SDN-Assisted Client Selection to Enhance the Quality of Federated Learning Processes

Ahmad Mahmod
ICube Laboratory
University of Strasbourg
Pole API, Illkirch, France
mahmod@unistra.fr

Pasquale Pace
DIMES - University of Calabria
CNIT - National Inter-University Consortium for Telecommunications
Rende, Italy
p.pace@dimes.unical.it

Antonio Iera
DIMES - University of Calabria
CNIT - National Inter-University Consortium for Telecommunications
Rende, Italy
antonio.iera@dimes.unical.it

***Abstract*—An emerging modality, increasingly used by edge devices, to train machine learning models in a distributed and cooperative way is Federated Learning (FL). It combines an increase in the quality of the learning process with data privacy needs. Alongside the advantages of this emerging paradigm, however, there is a critical factor that risks seriously affecting its effectiveness in future 5G and 6G application scenarios: the possible delays deriving from the scarcity of communication resources to connect the clients to the server, which risks slowing down the process excessively and making it less effective in the presence of new types of real-time applications typical of 5G/6G scenarios. To face this issue, the paper proposes a new approach to client selection that, unlike the various approaches to streamlining FL communications proposed so far, starts from a typically networking research point of view and makes use of the potential of the Software-Defined Networking (SDN) paradigm for the choice and continuous dynamic update of the clients participating in the FL process. This allows to keep the distributed learning process at high levels of effectiveness and efficiency, i.e., guaranteeing an overall time reduction of the FL process under different network traffic load conditions, as demonstrated by the performance evaluation campaign conducted through the implementation of a testbed platform.**

***Index Terms*—SDN for AI, Federated Learning, FL client selection, SDN-based orchestration.**

I. Introduction

Initially proposed by Google [1], FL is rapidly emerging as a distributed paradigm capable of attracting significant interest in several vertical markets to support intelligent pervasive applications in many domains of everyday life. Interest in FL is primarily motivated by its ability to effectively train models on vast amounts of data available at a multitude of remote user devices while ensuring stringent levels of data privacy. In fact, in FL the model is locally trained on data available near remote devices participating in the process (so-called FL Clients) and then the result is sent to a central server (so-called FL Server). The advantage is that the raw data generated by or available to Clients are not sent, but only the parameters relating to the trained model, which are processed by the Server to aggregate them and send the most performing resulting model back to the clients. The procedure is reiterated over several rounds until the desired accuracy level is achieved [1], [2].

It is emerging from the literature that, although FL is still in its initial stages of development, it is widely recognized as one of the most promising solutions to fulfill 6G's vision of ubiquitous AI [3], [4].

An issue that may affect the FL process is the possibility that the scarcity of resources in the network that the traffic of the various clients has to cross, negatively influences the overall achievable performance. This does not represent a problem in cases of support for typical IoT applications in which FL is intended for a high number of mobile and/or constrained IoT devices in which losses and delays in the various update rounds are already expected. The same can be said for several long-term applications that exploit FL for example to make predictions in the field of e-Health or in the environmental field. However, if we place ourselves in an evolutionary context in which several of the typical applications that users will want to use through future 5G/6G platforms are expected to be time sensitive, then the situation changes radically and a reduction of the delay in the learning process becomes mandatory.

The problem of making communications more efficient in the FL process is not new and has already been addressed in the literature, but the vast majority are research and proposals that come from the scientific communities of Learning and aim at adapting the process to streamline the transfer of model parameters between client and server.

In this paper, the proposed approach starts from the very recent literature dealing with the topic of the *"Network for AI"* (as opposed to the more traditional approach of "AI for networks") to propose an SDN-assisted Federated Learning mechanism; in which the possibilities offered by the recent network softwarization techniques are put at the service of FL processes. In a previous initial work [5], we simply demonstrated how the use of Software Defined Networking techniques can actually be able to reduce training times while maintaining the same performance levels.

In this work we go one step further and propose an SDN-assisted Federated Learning solution that leverages a closer collaboration between the Network Controller and the FL Server. We, therefore, hypothesize also a dynamic selection of clients assisted by the SDN controller during the various cycles of the training process. In practice, round after round the SDN controller will not only perform a smart routing of

client-server data flows in order to minimize delays. But it will also take care of making estimates of the delays on the traffic flows originating from all clients in order to be able to provide useful information to the FL Server and allow it to select the clients to be used in the next round by also accounting for data transfer delay and load on the network.

