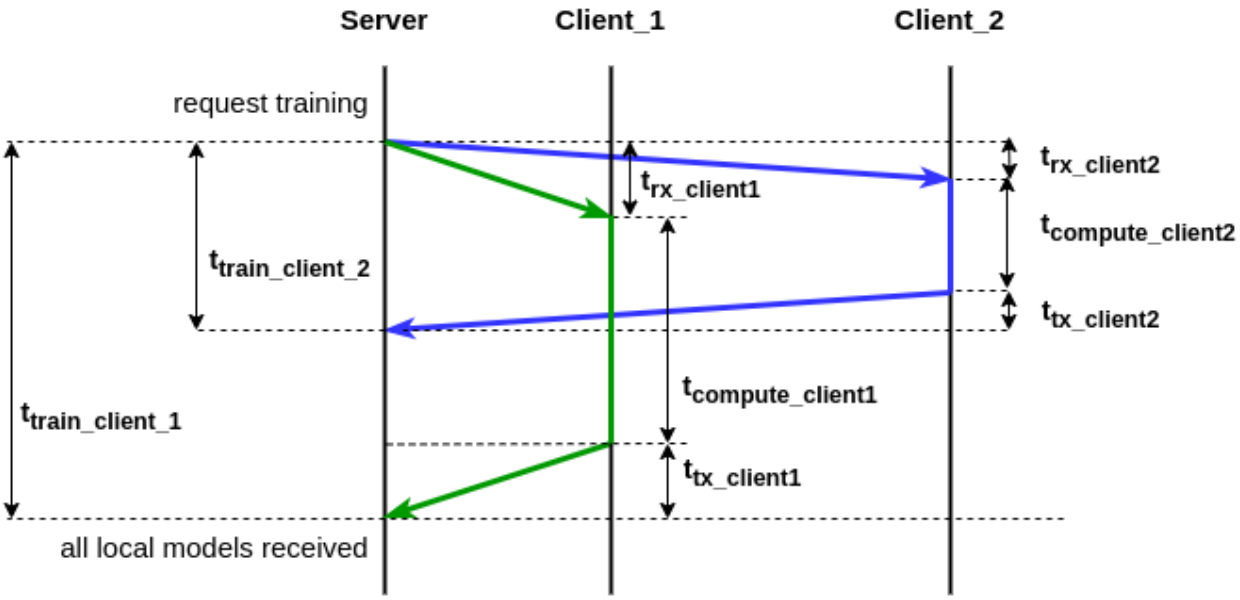

Figure 4.4. Timer details for a server and two clients in a federated learning round.

The decentralized behavior of the client is achieved by adding to the client a component which is able to compute new training values. This computation is done at each round. The client receives the server provided training values and is able to overwrite them. Such capacity allows to take local circumstances and conditions into account. Different policies may be configured at the client to customize its decentralized decisions to specific needs.

With regards to computing capacity, the client for instance may run on a compute node where other applications are executed as well. If other applications have a higher priority or need to maintain a certain service level, then the client may decide to reduce the training effort done, such as reducing the number of training samples in this round.

Bandwidth availability is another factor that the client can be taken into account: in low bandwidth links where another network-intensive application started, the transmission of the machine learning model between client and server will take longer, hence producing that the overall training round to be finalized at the server suffers from this delay. A

Co-funded by the Horizon 2020
Framework Programme of the European Union

decentralized client decision may reduce the number of training images, thus sending the model back to the server earlier, and compensating for the reduced bandwidth availability.

Figure 4.5 illustrates the components and flow of messages within the client. Essentially, a component called *TrainindDecisionSupport* is added to the client. This component intercepts the received training values and calculates new ones. This calculation is done according to policies which the client is configured with, which allow the client to behave as to the specific needs of the local node. The machine learning model training is then performed with the client specific training values.

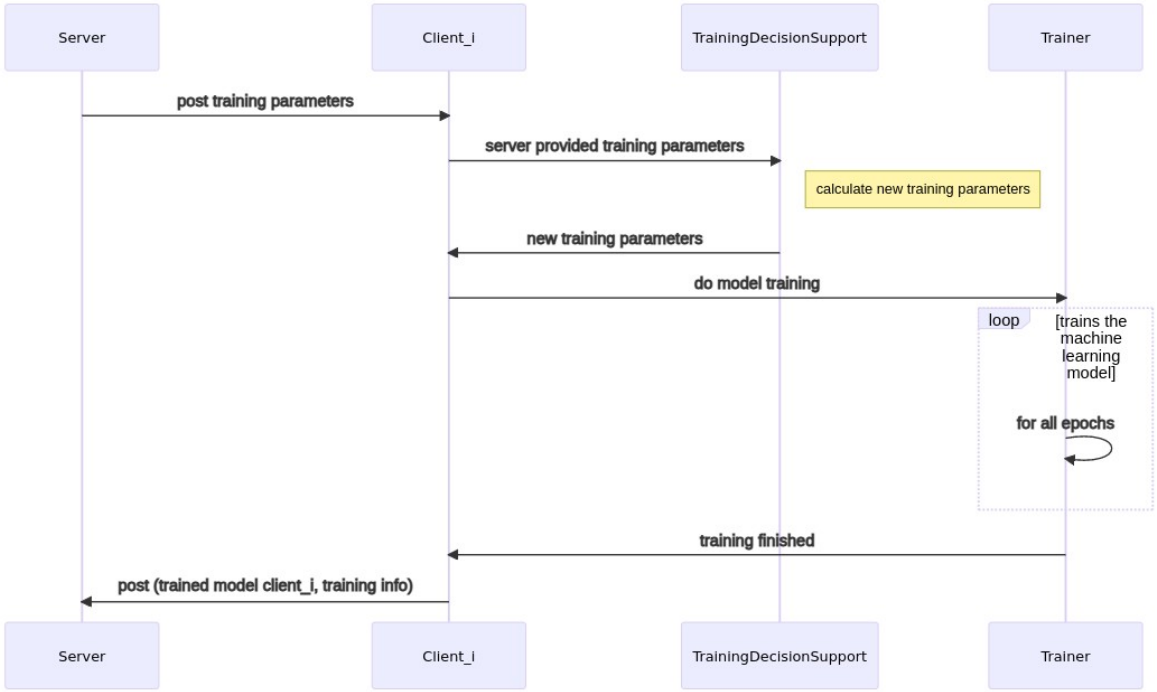

Figure 4.5 Sequence of interactions between components for decentralized FL client.

