On Predicting Service-oriented Network Slices Performances in 5G: A Federated Learning Approach

Bouziane Brik‡ and Adlen Ksentini‡
‡EURECOM, Sophia Antipolis, France
Email: ‡name.surname@eurecom.fr

***Abstract*—To achieve the vision of Zero Touch Management (ZSM) of network slices in 5G, it is important to monitor and predict the performances of the running network slices, or their Key Performance Indicator (KPI). KPIs are usually monitored, but also, with the advance of Machine Learning (ML) techniques, predicted, in order to proactively react to any service degradation of a running network slices. While network- and computation-oriented KPIs can be easily monitored and predicted, service-oriented KPIs are difficult to obtain due the privacy issue, as they disclose critical information on the performance of services. To tackle this issue, in this paper, we propose to use a new ML technique, known as Federated Learning (FL). It consists in keeping raw data where it is generated, while sending only users' local trained models to the centralized entity for aggregation. Therefore, FL is adequate to predict slices' service-oriented KPIs.**

***Index Terms*—Network slicing, 5G, Machine learning, Federated Learning, Service-level performance.**

I. Introduction

5G system is rapidly growing to support various new use cases related to different vertical industries [1] [2]. These use cases differ from traditional mobile use cases in terms of imposed heterogeneous requirements, such as low access latency, high bandwidth, communication reliability, the support for massive numbers of devices, etc. In this context, network slicing is considered as one of the key enablers of 5G to support these heterogeneous requirements [3]. Network Slicing aims at allowing the share of common physical resources (radio, network, computation) among several tenants using the concept of network softwarization; i.e. building flexible and virtual networks tailored to services. Network softwarization relies on three key technologies: Software Defined Networking (SDN), Network Function Virtualization (NFV) and Cloud computing (central and edge). Usually, a network slice is described and composed by a set of Virtual and Physical resources, in form of Network Functions (known as VNF and PNF), deployed and interconnected together on top of a shared infrastructure. The description of a network slice is based on a Network Slice Template (NST), which is used by an orchestrator/management framework[1], to orchestrate and manage life-cycle of network slices; instantiation, configuration, activation, runtime management and destruction. Recent trends in management and orchestration of network slices aim to achieve the concept of Zero touch network and Service Management (ZSM) [4]; which consists in automating the orchestration and management process of running network slices, without involving humans. Reaching ZSM objectives requires a heavy usage of Machine Learning (ML) techniques; aiming at building the necessary components to analyse the network slice performances and auto build the decision mechanisms to react (or proactively react) accordingly. Usually, the first step consists in monitoring the performances of a network slice, and uses the monitored data to run analytic functions (using learning mechanism or predefined policies) that aim at detecting performances degradation or Service Level Agreement (SLA) violation. The performances of a network slice or Key Performance Indicator (PKI), can be network oriented, service-oriented or computation oriented [5]. The network- and computing-oriented KPI can be easily monitored via the SDN controller and NFV Infrastructure (NFVI) manager. Neverthe-

[1]3GPP uses the term Communication Service Management Function (CSMF), while ETSI uses Management and Orchestration (MANO) to refer to the orchestration and management element handling the Life Cycle Management (LCM) of network slices

less, service-oriented KPIs, such as the response time of the Mobility Management Element (MME), IP address allocation of a Dynamic Host Control Protocol (DHCP), the number of packets handled by a router, are more difficult to obtain, as they are linked to a running vertical service or application, where privacy and confidentiality of the data regarding the service or the application are required. Traditionally, ML schemes are cloud-centric and require the data to be sent and processed in a central entity, e.g. a cloud server or a data center. However, these schemes are not suitable for the network slicing case, due to: (i) the inaccessibility of private data, since the service-level metrics may contain personal and sensitive information of slice tenants; (ii) Network slices are isolated from each other, which makes difficult to collect data and build centralized machine learning models that identify the performances of network slices. Therefore, there is a critical need to go toward decentralized learning solutions to handle efficiently distributed private sub-data sets of network slices.