The research presented aims to evaluate the effectiveness of the illustrated approach by conducting a proof-of-concept study. Obviously, in the future it will have to be the basis for other works in which also other parameters of the client devices can be taken into consideration together with the delay (*e.g.* Age of Information, Computation and Memory Capabilities) for an effective selection of the clients themselves.

Specifically, the remainder of this paper is organized as follows. Section II describes some significant related works mainly focused on increasing the performance of ML techniques by using SDN. In Section III, the proposed SDN-assisted client selection strategy is described. Details on implementation choices for the used performance assessment tool are given in Section IV, while performance evaluation analysis and obtained results are discussed in Section V. Finally, Section VI summarizes the main conclusions and presents future research directions.

II. Related works

An intense research activity during the last years has been focused on increasing the performance of SDN using ML techniques, as surveyed e.g., in [6]. Conversely, studies that have tried to understand the role that the SDN paradigm can play in supporting FL (or distributed learning more generally) are few and quite recent.

In the overlay network proposed in [9] the various clients and the server are modeled as auction buyers and sellers and are guided by an SDN controller during their bidding and provisioning activities respectively. The authors of [11], as part of a study on implementing FL application on a distributed architecture based on SDN, also address problems related to synchronization schemes in the context of FL and attempt to alleviate them by using distributed SDN.

Other issues where SDN has been proposed to support FL include, among others, QoS control and content caching [13] in the presence of mobile users; failure recovery in industrial IoT scenarios [15]; slice creation and management of associated optimal forwarding graphs [10]; data collection and secure data delivery for a group of local IoT devices performing distributed ML [12]. Also, the authors of [14] explored software-defined networks and mobile edge computing as architectures to deploy a FL system for e-health treatment recommendation based on Internet of Health Things devices.

Finally, it is also worth mentioning the recent survey work in [16] in which the authors provide an overview of solutions and challenges on the FL over SDN issue as a solution to provide more flexible and effective mechanisms for participant collaborations.

Despite the previous cited works, the contribution of our research is the first, as far as the authors know, which evaluates the potential of a SDN-based framework for the choice and continuous dynamic update of the clients participating in the FL process to maintain the distributed learning process at consistently high levels of effectiveness, efficiency and quality.

This is an approach completely different from (and complementary to) the many interesting improvements to the FL process proposed in the literature to make it more communication-efficient, such as in [7] and [8], which intervene on aspects related to the learning process, while in this paper we aim for a more network management oriented approach.

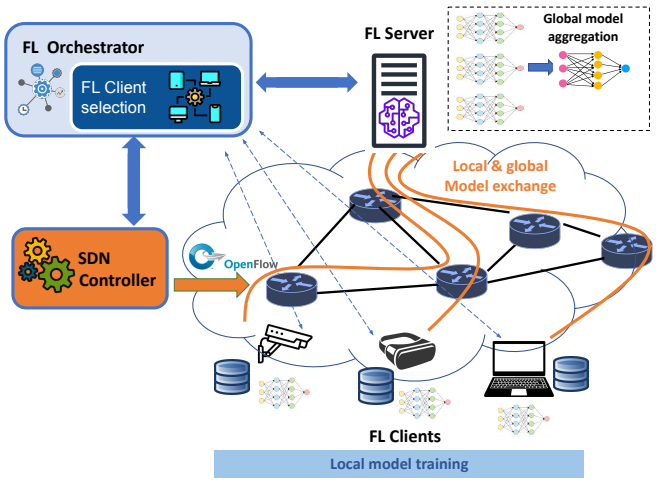

Fig. 1. Reference SDN-based architecture to support FL processes.

III. SDN-based framework to support FL processes

This section introduces the proposed SDN-assisted client selection strategy, by highlighting how the SDN controller can support client selection in FL processes.

A. The Reference Architecture

Figure 1 shows our reference system consisting of a framework to implement Federated Learning in which the main players of the FL process (FL Clients and FL Server) are supported at the Application layer by an FL Orchestrator and at the Network layer by a network resource orchestrator coupled with the SDN Controller.

As for the role associated to the FL Server, we can say that the basic functionalities usually associated to it, i.e the training of the local models received from the client devices and the creation of the global model to send back to them, are obviously still maintained. It works closely with the FL Orchestrator, whose main task is basically to perform client selection and, in addition, through appropriate APIs, establish two-way communications with the SDN Server.