Figure 4.6 shows the workflow of the federated learning training. Initially, the server waits for the clients to be registered. If no previous information exists, all clients are assigned the same federated learning parameters. After sending the models to the clients, the server starts timers in order to measure the client training time. The clients send back the models along with some performance data. This performance data can contain machine learning data and system data of the client. After receiving all client data, the dynamic configuration module is fed with the metrics and sends back the new values for the training parameters of each client.

Co-funded by the Horizon 2020
Framework Programme of the European Union

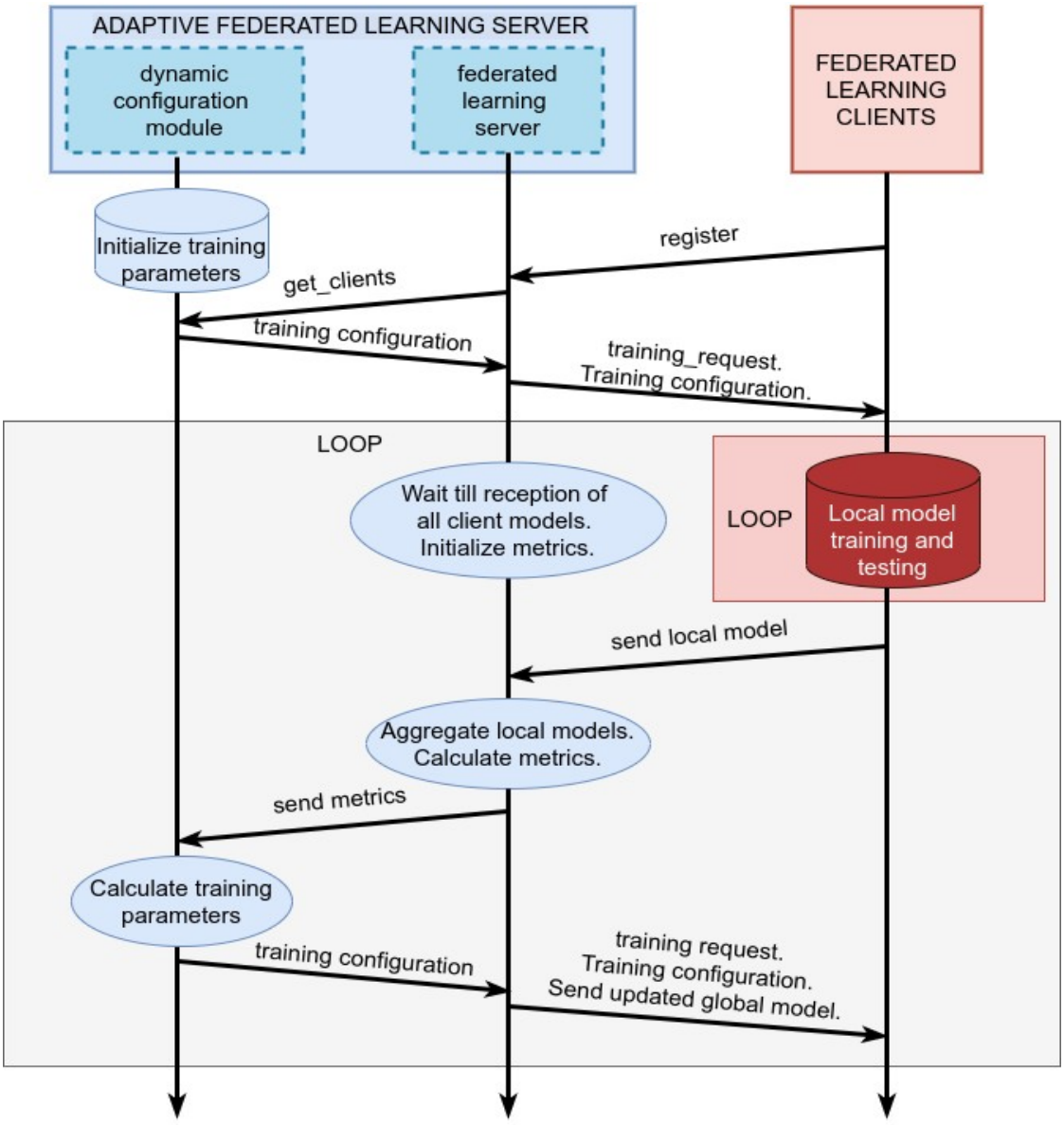

Figure 4.6 Sequence of interactions between server, clients, and dynamic configuration module to do adaptive federated learning.

Experimentation

Analysis of resource consumption pattern

The objective is to study the resource consumption of the devices in the testbed when running federated learning and being interconnected over the wireless mesh network.

For the experiment, the federated learning task to be executed is to train a 6-layer Convolutional Neural Network (CNN) model of around 420,000 parameters with the

Co-funded by the Horizon 2020 Framework Programme of the European Union

Chest_X_ray dataset[1]. The clients are configured to train 1 epoch in each round and the number of images for training and testing are 200 and 100, respectively. Three rounds are trained in both of the following experiments.

Experiment 1: Federated learning clients on different hardware. The objective of this experiment is to observe the behaviour and resource consumption of FL clients w*hen run on different hardware*. For this experiment we run one client in a device of the APU2 cluster e104, and the other client in a device of the Minix cluster e104 (see Figure 4.1). The server is deployed on another device in the PC Engines APU2 cluster e104. Figure 4.7 shows the results. Comparing the times in Figure 4.7d to 4.7f, when the model is exchanged with the server, it can be seen that the client in the Minix device replies quicker to the server with the trained model than the client in the APU2.

