ACCEPTED FROM OPEN CALL

The Role of SDN to Improve Client Selection in Federated Learning

Ahmad Mahmod, Pasquale Pace, and Antonio Iera

The authors introduce an approach to client selection that is augmented and made more effective by a joint orchestration carried out by the federated learning server and the controller of a software-defined networking network.

Abstract

In an ever-increasing number of contexts, it has now become common to use federated learning (FL) techniques, through which several heterogeneous devices cooperate in a distributed manner to increase the effectiveness in training machine learning (ML) models while maintaining confidentiality of the respective data. The federated learning process shows performance levels that are highly dependent, not only on the data available to each participant in the process, but also on the choice of devices that act as clients from time to time and that collaborate with each other. So far, much of the literature has focused on a client selection that considers the device computational/memory capabilities and the end-to-end delays of the process. However, no one so far has fully exploited the intrinsic capabilities of current and future programmable networks. This article introduces, for the first time, an approach to client selection that is augmented and made more effective by a joint orchestration carried out by the FL server and the controller of an software-defined networking (SDN) network. It will be demonstrated through a performance evaluation campaign that the use of typical SDN principles also in the client selection phase leads to significant advantages in terms of effectiveness and efficiency.

Introduction

Recently, federated learning (FL) is attracting growing interest, being an efficient paradigm capable of achieving similar results to centralized learning but through a fully distributed process [1]. According to FL, in each round the learning model is trained with local data available at remote devices participating in the process (the so-called FL Clients) and only the resulting model parameters are transmitted to a central server (the so-called FL Server). This latter aggregates them to obtain a more performing model and returns the newly updated parameters to all clients for the next training round. The procedure is repeated for several cycles until the desired level of accuracy is achieved. In this way, data transmission costs are significantly reduced and user privacy is protected [2].

The shorter the iterations, the better performance the FL process can achieve [3]. Considering that the FL server cannot aggregate the global model until all clients have successfully delivered their parameters, there are two main causes of reduced effectiveness of the entire process (for the same available dataset). The first is the involvement of FL client devices with low processing capacity and memory. The second main reason for process inefficiency is the presence of high-performance devices which however have poor quality connections or a congested path to the server. In both cases the central server will have to wait for stragglers (i.e., devices that delay FL training), which could significantly reduce the convergence time to the desired model accuracy value. Therefore, carrying out effective client selection that avoids the participation of stragglers in the FL process becomes crucial to speed up the training process.

The literature is rich in proposals addressing the issue of *Client Selection* in a very sophisticated and effective way. Most of them come from the data science and ML research communities and approach Client Selection by primarily focusing on the device computational resources and on the nature of the local data used for training. Equally interesting works, reflecting the contribution of the wireless communications research community, address the problem also considering the availability of communication resources at the air interface. Instead, solutions addressing the possible impact that the delay on Client-Server network segments could have on client selection, which the networking scientific community contributes to, do not appear to have been adequately studied yet.

At the same time, new available paradigms in software-based networks such as emerging 5G and 6G ones, first and foremost that of Software-Defined Networking (SDN), have the potential to provide strong support to federated learning processes [4]. It is not just about creating a sort of overlay network that constantly optimizes the paths followed by client-server traffic and vice versa, which is part of the standard purposes of SDN. Rather, an interesting and still missing contribution, as far as we know, consists in evaluating the potential advantage of leveraging SDN within the choice and continuous dynamic updating of FL clients. This is with a view to maintaining the distributed learning process at constantly high levels of effectiveness, efficiency and quality.

In a previous article [5] we began to investigate the performance achievable from SDN-assisted client selection in the presence of homogeneous clients. This allowed us to evaluate the positive effects of introducing SDN into the FL selection

Digital Object Identifier: 10.1109/MCOM.003.2400093

Ahmad Mahmod is with ICube Laboratory University of Strasbourg, France;
Pasquale Pace and Antonio Iera are with University of Calabria, Italy, and also with CNIT, Italy.

0163-6804/24/$25.00 © 2024 IEEE

process in the presence of clients that all have the same computational performance. In this way, since differences in computational power and client memory did not come into play, the benefit of the reduction in network delays on the performance of the FL process clearly emerged.

In this article, we go further and improve the performance of the entire FL process, considering the following aspects not yet the subject of any work in the literature, to the best of the authors' knowledge:

- When carrying out client selection round after round, consider *in a joint manner* both the available computational resources (Memory and CPU) of each client and the conditions of the network links that connect each of them to the server.
- Implement mechanisms whereby SDN enters the selection process by providing the server with additional information from all potential clients (both the one already engaged in each round and the others not currently engaged but available) over their optimized paths.
- Exploit this information, by the FL server/orchestrator, together with that relating to the computational resources to decide whether to confirm the clients in the subsequent round or replace them with other clients estimated to be better performing also considering the network crossing delay.
- Assess SDN's ability to improve the convergence times of the FL process and quantify the computation and traffic burden added to the system and controller.