The Orchestrator will always have to send to the SDN Controller information about the selected clients to allow the latter to perform the control, management and monitoring of the traffic flows associated with the FL process at each round. More specifically, the controller will implement algorithms, which, in the first instance, are basic dynamic path selection

and load balancing algorithms, usually implemented in software defined networks, to always guarantee "fast" and non-overloaded client-server paths. Furthermore, it will be able to implement additional functions (not yet implemented in this paper) in which it will also be possible to take into account the outgoing traffic from the various clients not belonging to the FL sessions but in any case which may create additional overloaded. The objective is obviously to continuously assure the minimization of the FL traffic transmission delay on the client-server paths to guarantee a higher quality of the process.

In the opposite direction, the Controller must send synthetic information about the delays experienced on the various client-server paths when sending model parameter updates to the previous round of the FL process, to allow the FL Orchestrator to make the best *selection of the clients* to be involved in the current round. To make a dynamic selection that can allow the server to always choose the clients with the potential best performance in terms of delay, estimates of the delays on the routes from the server to potential clients not participating in the training in the previous round are also monitored by the SDN Controller and reported to the Orchestrator.

Obviously, it clearly emerges that during the dynamic client selection and re-selection procedure, the FL Orchestrator and the SDN Controller continuously exchange control data with each other; which could suggest an excessive overload on the network. We will see in the Section V, showing the results of our proof of concept study, that the additional overload of these exchanges has minimal impact on the load that the network has to handle.

B. SDN-assisted Client Selection Procedure

Always selecting the clients experiencing the best conditions in the network, could for sure limit the risk that a client either is forced to abandon the FL process for reasons related to the saturation of communication resources or it drastically reduces the quality of the whole process due to the accumulated delays caused by an overloaded path to the server. This is the principle driving us to consider in this paper the possibility to carry out the client selection not in a traditional way (for example, in the simplest case, randomly within a set of N devices suitable for training and which have expressed an interest in participating in the FL process) but as illustrated below.

It is important to underline that the focus of our work is not to propose a new sophisticated client selection scheme but rather to evaluate the influence that the use of the SDN paradigm can have on one of these schemes. Therefore, we propose to implement an SDN-assisted client selection strategy to demonstrate that the SDN controller, relying on round-by-round information about the average delay experienced by clients in data transmission, can help select those of higher value for the FL process.

In light of this, the selection strategy we implement takes into account only the lowest delays among the available clients following the assumption that all the clients have the same computational resources. This is in fact the only way that allows to carry out a reliable proof-of-concept of the potential

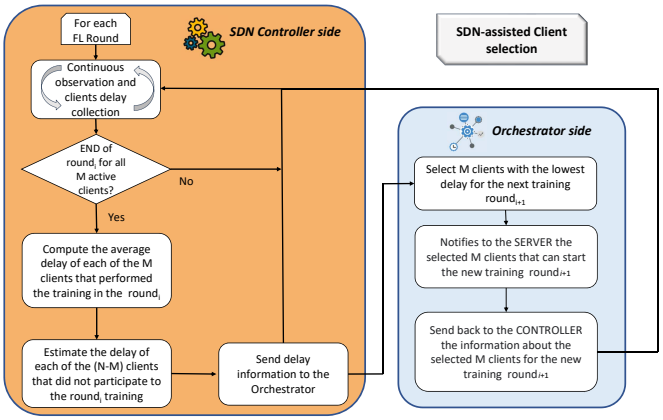

Fig. 2. SDN-assisted client selection scheme: interactions between the Controller and the Orchestrator.

of a clients selection approach assisted by the SDN; in fact, it allows to cut away from the selective process the performance bias that would derive from the presence of more or less performing clients in terms of computation and buffering capabilities for example.

Obviously, the next step, to be carried out after evaluating the positive impact that the dynamic control of the network status and related delays has on client selection, is to investigate the effect of a selection which, in the presence of heterogeneous devices, takes into account parameters relating to both the state of the communication resources and the capabilities of the devices. This is the subject of future research. In detail, our SDN-assisted client selection strategy works as follows:

1) At the beginning of the selection process, a data packet is sent from all N clients that would like to participate in the federation learning process; the M $\subset$ N clients with the lowest delay values are chosen to run the first round of the training process;
2) The learning process is then activated on the selected clients and, during each learning round, the SDN controller keeps updated statistics on the average transmission delays of each selected client.
3) At the end of each round, the new selection process is carried out on the whole set N, considering the average delay values experienced by the M participating clients in the previous round and the instantaneous delay values of the N-M clients not participating in it; this allows the Orchestrator to change, if needed, the set of clients that will perform the training in the next round. The choice in the turnover phase is constantly supported by the controller which also takes care of measuring the delay of the clients that have not participated in the previous round using a background ping mechanism; notifying the orchestrator of this delay at the start of each training round allows it to perform a wiser selection mechanism.