Experiment 2: Federated learning clients with different link bandwidth. In this experiment we aim to observe *the effect of different link bandwidth available at the clients*. We chose one device from each of the 5 locations of the testbed. The FL server is installed in one of the APU2 devices from its cluster e104. The FL clients are installed on the Minix Pisuerga device, Minix Bellvitge device, a device from the Minix cluster e208, and one from the Minix cluster e104, in total 4 clients, all on Minix devices.

Figure 4.8 shows the measured resource consumption. With regards to the CPU and memory consumption of the clients (Figures 4.8c, e), the three training rounds done by each client can be clearly observed by the peaks in the CPU consumption. For the training, almost the complete CPU capacity (the four cores) are used. The memory consumption is moderate, as being below 1 GB and taking into account that the devices have 4 GB RAM available. It can be observed that the FL client Minix Pisuerga (Figure 4.8c) started with a higher memory consumption compared to the other client (Figure 4.8e). This is due to the fact that the Minix Pisuerga client participated already previously in a federated learning round with the server, while the other clients joined the federated learning network later. The traffic produced during the federated learning rounds is shown in Figures 4.8 d, f. It reflects the available bandwidth between the locations. For instance, in the low bandwidth link to the Minix Pisuerga client, the traffic produced by sending the ML model between the client and server is lower and it takes longer to transmit the model, while in the faster link of the client in the Minix cluster e208 the traffic due to the model exchange has higher peaks.

[1]Chest X-Ray Images. https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

Co-funded by the Horizon 2020
Framework Programme of the European Union

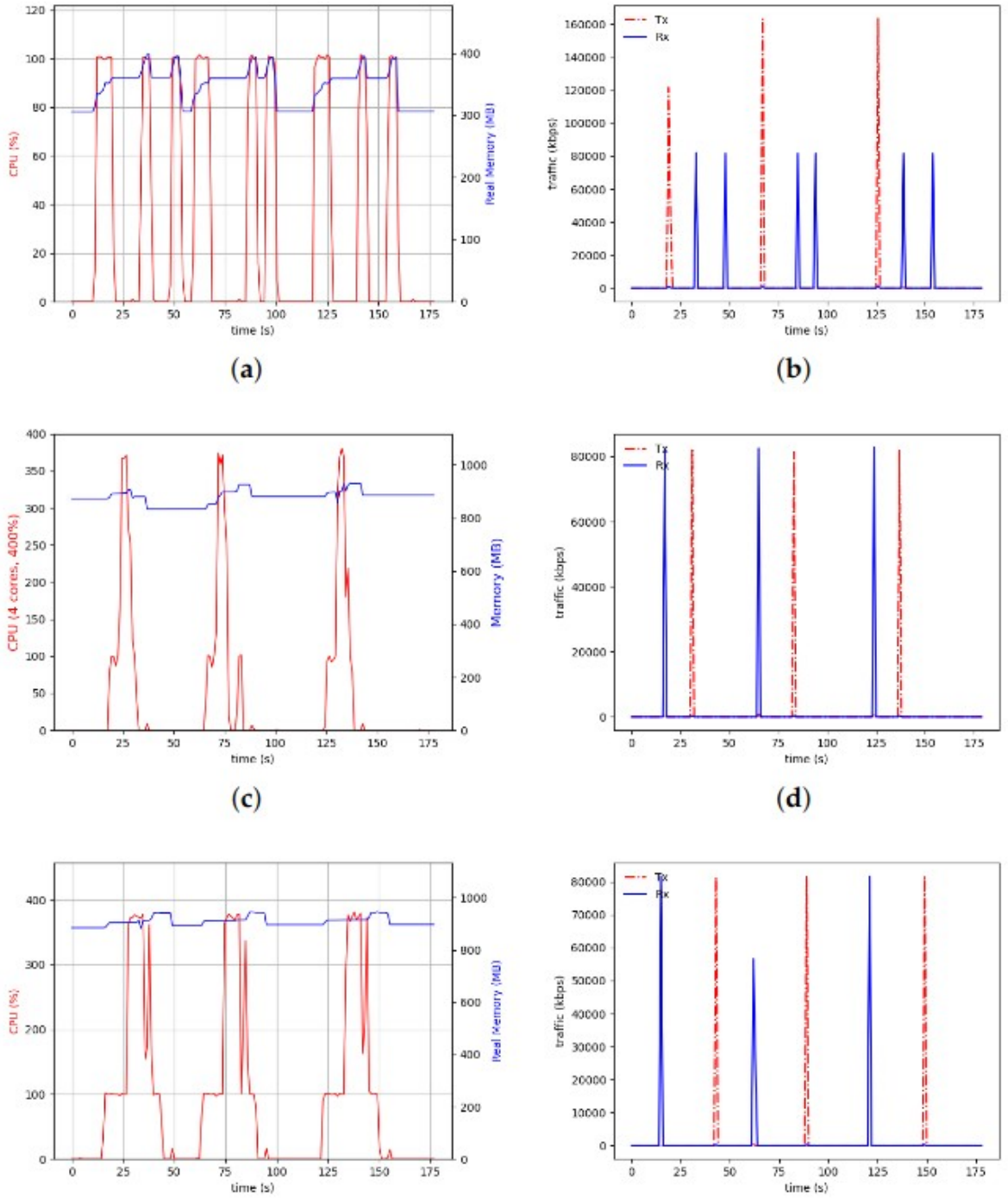

Figure 4.7. Federated learning clients in different hardware. Resource consumption in three training rounds. (a) Server APU2 CPU and memory consumption. (b) Server APU2 bandwidth consumption. (c) Client Minix CPU and memory consumption. (d) Client Minix bandwidth consumption. (e) Client APU2 CPU and memory consumption. (f) Client APU2 bandwidth consumption.