Specifically, the work is organized as follows. Following the brief overview of reference works in the next section. Then we describe the reference system for our study. Following that we introduce the proposed client selection policy. A proof-of-concept test campaign with the results obtained follows next. Finally, we highlight possible open research issues and report our conclusions.

Related Works

For some years now, several works have appeared in the literature that propose more efficient *Client Selection* methods than those simply based on random or quasi-random criteria. Some contributions, like the one in [6], propose a multicriteria-based approaches to FL client selection by accounting for CPU, memory, energy, and time. Other contributions also add as a criterion of choice the guarantee of fairness to the clients [7] and the reputation of the clients themselves [8].

Another interesting work proposes to increase the training efficiency, as well as the final model performance, through a stochastic client selection scheme [9]. The authors of [10], instead divide clients into tiers based on their training performance and select clients from the same tier in each training round to mitigate the straggler problem caused by heterogeneity in resource and data quantity.

The examples shown are representative of a large body of research that mainly addresses the problem of FL client selection by focusing on heterogeneity of local data and terminal computing resources. Some also consider efficiency aspects in communication but mainly limiting themselves to the wireless segment that connects IoT devices to the network. Our approach instead aims precisely to evaluate the influence that any bottlenecks in the fixed segment of a 5G/6G softwarized network can have on the process and how to make the latter more efficient by giving a role to the SDN controller within the client selection process.

Research on the role of *SDN in supporting distributed learning* is at the initial embryonic stage. For example, in [11] the authors propose the use of an SDN controller to create an overlay network in which the FL server and clients can perform auction bidding and product provision. Differently, in [12] an efficient resource slicing scheme for optimizing federated learning in software-defined IoT is proposed. In [13] it is suggested that SDN could be used to make IoT gateways alleviate some of the distributed ML issues by collecting data from local IoT devices and cooperate with nearby edge servers. In a scenario with mobile IoT devices involved in FL processes, the authors of [14] propose a clever content placement supported by SDN controllers capable, in the presence of multiple points of computation and caching, to control the data plane and enable seamless communications while maintaining QoS.

Giving a role to the SDN controller within the client selection process, as far as we know, is a concept not addressed so far in the literature. And this is precisely the main objective of our research, in line with the emerging interest in the "Network for AI" paradigm, as opposed to the more traditional approach of "AI for networks".

The Reference System

The schematic reference system architecture underlying our proposal is shown in Fig. 1. The proposed selection process supported by SDN is implemented by what we indicate in the figure as the orchestration module. The latter includes the functions traditionally implemented by the FL server (it could also coincide with its augmented version), namely the training of the local models received by the heterogeneous devices and the creation of the global model to be redistributed to them. It also implements additional functions that enable the interaction with a Network Orchestrator (SDN Controller) and the selection of clients with the support of the latter, as illustrated in the next section.

The data exchange between FL Orchestrator and Network Controller is continuous. In one direction the Controller sends information on the delays relating to the paths on which the data of the active clients travel, as well as estimates on the delays of the remaining clients not currently involved in the process. The information relating to the outcome of the client selection travels in the opposite direction, allowing the Network Controller to dynamically select and maintain the best paths for the selected clients and to estimate the delays of clients that are temporarily inactive but potentially selectable later.

SDN-Assisted Client Selection

Several sophisticated client selection schemes could be considered to jointly exploit information received from the devices (relating to their capabilities) and from the SDN network controller

Several sophisticated client selection schemes could be considered to jointly exploit information received from the devices (relating to their capabilities) and from the SDN network controller (relating to the conditions of the network and the selected paths).