Figure 2 shows the logical flow diagram of the described strategy and gives details on the interactions among the

different modules of the proposed SDN-based FL framework.

IV. Testbed and Performance Metrics

In this section, we provide details on the testbed implemented for performance evaluation, and on the metrics used to evaluate the goodness of the obtained results.

A. Testbed Implementation

The GNS3 (Graphical Network Simulator version 3) [17], open-source software is used to implement and test the proposed SDN-based FL framework. The reference hierarchical network topology used in the test is shown in figure 3 and comprised of an ODL (OpenDaylight [18]) SDN controller located at the top of the tree, along with ten OVSs (Open vSwitches) distributed across three levels (S1...S10). Cross-connections exists between the switches belonging to level-2 and level-3 to provide more route choices for the clients. All links connecting the switches have a capacity of 100 Mbps. The network hosts a Federated Learning (FL) server and eight clients (C1...C8) that uses the *Flower* Federated Learning framework [19], properly installed and configured, to run a FL process based on the *FedAvg* algorithm [1].

In addition, virtual machines (O1...O8), implemented in VirtualBox and exported into GNS3, are included in the topology to support the overloading profile, whose operation is explained later in section V. Table I specifies the characteristics of all the virtual devices used in our tests.

TABLE I
Virtual Machines Specifications.

Virtual Machine	CPU cores	RAM (GB)
CONTROLLER	8	16
SERVER	8	8
Virtual Switches (S1...S10)	4	4
Other virtual devices (O1...O8)	2	4
Clients (C1...C8)	2	4

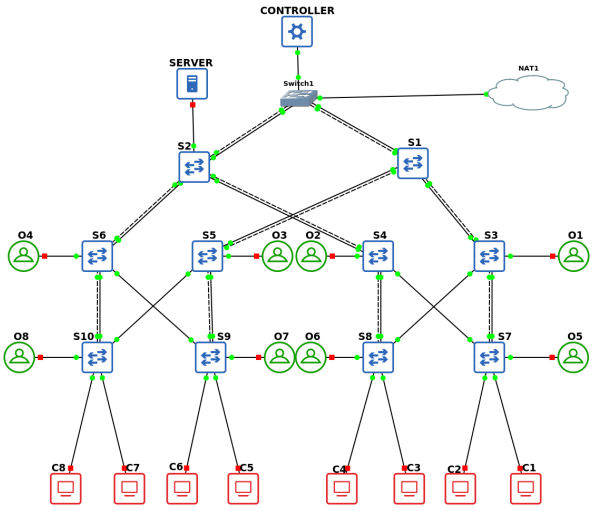

Fig. 3. Network topology implemented in GNS3.

The controller builds a virtual graph of the network that includes switches and connected clients to find the optimal routes from each client to the server. Specifically, in our testbed, it is responsible for:

- *Dynamic Routing* - implemented by running an application on top of the controller that retrieves the information about the participated OVSs, clients, the links, and the connection ports. It builds the weighted virtual network graph and determines the best route from each client to the server via a Dijkstra shortest path algorithm.
- *Delay collection and estimation* - measuring the delay of the clients and delivering it to the server at the beginning of each training round in order to use this information in the client selection phase.

The considered training process consists of 40 evolution rounds, each consisting of 2 epochs. It uses the well-known CIFAR-10 [20] dataset and the [21] DenseNet121 neural network, 33 MB in size with 8.1 million hyper-parameters. The HP Enterprise Proliant DL560 Gen10 hardware platform, equipped with 2x Intel Xeon-Gold 6225N processors (2.3GHz and 24 cores) and 256GB of RAM, is used to run the implemented architecture.

B. Performance Metrics

We considered *Accuracy* and *Loss* as reference performance metrics because they are very well-known and widely adopted metrics in machine learning. In particular, *Accuracy*, typically expressed as a percentage, is the count of predictions where the predicted value equals the true value and can provide a measure of the performance of a classification model. In contrast, *Loss* is a cost function that provides a more nuanced view of model performance by considering the probabilities or uncertainty of a forecast based on how much the forecast varies from the true value. Unlike Accuracy, Loss is not a percentage, rather it is a sum of errors made for each sample in the training or validation sets. Both performance metrics are implemented in the *Flower* platform and computed server-side at the end of each FL round.