Federated Learning (FL) [6] [7] is a novel concept which is gaining momentum as a decentralized approach in ML. In FL, agents use their local data to train, cooperatively, local learning models. These agents, then send the local models, i.e. models' weights, to a FL server for aggregation. Thus, FL allows keeping the private data where it is generated and training learning models in a distributed manner. Accordingly, FL is much more appropriate to build learning models, without exchanging data; hence respecting privacy and confidentiality of the training data. In this paper, we aim to use FL to build a learning model that predicts service-oriented KPI of running network slices. We focus on predicting a service-level KPI of MME, which is run as a VNF in a network slice [8]. This KPI corresponds to the latency to handle UE attach requests by a MME or the response time, when considering different parameters: number of used CPU, number of attach request per second, etc. We use OpenAirInterface (OAI) [9] to generate two different data sets, within two different network slices. Then, we applied FL to build a learning model to predict the KPI. Finally, we demonstrate the performance of the FL model, by comparing it to the centralized ML model. To the best of our knowledge, this is the first work that relies on FL to handle the service-level performances of network slices.

The remainder of this paper is organized as follows. Section II gives a background on the use of FL concept for networks. Section III introduces our proposed scheme and shows how we use the FL concept to predict slices' service-oriented KPIs. We discuss the obtained results in section IV and conclude the paper in section V.

II. Federated learning for Networks

FL is a distributed learning approach that recently was introduced by Google [6] [7]. The FL concept is composed of two main entities: the clients (participants) and the Federated Learning (FL) server. Initially, the FL server generates an initial global model G_0 before broadcasting it, along with data type requirements and training hyper parameters, to targeted clients. Then, each participant i starts to collect new data and updates parameters of its local model L_i^j, based on the global model G^j, where j is the current iteration index. When receiving the local models from clients, the FL server aggregates them and sends back the updated model parameters to the participants. In order to aggregate local models, the FL server uses an aggregation algorithm [7], namely *FederatedAveraging* (or *FedAvg*); which is based on distributed Stochastic Gradient Descent (SGD) algorithm.

In [10], we highlighted possible applications of FL in unmanned aerial vehicle (UAV) enabled wireless networks by addressing the suitability and the usage of FL to deal with UAVs' challenges. In [11], the authors address the challenge of providing an ultra-reliable and low-latency vehicular communication, by proposing a new joint transmit power and resource allocation framework based on FL concept. Thus, a power minimization problem is formulated, while ensuring low latency and high reliability. As end-to-end latency depends on both over-the-air latency and queuing latency, each vehicle collects statistics about when their queue length can exceed a threshold, before building a prediction model in a federated way. Vehicles then share their local models with the RSU for aggregation. Hence, the prediction of such information about vehicles' queues can be exploited to assign the needed resources to each vehicle. The authors in [12] study how the computation and communication latency of wireless nodes impact the FL convergence time and accuracy. They formulate an optimization problem which deals with two trade-offs: (i) between communication and computation latency determined by a learning accuracy level, and