Co-funded by the Horizon 2020
Framework Programme of the European Union

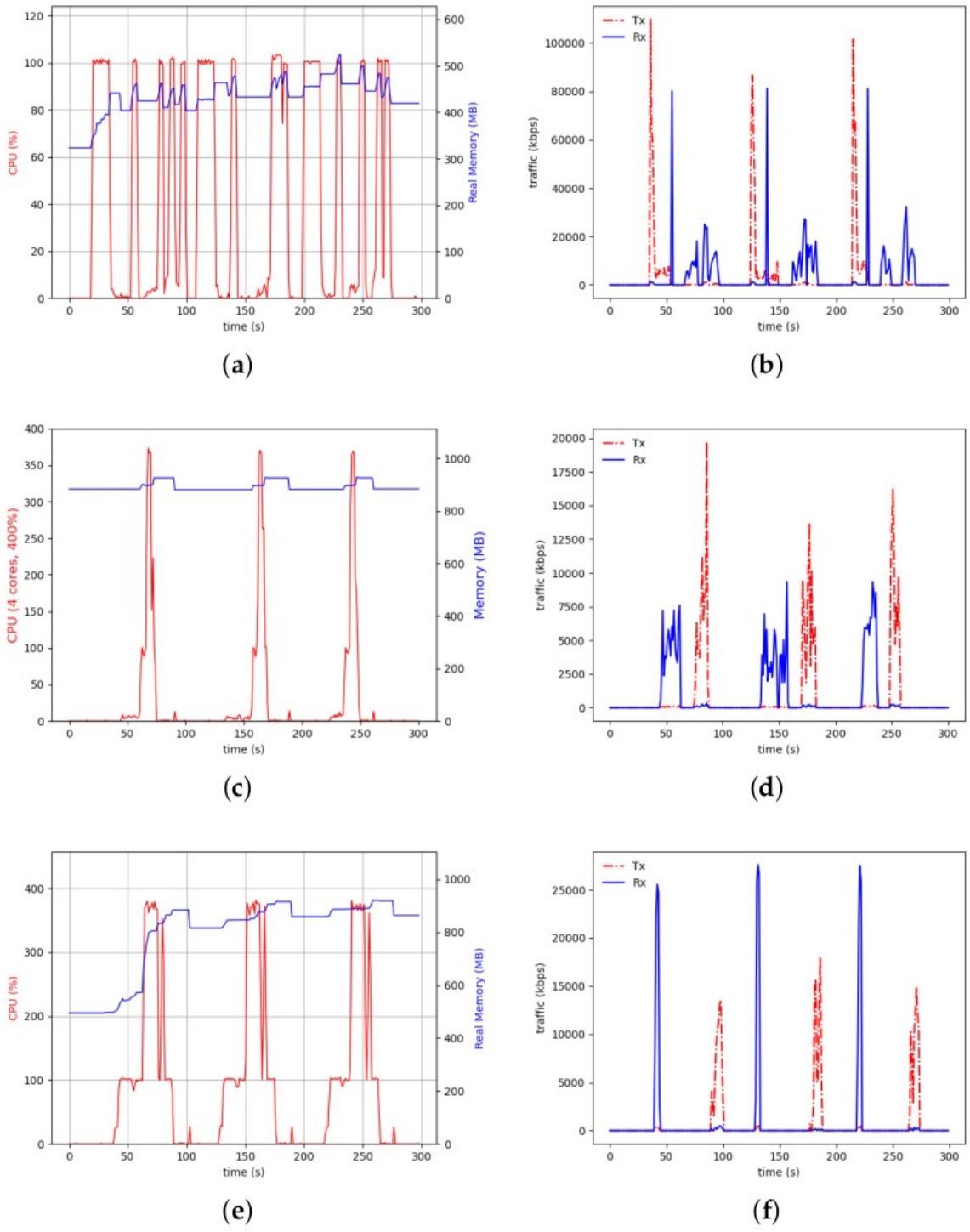

Figure 4.8. Federated learning clients with different link bandwidth. Two of four clients in Minix

devices are shown. Resource consumption in three training rounds. (a) Server APU2 CPU and memory consumption. (b) Server APU2 bandwidth consumption. (c) Low-bandwidth client CPU and memory consumption. (d) Low-bandwidth client bandwidth consumption. (e) High-bandwidth client CPU and memory consumption. (f) High-bandwidth client bandwidth consumption.

Co-funded by the Horizon 2020
Framework Programme of the European Union

Analysis of the hardware affecting the training time

We conduct an experiment to understand better how the different hardware configurations can increase the training training time. For this we run the training of the CNN introduced previously on the APU and Minix devices, first using the four cores of the processor and then limiting the processor usage to one core. Table 4.7 shows the results which are computed to the reference of the training time of the Minix with four cores. First it can be seen that the Minix devices is performing the training faster than the APU2. Secondly, reducing the number of cores used to one core produces an increase of the training time in both devices.

	Minix 4 cores	APU2 4 cores	Minix 1 core	APU2 1 core
Relative training time increase	reference value	+13%	+42%	+75%

Table 4.7 Increase of training time for different hardware.

Server-side adaptive federated learning with heterogeneous clients

In this section server-side adaptive federated learning is experimented. In this experiment we use three types hardware to host three federated learning clients. Specifically, we use a VM with two cores running on a host with a i5 processor, the Minx device and the APU2 device, which in terms of computational capacity result in having a fast, medium and slow client. We do 50 training rounds.

Experiment 1: Baseline. In this experiment we measure different metrics when the clients over the 50 rounds train with 6 training samples each round (Figure 4.9). Slight variations in the fastest client along the 50 rounds are due to the fact that this client was running on a non-dedicated machine.