V. Results

In this section, we analyze the performance achievable when using or not using SDN to support the FL framework, with particular attention to the SDN-assisted client selection strategy described in Section III-B.

To flexibly test all the proposed features, we use traffic profile configurations that dynamically overload client paths by properly configuring the network devices (O1...O8) to transmit data traffic between them over 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. Figure 4 shows the various overload profiles of the used network links, which alternate during 4 periods of 500s each. The goal is to periodically overload the two available links of each virtual switch, at the third level of the network topology, so as to force its connected clients to

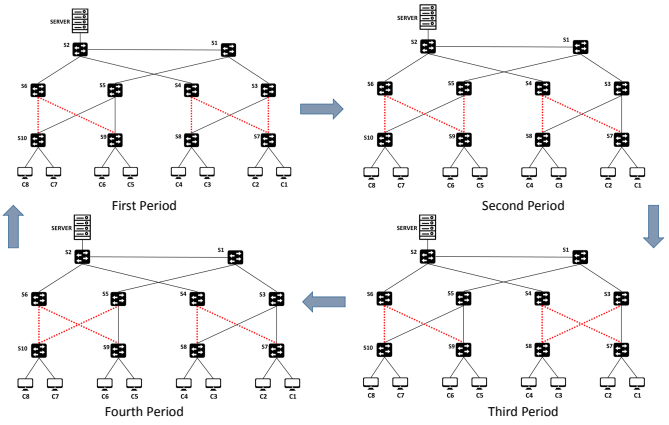

Fig. 4. Time-varying Traffic profile (red dashed lines = overloaded links).

experience congestion in all available paths to the server and observe the reaction of the SDN-assisted FL system.

The simulations are performed with both random and SDN-assisted client selection algorithms which are called upon to always choose six out of eight clients in each round; the random algorithm selects clients randomly and might, therefore, also include clients with overloaded paths. We first test a system with clients (homogeneous, with 4 GB RAM and 2 CPU cores each), subject to a background additional data rate of 25 Mbps on each link. This allows us to assess that both SDN-assisted dynamic routing and client selection, when coupled, can improve performance of the FL framework.

In particular, figure 5 shows that, thanks to the SDN-assisted client selection strategy, only the best-performing clients presenting the least communication delay will be selected, each round, to contribute to the FL process; this wise and flexible choice can allow to reach the maximum accuracy value (above 70% in the case illustrated) saving 20% of the time.

To further validate our results, we also study scenarios with lower and higher additional background data rates of 10 Mbps and 40 Mbps, respectively. It can be seen that the joint use of SDN-assisted routing together with SDN-assisted client selection greatly improves FL processing time reduction the more congested links become. Table II summarizes the time reduction of the FL process in each simulated scenario.

TABLE II
TIME REDUCTION OF THE FL PROCESS VARYING THE TRAFFIC OVERLOAD.

Overload traffic	Time reduction (secs)	Time reduction [%]
10 Mbps	353	9.48%
25 Mbps	2957	20.07%
40 Mbps	4592	47.55%

A. Overhead Analysis

Finally, the *"price to pay"* for using the proposed mechanism, can be evaluated in terms of the amount of additional control data packets traveling from/to the SDN controller. This performance metric is highly relevant because it can help understand the actual impact of network resource orchestration on the tested SDN-based FL framework. Table III shows the values of the average control data amount that the controller sends over time; we measured these data over the whole rounds of the FL process (40 rounds) and averaged them.

The fact that, even in cases where both SDN-assisted dynamic routing and FL client selection strategies are not used, the SDN controller is still generating additional control data has not to surprise. This minimum amount, in fact, represents the expected overhead due to the presence of an SDN controller implementing static routing in our software defined test network and is the reference value for our evaluations. As expected, the overhead, in terms of control data packets, increases when using the SDN-assisted dynamic routing and even more using SDN-assisted FL client selection; however, the measured average overhead data rate is quite low (for example, around 1.2Mbps with 25Mbps of overload traffic).