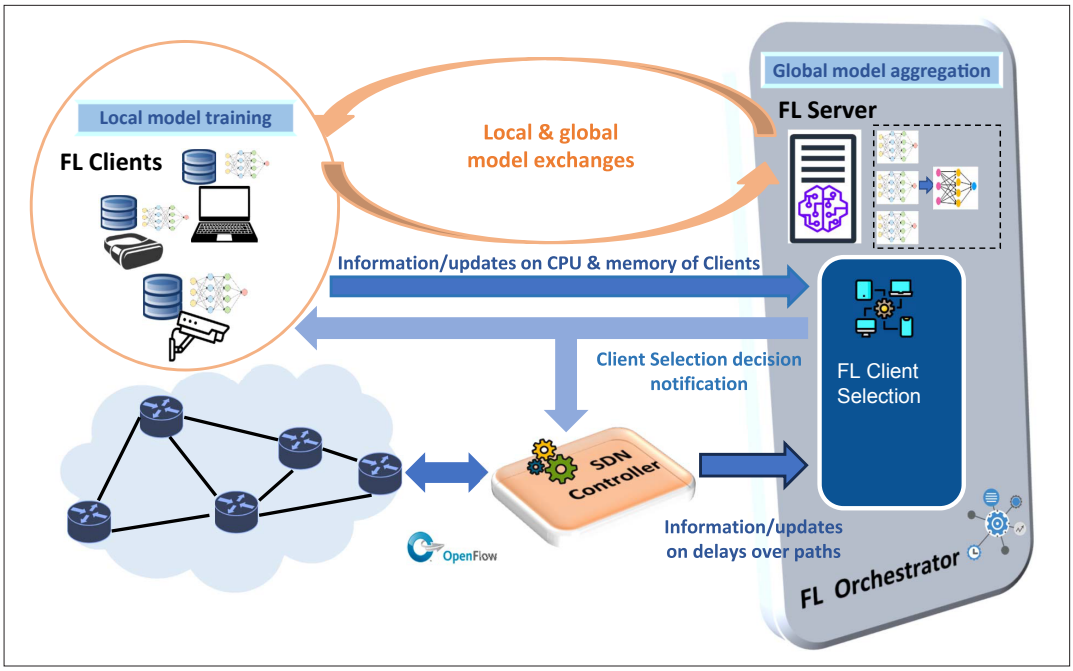


FIGURE 1. The proposed SDN-assisted FL system.

(relating to the conditions of the network and the selected paths). This is not within the scope of this work, since what we intend to provide is a proof of concept that the proposed approach can show interesting advantages in the face of limited additional costs. Therefore, for now we will use a very elementary scheme that relates the two contributions mentioned, reserving the right to delve deeper into this specific aspect in future works.

We simply rely on a linear combination of two indices. A first index is the Computational Index (CI), calculated for each client as a weighted average of the resources that the device can make available for the learning process. It takes into account the contributions of CPU and free memory FM which, without any loss of generality, have the same weight in this proof-of-concept study; future investigations will be dedicated to discovering the effects of a dynamic choice of these parameters too. This index can give a rough estimate of the goodness of a device in terms of precision and computational speed. A device with a high-performance and less busy CPU is potentially more capable of performing a greater number of operations while taking less time to perform parameter updates; it can store a larger training data set if its memory is larger; if the charge level of its battery is high the performance curve can reach the maximum to obtain greater precision. The higher the index value, the better the device's ability to perform fast and accurate computations.

As shown in Eq. 1, the processing capacity of the generic client j is computed by considering the ratio between the CPUs available in the client compared to the maximum number of CPUs available in the best performing client and their percentage of utilization.

$$CPU_j = \frac{\#\ of\ CPUcores_j}{MAX\ CPU\ cures} * \left(1 - CPUusage_j\right); \quad 0 \le CPUusage_j \le 1 \tag{1}$$

A similar calculation is made for the second contribution to the CI index, which is the available free memory (FM) in each client normalized to the maximum amount of RAM available in the most powerful client.

Furthermore, for each device a Normalized Delay Index (NDI) will also be used to account for the expected delay to deliver the update to the Server, based both on the value of the throughput at the device interface and on the network resources available along the path (of which the SDN Controller has an always updated view through the information exchanged with the Openflow switches). *NDI* is expressed as the normalized communication delay of a given client with respect to the maximum delay experienced by the clients during the previous training round.

By appropriately combining the two indexes, an overall performance metric *PI* can be obtained, which uniquely indicates for each device *j* how efficient it can work in terms of computational capacity and (predicted) network conditions. This overall metric can drive the choice of the best subset of devices to be selected at each stage of the FL process. A coefficient $\alpha \in \{[0, 1]\}$ that weights the two parts of the index PI can be chosen, from time to time, to give greater weight either to the computational index (CI), if the performance of each device is to prevail, or to the normalized delay index (NDI), thus involving devices which experience a lower communication delay.

$$PI_j = \alpha * CI_j + (1 - \alpha) * (1 - NDI_j) \tag{2}$$

Equation 2 shows how to compute the performance index of the generic client j at the end of each training round.