Co-funded by the Horizon 2020
Framework Programme of the European Union

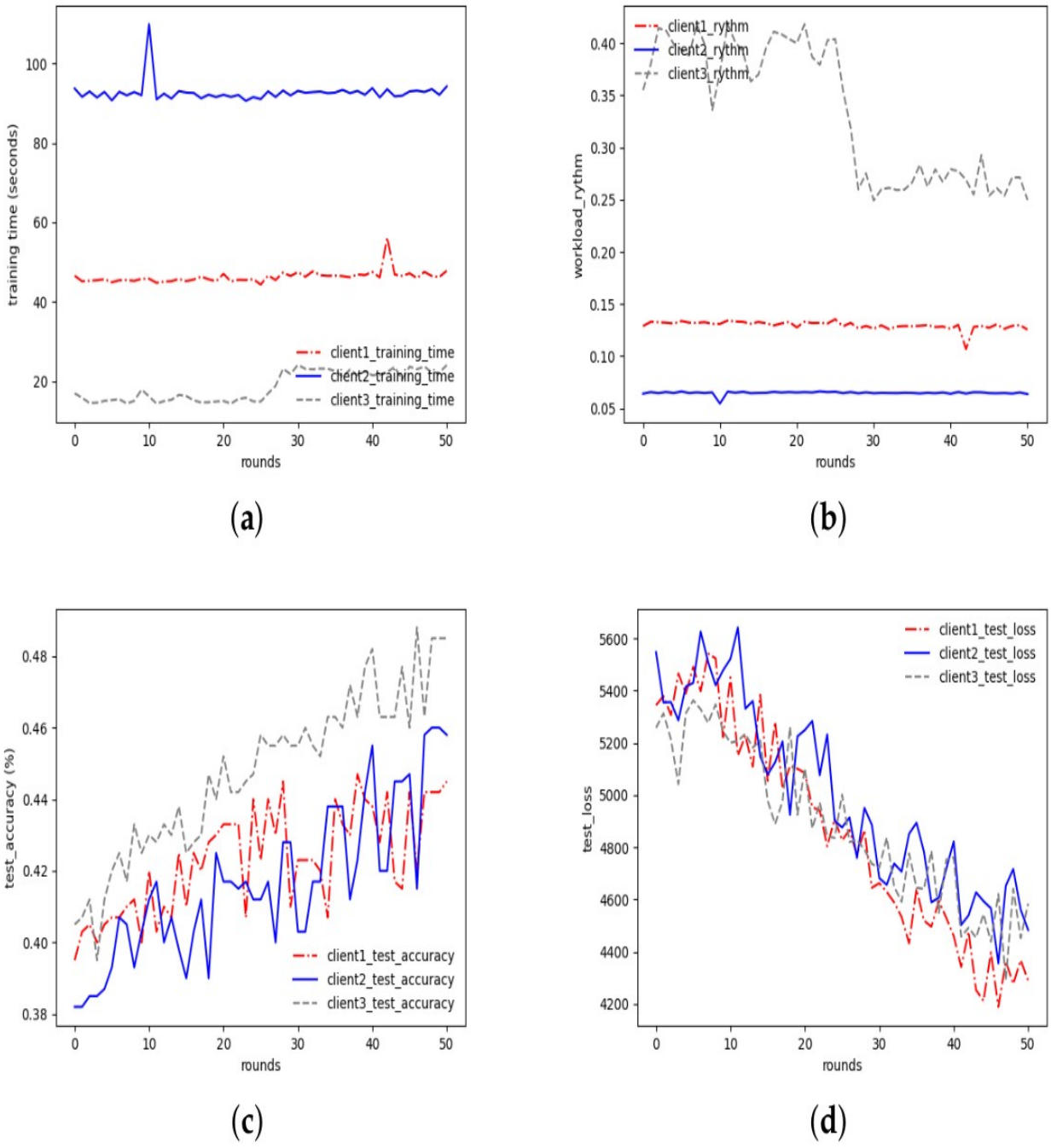

Figure 4.9. Baseline. Behavior of three clients. (a) Training time. (b) Server-measured client rythm. (c) Test accuracy. (d) Test loss.

Experiment 2: Adaptive server. In this experiment during the first 10 rounds the server applies its default behavior (without being adaptive). Then, from round 11 to 50 the server applies the algorithm implemented in the dynamic configuration module to perform adaptive behavior. The experimental setting for training was to have a minimum number of 6 training samples (which can be increased due to adaptive behavior) and inference after training was done with 200 test samples at all clients. Since the number of training samples

Co-funded by the Horizon 2020
Framework Programme of the European Union

used by each client varies with the adaptive behavior, the federated averaging algorithm at the server was extended to apply weighted averaging.

Figure 4.10 shows the obtained results. In Figure 4.10a it can be seen how the training time of the fast clients after 10 rounds increases to that of the slow clients. This is due to the fact that the server increased the number of training samples for the fast clients, as can be seen in Figure 4.10e. Comparing the accuracy achieved of the adaptive federated learning server in Figure 4.10e with the accuracy of the baseline training (Figure 4.9), it can be observed that the first is significantly higher. This is expected since the number of images used for training in the adaptive server configuration is higher. However, since the reference for adjusting the number of training images for the fast clients is the training time of the slowest client, the overall time for training the model in both baseline and adaptive federated learning is similar.