This reassuring trend further confirms that the additional control load, generated by the SDN-assisted FL client selection strategy, is highly acceptable and only weakly dependent on the amount of overload data traffic on the network links. Figure 6 shows the percentage increase of the overhead using the proposed schemes with respect to a standard SDN communication architecture used to manage a simple static routing configuration for the virtual switches.

TABLE III
OVERHEAD ANALYSIS VARYING THE DATA TRAFFIC.

Overload scenario	SDN-assisted Routing	SDN-assisted Client selection	Overhead [Mbps]
10Mbps	No	No	0.926
	Yes	No	1.101
	Yes	Yes	1.143
25Mbps	No	No	0.94
	Yes	No	1.166
	Yes	Yes	1.205
40Mbps	No	No	0.951
	Yes	No	1.180
	Yes	Yes	1.238

VI. CONCLUSION AND FUTURE WORKS

The main objective of the study presented in this paper was to understand the potential of coupling the SDN paradigm to *Client Selection* policies within a FL process. We started by postulating the hypothesis that a selection of clients that also takes into account the communication delays experienced on client-server paths can reduce the convergence time of FL processes and adapt it to typical real-time applications in future communication scenarios. We then proposed a framework in which an *Orchestrator* of the FL service and an *SDN Controller* work in synergy to select clients and to dynamically guarantee them the best load conditions on the links used by client-server traffics.

The conducted proof-of-concept analysis allowed to bring out interesting potential, confirming the feasibility of the proposed study also demonstrating a low impact in terms of additional signaling messages within the network.

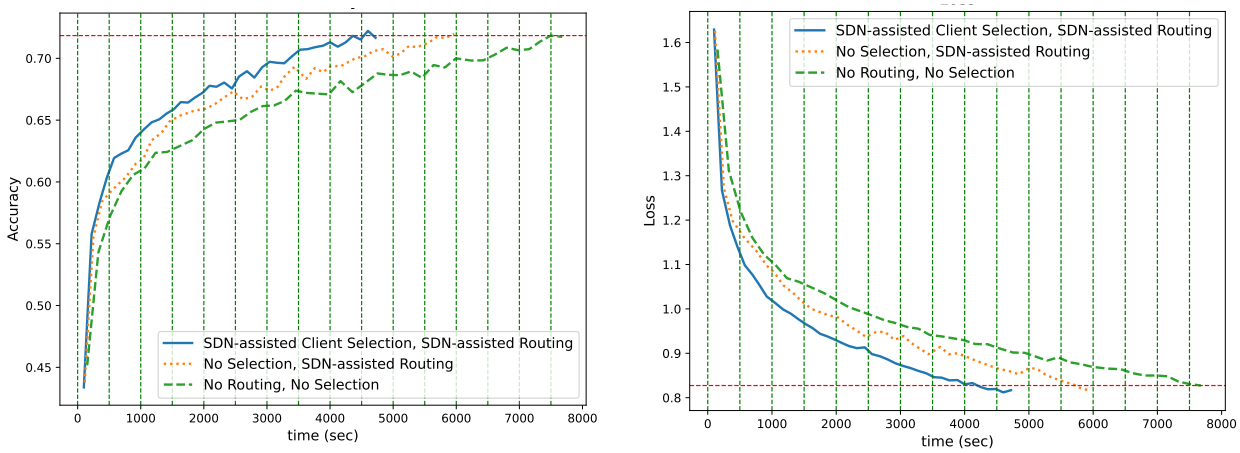

Fig. 5. Accuracy and Loss with 25Mbps overloading data rate.

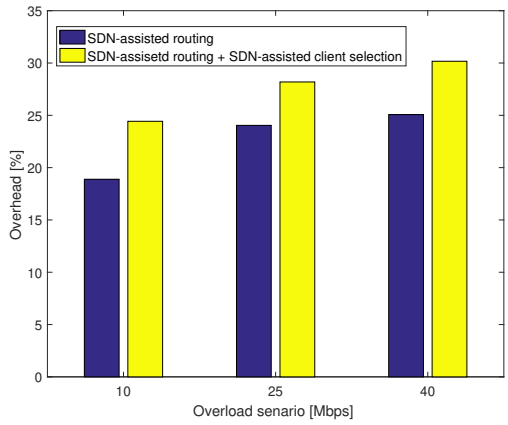

Fig. 6. Signalling overhead increase with respect to the reference scheme.

Future research could take inspiration from the positive results achieved to develop more sophisticated client selection policies considering not only the parameters derived from the SDN Controller but also the capabilities of the client devices and any optimizations on the traffic profiles generated by the clients downstream of the training.