The proposed SDN-assisted Delay/Computational-resources client Selection Strategy (DCSS) works as follows:

- The selection process begins with all N interested clients that send the orchestrator a request to participate in an FL session,

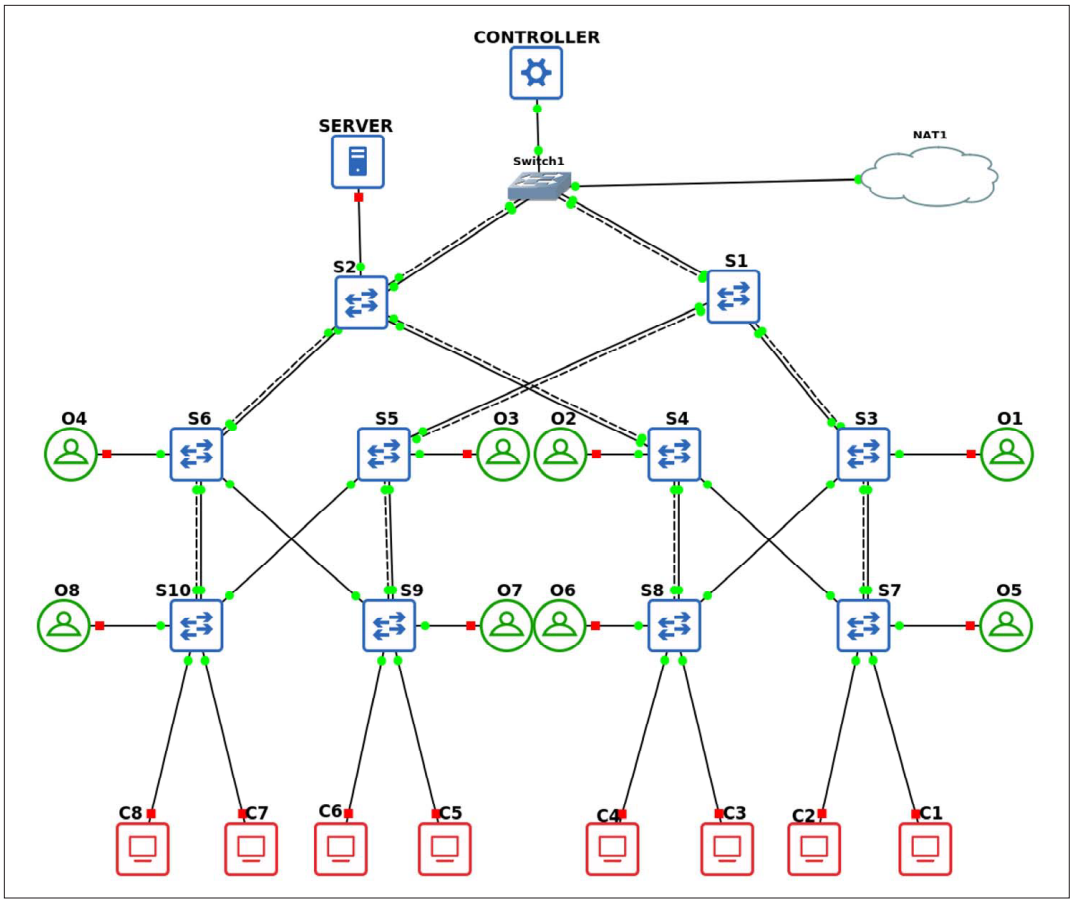


FIGURE 2. Proof-of-Concept network topology.

accompanied by information on the available computational resources (Free Memory and CPU).

- The list of interested clients is also forwarded by the orchestrator to the SDN controller, which proceeds to estimate the instantaneous delays of each client and sends them back to the orchestrator.
- The orchestrator is now capable of computing the PI for each client and selecting the $M \subset N$ clients with the highest performance index to run the first round of the training process.
- Once the FL process starts, the SDN controller proceeds to perform a double measurement in each FL round. On the one hand, it continuously observes the selected clients and collects their delay parameters on the client-server paths. On the other hand, it estimates the path delay between each of the (N-M) clients that do not take part in training in the current round, using a background ping mechanism.
- Whenever an FL round is completed, the orchestrator receives information from clients regarding their current availability of computing resources. It also receives from the SDN controller information on the average value of the delays on the client-server paths of the M clients involved in the previous FL round and on the estimated delay for the (N-M) that did not take part in it. It uses this updated information to make a new selection on the entire set of N devices using the previously illustrated criterion.
- The orchestrator sends back to the SDN controller the update on which clients have been selected for the new round to allow it to proceed with its new estimates and measurements.

Performance Evaluation

This section describes a study aimed at providing a proof-of-concept of the proposed SDN-assisted client selection technique.

The emulation environment used for the study is GNS3 (Graphical Network Simulator version 3) (freely downloadable at https://www.gns3.com/).

In this environment, a test network was used with a multi-level topology shown in Fig. 2 which includes Open vSwitches (OVSs), labeled with S_j, connected to each other by links with a nominal capacity of 100Mb/s, and controlled by an ODL (OpenDaylight) SDN controller. On top of the controller an application is implemented to perform *Dynamic Routing* and, in addition, implement the client monitoring and data exchange functions with the FL controller described in the previous section. Note that the topology is designed to allow multiple choices of routes from each client to the server.