Co-funded by the Horizon 2020
Framework Programme of the European Union

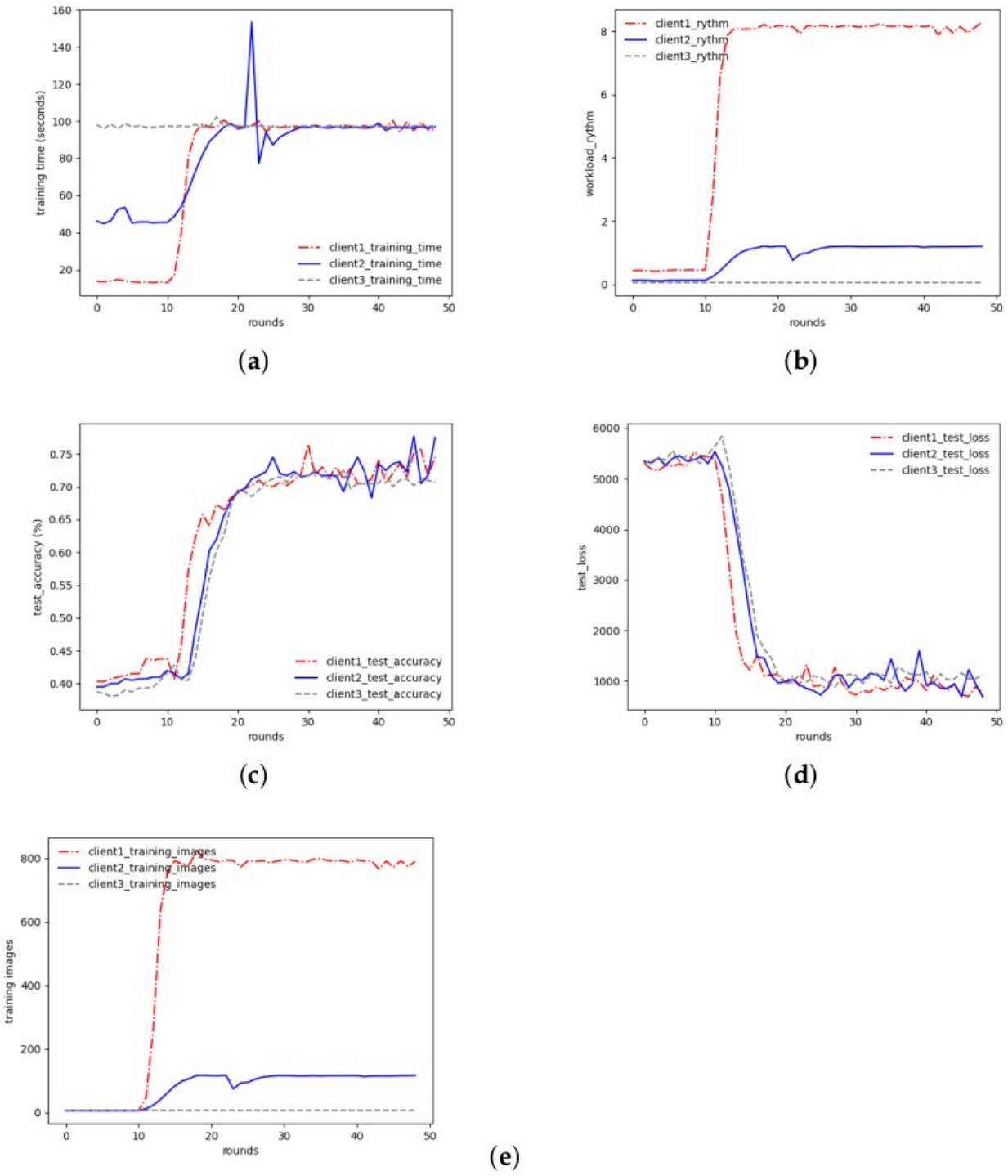

Figure 4.10 Adpative server. Behavior of three clients. (a) Training time. (b) Server-measured client rythm. (c) Test accuracy. (d) Test loss. (e) Number of training images.

Co-funded by the Horizon 2020
Framework Programme of the European Union

Adaptive federated learning with decentralized clients

In the following we report experimental results related to applying the *decision support module* at the client introduced previously. In order to take a decision at the client, the client was configured to be able to change by one sample the number of training samples suggested by the server. For this the client measured if there was a change in the difference of the training time measured at the server and at the client at previous rounds. The sign of this metric can indicate a change in the network conditions, according to which the client then decides to increase or decrease the number of training samples.

Figure 4.11 shows the obtained results. Adaptive behaviour can be observed in the figures 4.11a-d. A detailed look at the numerical values shown in Figure 4.11e shows indeed changes (within the allowed range of +-1) to the number of training samples suggested by the server. To take the decentralized local decision, the client evaluated the evolution of the measured training times at client and server shown in figure 4.11f. Here the training time measured at the server includes the communication overhead.