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] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. *Artificial intelligence and statistics*. 2017, pp. 1273–1282.

[2] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed. 2020. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access*, 2020, vol. 8, pp. 140 699–140 725.

[3] K. B. Letaief *et al*. 2019. The roadmap to 6G: AI empowered wireless networks. *IEEE Communications Magazine*. vol. 57, no. 8, pp. 84–90.

[4] F. Tariq *et al.* A speculative study on 6G. 2020. A Speculative Study on 6G. *IEEE Wireless Communications*, vol. 27, no. 4, pp. 118-125, 2020, doi: 10.1109/MWC.001.1900488.

[5] A. Mahmod, G. Caliciuri, P. Pace, and A. Iera. 2023. Improving the quality of Federated Learning processes via Software Defined Networking. NetAISys '23, June 18, 2023, Helsinki, Finland.

[6] J. Xie, F. Richard Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu. 2018. A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges. *IEEE Comm. Surveys & Tutorials*, vol. 21, no.1, pp.393-430.

[7] O. Shahid, S. Pouriyeh, Reza M. Parizi, Q. Z. Sheng, G. Srivastava, and L. Zhao. 2021. Communication efficiency in federated learning: Achievements and challenges. arXiv preprint - arXiv:2107.10996, 2021.

[8] W. Luping, W. Wei, and L. I. Bo. CMFL: Mitigating communication overhead for federated learning. IEEE 39th Int. Conf on Distributed Computing Systems (ICDCS), 2019, pp. 954-964.

[9] E. Seo, D. Niyato, and E. Elmroth. 2021. Auction-based Federated Learning using Software-defined Networking for resource efficiency. 17th IEEE CNSM Conference, 2021, pp. 42-48.

[10] P. Tam, S. Math, and S. Kim. 2021. Efficient resource slicing scheme for optimizing federated learning communications in software-defined IoT networks. *J. Internet Comput. Serv.*, vol. 22, no. 5, pp. 27-33.

[11] R. Firouzi and R. Rahmani. 2022. A Distributed SDN Controller for Distributed IoT. *IEEE Access*, vol. 10, pp. 42873-42882.

[12] J. Mills, J. HU, and G. Min. 2019. Communication-efficient federated learning for wireless edge intelligence in IoT. *IEEE Internet of Things Journal*, 2019, vol. 7, no. 7, pp. 5986-5994.

[13] V. Balasubramanian, M. Aloqaily, M. Reisslein, and A. Scaglione. 2021. Intelligent resource management at the edge for ubiquitous IoT: an SDN-based federated learning approach. *IEEE Network*, vol.35, no.5

[14] Z. Xue, P. Zhou, Z. Xu, X. Wang, Y. Xie, X. Ding, and S. Wen. 2021. A resource-constrained and privacy-preserving edge computing enabled clinical decision system: A federated reinforcement learning approach. *IEEE Internet of Things Journal* (2021), pp. 1–17.

[15] V. Balasubramanian, M. Aloqaily, and M. Reisslein. 2023. Fed-TSN: Joint Failure Probability based Federated Learning for Fault-Tolerant Time-Sensitive Networks. *IEEE Transactions on Network and Service Management*, doi: 10.1109/TNSM.2023.3273396, Early Access 2023.

[16] X. Ma, L. Liao, Z. Li, R.X. Lai, and M. Zhang. 2022. Applying Federated Learning in Software-Defined Networks: A Survey. *Symmetry* 2022, 14, 195. https://doi.org/10.3390/sym14020195

[17] GNS3 software. freely downloadable at https://www.gns3.com/

[18] J. Medved, R. Varga, A. Tkacik, and K. Gray. 2014. OpenDaylight: Towards a model-driven SDN controller architecture. IEEE WoWMoM, Sydney, NSW, Australia, Jun. 2014, pp. 1–6.

[19] D. J. Beutel *et al.* 2020. Flower: A Friendly Federated Learning Research Framework. arXiv:2007.14390v5, 2020. https://flower.dev/

[20] A. Krizhevsky, V. Nair, and G. Hinton. 2005. Cifar10 (Canadian institute for advanced research). Online, 2005. URL http://www.cs.toronto.edu/textasciitildekriz/cifar.html

[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. 2017. Densely Connected Convolutional Networks. IEEE CVPR'17, pp. 2261-2269.