As for the hosts, note that eight of them play the role of clients (labeled C1...C8); while others (labeled O1...O8) are implemented as specific virtual machines in VirtualBox and exported into GNS3 to generate variable background traffic used to load links. Readers find the characteristics of all the devices used for our proof-of-concept

Network Devices	CPU cores	RAM (GB)
CONTROLLER	8	16
SERVER	8	8
Virtual Switches (S1...S10)	4	4
Other virtual devices (O1...O8)	2	4
Heterogeneous Clients (C1...C8) Cat 1 – Low Cat 2 – Medium Cat 3 – High	 2 4 6	 2 4 6

TABLE 1. Network devices specifications.

described in Table 1.

Simulation Assumptions

Without loss of generality, the results shown in this article focus on an FL process based on the *FedAvg* algorithm [15] is implemented via the well-known *Flower* FL framework (available at https://flower.dev/), consisting of 40 evolution rounds, each consisting of 2 epochs. The trained neural network is one of the most widely used in the literature, namely *DenseNet121* neural network (33 MB in size with 8.1 million hyper-parameters), using a well-known dataset, CIFAR-10. The reference architecture is implemented on an HP Enterprise Proliant DL560 Gen10 hardware platform, equipped with 2x Intel Xeon-Gold 6225N processors (2.3GHz and 24 cores) and 256GB of RAM.

In the selection phase, N clients join the federated learning process and the client selection strategy will select the subset of $M \subset N$ best performing clients; furthermore, in each round the client selection strategy is executed again to replace any less performing clients.

The core network is loaded by additional background data traffic so that clients experience different delays in data traffic. To flexibly test all proposed features, we use traffic profile configurations that dynamically overload client paths by suitably configuring network devices (O1...O8) using the *iperf* tool to transmit data traffic between them via UDP connections. This allows us to test the reaction and adaptation of the SDN-assisted client selection scheme to changes in traffic and related overloaded network links.

In particular we have defined an overload traffic profile, in which the two available links of each virtual switch in the third level of the tested network topology, are periodically (i.e., every 500s) overloaded. This causes congestion for clients connected to these switches on all available paths to the server. The periodic overload sequence is as follows:

- First period — links S3 ↔ S7 and S4 ↔ S7
- Second period — links S5 ↔ S9 and S6 ↔ S9
- Third period — links S3 ↔ S8 and S4 ↔ S8
- Fourth period — links S5 ↔ S10 and S6 ↔ S10.

Simulations are performed with both random and SDN-assisted client selection algorithms, always asked to choose six out of eight clients in each round. The random algorithm selects clients randomly and can thus also include clients with overloaded paths. For the illustrated experiments we refer to a set of 8 clients whose category is randomly chosen, each connected to a switch of the third level of the topology.

To validate the SDN-assisted Delay/Computational-Resources client Selection Strategy (DCSS) different data rates (i.e., 40Mb/s, 60Mb/s, and 80Mb/s) are generated to emulate the overload traffic profile consisting of four periods of 500 seconds each. Additionally, three client categories (high performance, medium performance and low performance) have been implemented, based on the memory and CPU characteristics reported in Table 1. Clients with different hardware characteristics are uniformly distributed.

The α parameter in equation Eq. 2 is set to 0, 0.35, 0.7, and 1 during different tests. In this way, the contributions of computation capability and delay are weighted differently from time to time, with the aim of discovering the most suitable a value with respect to the load conditions of the network. When $\alpha = 0$ it means that the selection strategy is based exclusively on client delays; on the other hand, when $\alpha = 1$, the clients are selected solely based on their computational and memory resources, ignoring delay.

Performance Metrics

Well-known and widely adopted metrics in machine learning are used to evaluate the performance of the learning process, namely Accuracy and Loss.

In short, the former expresses the percentage of success in predicting a value equal to the true value and is therefore appropriate to measure the learning model performance. While the latter typically represents a measure of the error made by the ML model (in other words, the difference between model predictions and real data). Both performance metrics are implemented in the Flower platform and computed on the server side at the end of each federated learning round.

It is also important to evaluate the "*price to pay*" for using the proposed mechanism. To this end, the *signalling overhead* in terms of the amount of additional control data packets traveling from/to the SDN controller is evaluated. This metric is extremely relevant as it can help understand whether the actual impact of network resource orchestration on the tested SDN-based FL framework is such that it is worth introducing it or not.

Proof-of-Concept Results

This section illustrates the results of the conducted test campaign aimed at evaluating the actual convenience, in scenarios with heterogeneous clients, of using Delay/Computational-resources Selection Strategy (DCSS) instead of selection schemes solely based either on experienced delay or on client's computational resources. As a basic benchmark, results relating to a scheme based on fully random choices are also shown.