Co-funded by the Horizon 2020
Framework Programme of the European Union

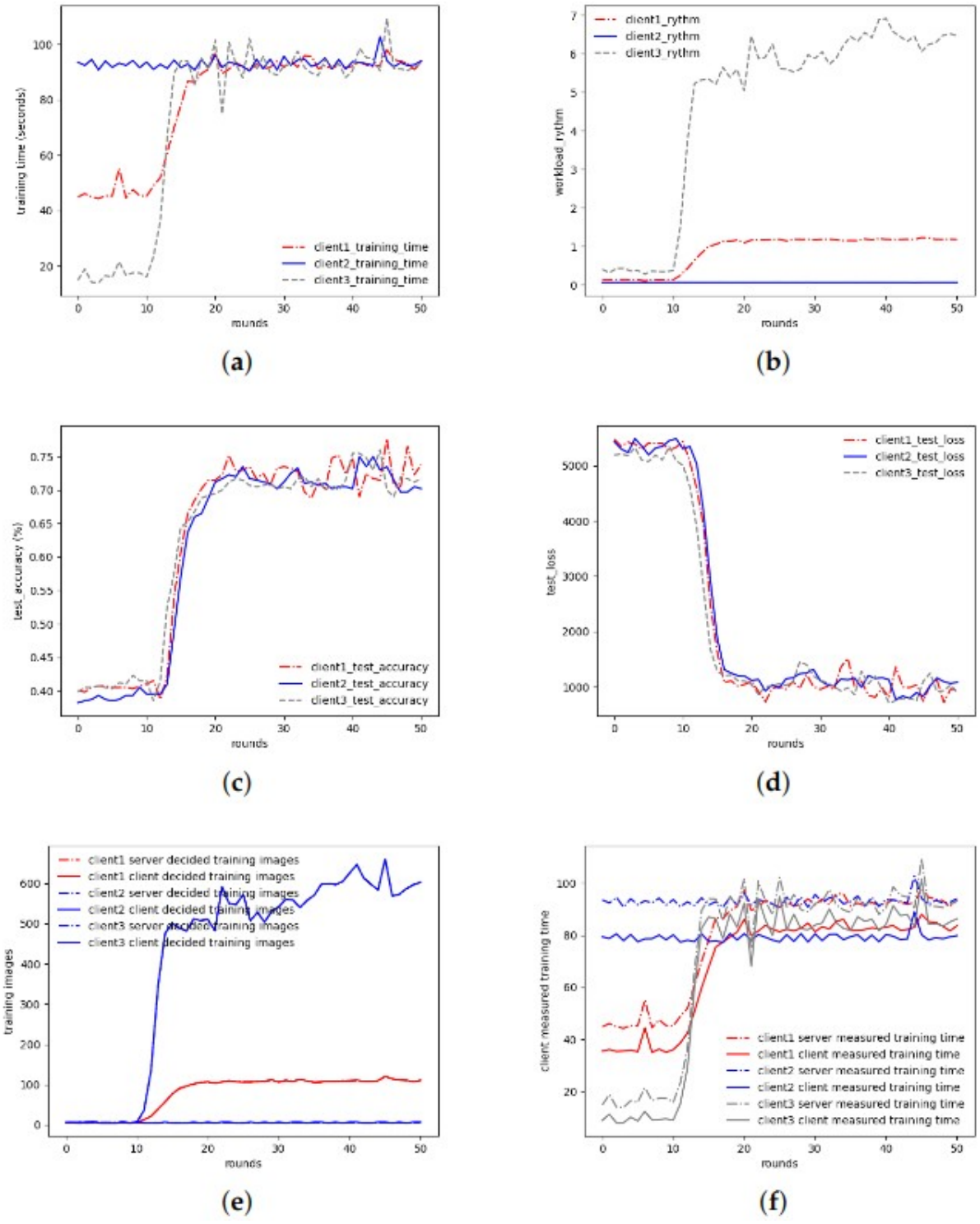

Figure 4.11 Decentralized adaptive FL. Behavior of three clients. (a) Training time. (b) Server-measured client rythm. (c) Test accuracy. (d) Test loss. (e) Number of decided training images. (f) Server vs client measured training time.

Co-funded by the Horizon 2020
Framework Programme of the European Union

4.1 Discussion and Analysis on Results

The results obtained from the diverse experiments provided new insights about the options to design adaptability and decentralization in the federated learning process. As the main result, we could indeed develop and evaluate decentralized adaptive federated learning with 1) a server capable to adapt to the heterogeneity of clients, and 2) clients able to take decentralized decisions according to local criteria on how to perform the model training. We also identified several issues for future work, as well as directions that could be pursued further.

The presented findings support the direction for a new way to conceive the federated learning process, where the *intelligence* moves from the centralized server to the clients, leading to a network of peers. This intelligence at the client can address different criteria of a federated learning application. One option could be to improve the performance of the federated learning process itself, e.g. being the client most collaborative to obtain the highest model accuracy in the fastest way. With such a policy a client could adapt its training time to maximize this goal.

However, also more client-centric policies are possible, where a non-dedicated client could collaborate as to a local criteria in a federated learning process along to doing other tasks as well. Making local data available for training could be evaluated by an intelligent client. The value of making a training contribution could be estimated, which may depend on the available local dataset for the training at the client and if other clients are able to make a similar contribution. Federated learning with intelligent clients could be combined with a marketplace where contributions to a federated learning process could be requested, offered and traded.

Heterogeneity of the computing infrastructure is present in real environments at the network edge, as demonstrated by the testbed characteristics. A default federated learning process designed for homogeneous conditions will possibly work, but will not run in an optimal way. With federated learning moving to the edge, there are several reasons for addressing this heterogeneity: 1) Improving the resource usage efficiency of federated learning, since it is a computing service which has energy consumption and needs to be optimized 2) Allowing the individual nodes to offer different data sharing policies, which could increase training data availability while respecting the privacy concerns for certain data. 3) Being able to form federated learning networks which integrate nodes of different capabilities. This may allow to benefit from slower nodes which have valuable data.

A mixed scenario as experimented previously where the *intelligence* resides both in the server and in the clients has the potential to best adjust to diverse conditions and is suggested to be investigated further. The decisions taken at the server and the clients can principally be even conflicting since they are based on different local knowledge. The server

Co-funded by the Horizon 2020
Framework Programme of the European Union

can compute the most suitable workload for a client based on the server's perspective. The resulting suggested workload, however, may not fit to the real conditions at that client, of which the server is not aware of. The scenario where both server and clients aim to optimize according to their local knowledge may thus lead to situations where decisions are conflicting with each other. The sum of intelligent behaviors could lead to a worse outcome than leaving the decision either to the clients or to the server. Further investigations are needed for which simulations may be more suitable than real world experimentation.

5 Present and Foreseen TRL

The experiments validated the designs in a relevant environment (i.e., an operational wireless mesh network with real nodes). Such an environment can classify as TRL 5.

6 Exploitation, Dissemination and Communication Status

We have achieved to publish three papers within the project, a poster paper presented at IoT 2021[2], a demo paper presented at CCNC 2022[3], and a performance evaluation paper presented at ICITS 2022[4]. In the IoT 2021 paper more details regarding the testbed implementation and preliminary experiments are explained. The CCNC demo paper, for which also a video presentation is available[5], explains in detail how the practical execution of experiments works and the obtention of results. In the ICITS 2022 paper we present several experiments with federated learning conducted in the testbed. We made a post in the Cloudy Web site[6], where the Cloudy platform is related to some other services hosted in the testbed nodes within Guifi.net and to which federated learning could potentially be added.

[2] F. Freitag, P. Vilchez, Ch. Liu, L. Wei, M. Selimi. Testbed in Wireless City Mesh Network with Application to Federated Learning Experiments. ACM/SIGCHI International Conference on the Internet of Things (IoT 2021), Nov 2021, St.Gallen, Switzerland. https://my.ece.msstate.edu/faculty/chliu/papers/conference/PosterFL_IoT2022.pdf

[3] F. Freitag, P. Vilchez, L. Wei, C.H Liu, M. Selimi, I. Koutsopoulos. Demo: An Experimental Environment Based On Mini-PCs For Federated Learning Research. 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC). https://personals.ac.upc.edu/felix/CR_2022_CCNC_Demo.pdf

[4] F. Freitag, P. Vilchez, Ch. Liu, L. Wei, M. Selimi. Performance Evaluation of Federated Learning over Wireless Mesh Networks with Low-Capacity Devices. International Conference on Information Technology & Systems (ICITS 2022), Feb 2022. https://personals.ac.upc.edu/felix/CR_2022_02_ICITS.pdf

[5] Demo presentation. https://drive.google.com/file/d/1cxJZen1buB076HH0RBL4rD-aOs2sY5YI/view?usp=sharing

[6] http://cloudy.community/federated-learning-experimentation-on-cloudy-nodes/

Co-funded by the Horizon 2020 Framework Programme of the European Union

The code we use for the experimentation has been made publicly available and can be obtained from our gitlab repository[7].