Overloading Rate	Convergence Time Reduction (sec)			
	$\alpha = 0$	$\alpha = 0.35$	$\alpha = 0.7$	$\alpha = 1.0$
40 Mb/s	0	558.43	1035.40	366.47
60 Mb/s	282.64	1760.86	1091.52	0
80 Mb/s	85.04	1140.57	729.87	0

TABLE 2. Delay/Computational-resources selection strategy: convergence time reduction.

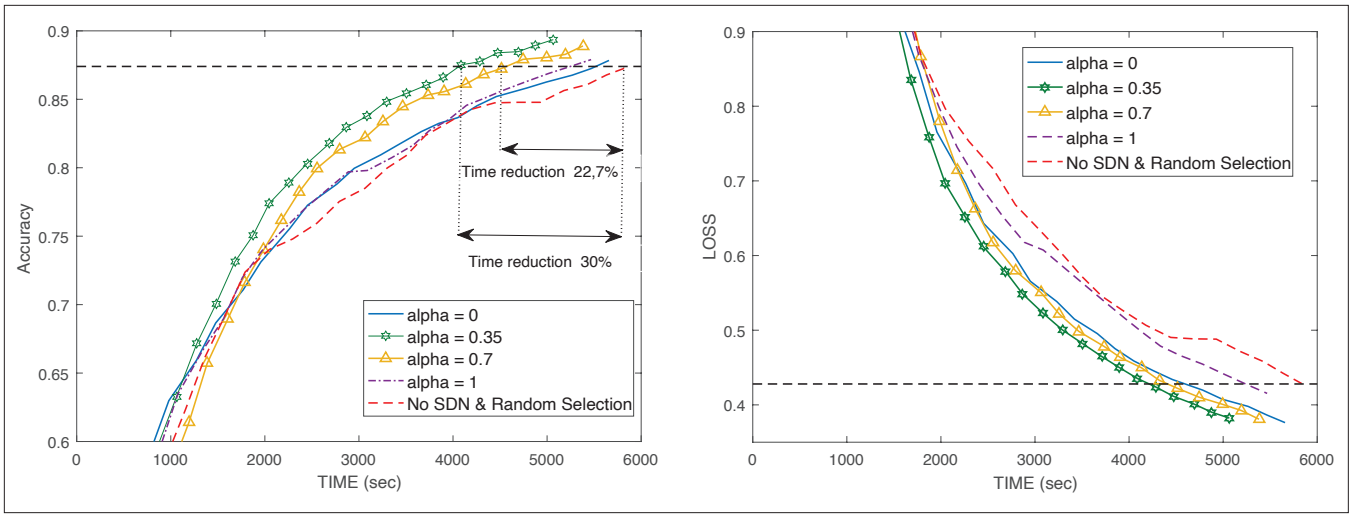


FIGURE 3. Results for 80Mb/s overload scenario: a) Accuracy of delay/computational-resources client selection strategy with and without SDN-assisted routing; b) Loss of delay/computational-resources client selection strategy with and without SDN-assisted routing.

It is evident that scenarios considering both delay and computational/memory resources (α = 0.35 and α = 0.7) consistently outperform those that focus solely either on delay (α = 0) or on computational/memory resources (α = 1). However, as expected, as the overload traffic increases, we observe that it becomes necessary to increase the weight of delay component (i.e., reduce α) because the delay over the paths between FL Server and FL Clients becomes more significant than the one introduced by the training process. Table 2 shows the convergence time reduction comparing to the worst case for each scenario.

The specific trend of the *Accuracy* and *Loss* for the FL process is shown in Fig. 3. In the most demanding communication scenario with high overload network conditions (i.e., 80Mb/s), it is possible to appreciate a significant reduction in times of approximately 30 percent in the distributed FL training process thanks to the coupled use of both SDN-assisted routing and DCSS client selection strategy .

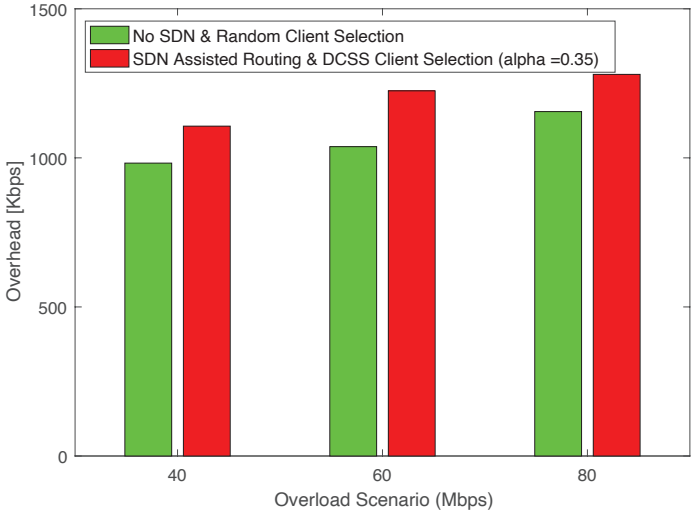
Figure 4a shows the values of the average amount of control data that the controller sends over time; we measured these data over the whole rounds of the FL process (40 rounds) and averaged them.