7 Impacts

Impact 1: Enhanced EU – US cooperation in Next Generation Internet, including policy cooperation.

Fleshnet has established a new successful collaboration of EU-US partners from the project team coming from three universities (TTU (US). MSU (US), UPC (EU)). With our progress and collaboration we could also raise the attention of two other European research groups (from AUEB and SEEU university), which triggered a collaboration on this topic. This collaboration resulted in a joint paper, which otherwise may not have happened.

Impact 2: Reinforced collaboration and increased synergies between the Next Generation Internet and the Tomorrow's Internet programmes.

The program of the U.S. partners is CNS (Computer and Network Systems) under the CISE (Computer and Information Science and Engineering) program. The impact the project of this NSF one includes: 1) verify the practical usefulness of the research findings on wireless sensor deployment of the U.S. partners; 2) verify whether the proposed repulsive processes are able to accurately model the performance of practical wireless networks; 3) assess the robustness and sensitivity of the proposed models for a given error tolerance. Synergies: 1) the research results of the U.S. partner will lead to key insights into the design and deployment of real-world ultra-dense networks for practitioners; 2) to evaluate the project findings, the EU-US collaboration provided the opportunity to work with Dr. Freitag's team at UPC for experimental validation using FL based implementations; 3) the collaboration delivered building blocks for adaptive decentralized networks experimentally validated in the realistic conditions in the wireless mesh network testbed.

Impact 3: Developing interoperable solutions and joint demonstrators, contributions to standards.

The code for the experimentation has been made available in public gitlab repository and is thus shared for allowing others to build on our work. A demonstration paper about the testbed and experimentation was written by the project team. It was presented at CCNC 2022 hold as a virtual event.

Impact 4: An EU - US ecosystem of top researchers, hi-tech start-ups / SMEs and Internet-related communities collaborating on the evolution of the Internet

[7] https://gitlab.com/dsg-upc/federated_learning

Co-funded by the Horizon 2020
Framework Programme of the European Union

We took advantage of scientific conferences which took place within the project duration and achieved three publications. The conferences were related to scientific communities which research on different aspects of the evolution of the Internet. Specifically, we published and presented the work done in the Fleshnet project in:

1. ACM/SIGCHI International Conference on the Internet of Things (IoT 2021)
2. IEEE 19th Annual Consumer Communications & Networking Conference (CCNC 2022)
3. International Conference on Information Technology & Systems (ICITS 2022)

8 Conclusion and Future Work

The practical experimentation of federated learning in a testbed deployed in a wireless city mesh network revealed resource usage patterns which gave evidence for the heterogeneity that exists in real edge computing infrastructure. It was shown how computing capacity of the nodes and bandwidth availability affect the federated learning process.

A more adaptive and decentralized design was experimented for the server and clients. Building blocks were designed which provided server-side adaptability and client-side decentralized decisions. The workflow for the interactions between clients and server was developed. The communication interfaces of the client and server were designed.

Applying the proposed design allows federated learning components to take into account different criteria for adapting or taking local decisions, such as the network situation, the computing capability, data privacy and the data sharing policy of each node.

The obtained results are not limited to the specific infrastructure and experimental environment. As machine learning moves into ever smaller computing devices with energy and computing limitations, applying the gained insights about adaptability and decentralization may be critical for being able to deliver the maximum outcome under important resource constraints. In this respect, the the Fleshnet partners have decided to exploit the momentum of the joint work and extend their collaboration with a new project proposal targeting the connected dynamic intelligence at the tiny edge.

9 References

1. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated Machine Learning: Concept and Applications. ACM Trans. Intell. Syst. Technol. 2019, 10. doi:10.1145/3298981.

2. Wang, S.; Tuor, T.; Salonidis, T.; Leung, K.K.; Makaya, C.; He, T.; Chan, K. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. IEEE Journal on Selected Areas in Communications 2019, 37, 1205–1221. doi:10.1109/JSAC.2019.2904348.

Co-funded by the Horizon 2020 Framework Programme of the European Union

3. Xu, H.; Li, J.; Xiong, H.; Lu, H. FedMax: Enabling a Highly-Efficient Federated Learning Framework. 2020 IEEE 13th International Conference on Cloud Computing (CLOUD), 2020, pp. 426–434. doi:10.1109/CLOUD49709.2020.00064.
4. Zhang, T.; He, C.; Ma, T.; Ma, M.; Avestimehr, S. Federated Learning for Internet of Things: A Federated Learning Framework for On-device Anomaly Data Detection, 2021, [arXiv:cs.LG/2106.07976].
5. Gao, Y.; Kim, M.; Abuadbba, S.; Kim, Y.; Thapa, C.; Kim, K.; Camtep, S.A.; Kim, H.; Nepal, S. End-to-End Evaluation of Federated Learning and Split Learning for Internet of Things. 2020 International Symposium on Reliable Distributed Systems (SRDS), 2020, pp. 91–100. doi:10.1109/SRDS51746.2020.00017.
6. Baig, R.; Roca, R.; Freitag, F.; Navarro, L. Guifi.Net, a Crowdsourced Network Infrastructure Held in Common. Comput. Netw. 2015, 90, 150–165. doi:10.1016/j.comnet.2015.07.009.
7. Baig, R.; Freitag, F.; Navarro, L. Cloudy in guifi.net: Establishing and sustaining a community cloud as open commons. Future Generation Computer Systems 2018, 87, 868–887. doi:https://doi.org/10.1016/j.future.2017.12.017.
8. Parareda, E.Y. Federated learning network: Training distributed machine learning models with the federated learning paradigm 2021.

10 Glossary

FL	Federated Learning
MSU	Mississippi State University
NGI	Next Generation Internet
TTU	Texas Tech University
UPC	Universitat Politècnica de Catalunya
WIT	Waterford Institute of Technology (Coordinating Partner)

Co-funded by the Horizon 2020 Framework Programme of the European Union

Deliverable 3: Part II

Financial and cost information

This part is to be treated as a consortium confidential deliverable, and access is restricted to consortium partners and EU commission operatives.

Co-funded by the Horizon 2020
Framework Programme of the European Union