It is worth noting that, even in cases where neither SDN-assisted dynamic client-to-server routing nor DCSS client selection strategies are used, the SDN controller is still generating additional control data. This expected overhead is due to the presence of an SDN controller implementing its standard procedures in our test network.

As expected, the overhead, in terms of control data packets, increases when using the SDN-assisted DCSS algorithm under all simulated overload traffic conditions; however, the measured average overhead data rate is quite low (e.g., approximately 1.3Mb/s with 80Mb/s of overhead traffic). This reassuring trend further confirms that the additional control overhead, generated by the SDN-assisted FL client selection strategy, is highly acceptable and only weakly depends on the amount of data traffic overhead on the network links.

Finally, Fig. 4b shows the computational overhead of the controller, in terms of percentage increase in memory and CPU usage to implement the DCSS strategy. What is shown is obtained by sampling the RAM and CPU usage in the SDN controller every second and then averaging the sample values over the entire working time.

It can be observed that having the SDN controller carry out the tasks of supporting the DCSS selection strategy in addition to standard routing,



a)

	Overload Profile		
	40 Mbps	60 Mbps	80 Mbps
	SDN controller RAM occupancy (%)		
SDN assisted routing & DCSS client selection	40.93	41.01	41.78
SDN assisted routing & random client selection	39.71	39.75	39.35
	SDN controller CPU usage (%)		
SDN assisted routing & DCSS client selection	5.21	5.91	5.99
SDN assisted routing & random client selection	3.43	3.6	3.75

b)

FIGURE 4. Overhead due to the proposed techniques: a) Signalling overhead; b) Computational overhead at the SDN controller due to the DCSS client selection.

Since clients in the future will increasingly connect to the network wirelessly, a next step could be to extend the proposed system to also include the access segment and its congestion conditions.

implies an only slightly higher use of its CPU and RAM (i.e., about 2 percent under high traffic load conditions). This validates the hypothesis that the computational overhead introduced by client selection features is negligible and can be easily supported by standard controllers.

Open Research Issues

The aim of the presented research is to investigate the role that SDN can play in the FL client selection process and to assess, via initial proof-of-concept, the benefits that could derive from it. Much remains to be investigated. Some ideas, exclusively linked to the role of SDN in supporting FL Client Selection, which do not claim to be exhaustive, are suggested below.

Future studies could consider further elements that distinguish clients and make them heterogeneous, in addition to their computation and communication capabilities; for example, aspects related to the extreme distribution of clients and extreme variability and heterogeneity of the characteristics of the communication channel.

Since clients in the future will increasingly connect to the network wirelessly, a next step could be to extend the proposed system to also include the access segment and its congestion conditions. SDN could play multiple roles, ranging from monitoring the Clients' mobility model (for example, to seamless enable handover in multi-RAT - multi radio access network - environments and avoid client disconnection before completing their FL task), up to intelligently allocating resources to some more performing clients to make them more attractive in the selection phase, which involves the choice of access point, RAT used, access time, and so on.

A further open aspect is to go beyond the simple selection policy used for the sole purpose of carrying out an initial proof of concept. We could use more sophisticated client prioritization and scheduling mechanisms that, for example, leverage measures of their utility. This will involve moving from model-based or data-driven utility definitions to definitions based on the measured utility (also with support provided by SDN) of the entire end-to-end client-server system.

Finally, the same dynamic choice and adaptation of the weights to be used in our policy or in similar policies (for example, the importance to attribute to each client characteristic, as system conditions vary) could be governed by sophisticated and promising reinforcement learning techniques, to be implemented in the proposed orchestrator and to be investigated in future works.

Conclusion

The presented work proposed an SDN-assisted FL client selection strategy aimed at improving the performance of FL processes, which takes into account both the network conditions and the peculiarities (CPU and available memory) of the selected clients during the entire evolution of the learning/ training process. The initial proof-of-concept activity conducted has demonstrated how the proposed DCSS mechanism, thanks to the indispensable support received from the SDN controller in the various phases of the process, can achieve better performances compared to selection policies that only take capabilities into account of heterogeneous clients or just the delays on the paths between client and server. It was also observed that, by carefully combining the two contributions, it is always possible to obtain good performance when the total load conditions on the network vary.

We are confident that the presented study can contribute to the advancement of the state-of-the-art in the field of FL client selection leveraging typical paradigms of softwarized networks and inspire several related future research activities, some of which have been listed.

Acknowledgment

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").

References

[1] T. Li *et al.*, "Federated Learning: Challenges, Methods, and Future Directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, 2020, pp. 50–60.
[2] M. Xu *et al.*, "Multiagent Federated Reinforcement Learning for Secure Incentive Mechanism in Intelligent Cyber–Physical Systems," *IEEE Internet of Things J.*, vol. 9, no. 22, 2022, pp. 22,095–108.
[3] M. Chen *et al.*, "Distributed Learning in Wireless Networks: Recent Progress and Future Challenges," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, 2021, pp. 3579–3605.
[4] J. Xie *et al.*, "A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges," *IEEE Commun. Surveys & Tutorials*, vol. 21, no. 1, 2019, pp. 393–430.
[5] A. Mahmod *et al.*, "SDN-Assisted Client Selection to Enhance the Quality of Federated Learning Processes," *Proc. IEEE Wireless Commun. and Networking Conf.*, Dubai, United Arab Emirates, Apr. 2024, pp. 1–6.
[6] S. Abdulrahman *et al.*, "FedMCCS: Multicriteria Client Selection Model for Optimal IoT Federated Learning," *IEEE Internet of Things J.*, vol. 8, no. 6, Mar., 2021, pp. 4723–35.
[7] T. Huang *et al.*, "An Efficiency-Boosting Client Selection Scheme for Federated Learning With Fairness Guarantee," *IEEE Trans. Parallel and Distributed Systems*, vol. 32, no. 7, 1 July 2021, pp. 1552–64.
[8] Y. Wang and B. Kantarci, "A Novel Reputation-Aware Client Selection Scheme for Federated Learning within Mobile Environments," *Proc. IEEE 25th Int'l. Workshop on Computer Aided Modeling and Design of Commun. Links and Networks*, Italy, 2020, pp. 1–6.
[9] T. Huang *et al.*, "Stochastic Client Selection for Federated Learning With Volatile Clients," *IEEE Internet of Things J.*, vol. 9, no. 20, 15 Oct. 2022, pp. 20,055–70.
[10] Z. Chai *et al.*, "Tifl: A Tier-Based Federated Learning System," *Proc. 29th Int'l. Symposium on High-Performance Parallel and Distributed Computing*, 2020, pp. 125–36.
[11] E. Seo *et al.*, "Auction-Based Federated Learning Using Software-Defined Networking for Resource Efficiency," *Proc. 2021 17th Int'l. Conf. Network and Service Management*, Izmir, Turkey, 2021, pp. 42–48.
[12] P. Tam, S. Math, and S. Kim, "Efficient Resource Slicing Scheme for Optimizing Federated Learning Communications in Software-Defined IoT Networks," *J. Internet Computing and Services*, vol. 22, no. 5, Oct. 2021, pp. 27–33.
[13] J. Mills *et al.*, "Communication-Efficient Federated Learning for Wireless Edge Intelligence in IoT," *IEEE Internet of Things J.*, vol. 7, no. 7, July 2020, pp. 5986–94.
[14] V. Balasubramanian *et al.*, "Intelligent Resource Management at the Edge for Ubiquitous IoT: An SDN-Based Federated Learning Approach," *IEEE Network*, vol. 35, no. 5, Sept./Oct. 2021, pp. 114–21.
[15] B. McMahan *et al.*, "Communication-Efficient Learning of Deep Networks From Decentralized Data," *Proc. Int'l. Conf. Artificial Intelligence and Statistics*, 2017, vol. 54, pp. 1273–82.

Biographies

Ahmad Mahmod received the B.Sc. degree in Communication and Electronic Engineering from Tishreen University, Syria and a Master in Telecommunication Engineering: Smart Sensing, Com-

puting and Networking from the University of Calabria, Italy. He is currently pursuing the Ph.D. degree with the University of Strasbourg, France. He is a member of the Network Research Group, ICube Lab. His research interests focus on software-defined networks, the Internet of Things, and federated learning.

PASQUALE PACE received his Ph.D. in Information Engineering in 2005 from the University of Calabria, Italy. He is currently an Assistant Professor in telecommunications with the Department of Computer Engineering, Modeling, Electronics and Systems (DIMES), University of Calabria, Italy. He has authored more than 100 papers in international journals, conferences, and books. His research interests include cognitive and opportunistic networks, sensor and self-organized networks, software-defined networks and interoperability issues of IoT platforms and devices.

ANTONIO IERA is full professor of Telecommunications at the University of Calabria, Italy. He holds a Master in Information Technology from CEFRIEL/Politecnico di Milano and a PhD from the University of Calabria. His research interests include next generation mobile and wireless systems, and the Internet of Things. He is currently editor-in-chief of the Computer Network Journal, Elsevier.