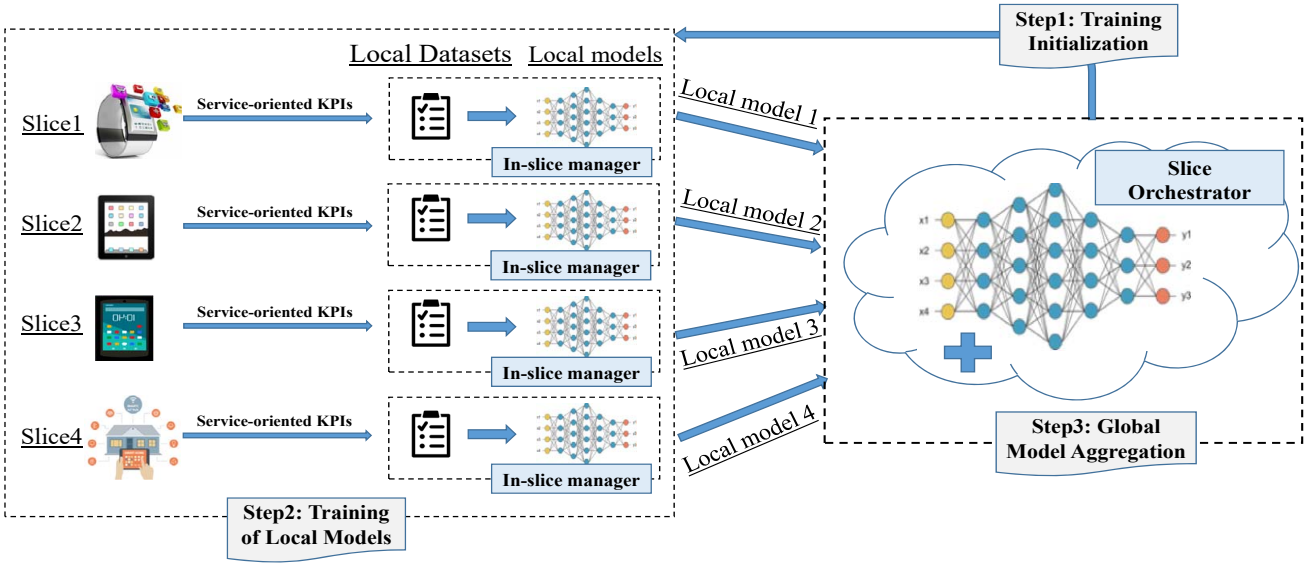


Fig. 1. Training Process of Federated Learning.

thus (ii) between users energy consumption and the Federated Learning time.

III. Federated Service-oriented KPIs Prediction

In this section, we describe how the FL concept is used to deal with the prediction of service-oriented KPIs. Although the presented model is applied to predict the service-level KPI of a MME, it can be easily extended for other type of service-oriented KPI and for other type of VNF. The main requirement is to have access to the data on the VNF running the service and monitor its performance.

A. Modeling the prediction of service-oriented KPIs

As stated earlier a network slice is composed mainly by VNFs, with some few PNFs. To ensure a scalable management of network slices, in [13], the authors introduced the concept of in-slice manager. Besides monitoring service-level KPIs of its network slice, the in-slice manager is in charge of taking local decision to ensure optimal performances of the network slice. In this work, we assume that the in-slice manager is in charge of monitoring the service-level KPI, and training the local FL model. Let $M = \{1, \ldots, N\}$ denotes the set of N local in-slice manager (clients), and each $j \in M$ has a private Data Subset $D_{j \in M}$. Each in-slice manager j trains a *Local* Model L_j using its local data subset $D_{j \in M}$. It then sends only the local model parameters, i.e. model weights, to a central node, e.g. the slice orchestrator. It worth mentioning that the latter is in charge of LCM of end-to-end network slices [8], and interacts with the NFV Orchestrator (NFVO), RAN (Radio Access Network) as well as the SDN controller to deploy and monitor the performance of the network slice. All received local models are aggregated to create a *Global* Model $GLM = \cup_{j \in M} L_j$, using the *FedAvg* algorithm. Fig. 1 illustrates our FL training process.
To train the learning model, FL uses a deep neural network (DNN) which comprises three main layers: (i) an input layer, (ii) multiple hidden layers that map an input to an output, and (iii) an output layer. A weighted and bias-corrected input values are passed over an activation function, such as ReLu and Softmax functions [14], to obtain an output [14]. In addition, the weights are updated using the SGD method with a predefined rate (learning rate), such that the loss function, i.e., distance between the real and model output, is minimized. The training process is repeated over many epochs, i.e., full passes over the training subset, for accuracy improvement.

B. Data Collection

Like any ML model, FL requires data to build the learning model. To generate the needed data, we conducted a set of experimentation using OpenAirInterface (OAI) platform. OAI provides a set of

open source software to run the 4G core network and radio access network. We emulated a set of UEs and one eNodeB using the OAI simulator (OAISIM) and two instances of OAI MME, run as VNF inside two separated network slices. It is worth nothing that OAISIM allows to run both UE and eNB stacks in a computer, and generates attach request messages as well as real user plane traffic following the LTE standard. The MME treats the traffic received from OAISIM as real UE traffic. By increasing the number of OAISIM instances, we were able to generate up to 2500 attach per second, covering different traffic intensity.

We considered two different configurations of MME VNF: 1 CPU and 1 GB of memory (network slice 1); 2 CPUs and 2 GB of memory (network slice 2). Thus, we generated a local dataset for each network slice, by varying the number of attach request per second. We obtained two datasets comprising four features: Slice ID, number of CPU resource, number of memory resource, and number of attached users, as input data, and average duration of user attachment, as output data.

The size of input space is $[d; f]$, where the number of samples is $d = 500$ rows for each network slice. The number of columns corresponds to the number of input data $f = 4$. The size of output space is $[d; 1]$, which corresponds to the UE attach duration time value associated to the input feature.

C. Data Pre-Processing

Once the datasets are collected, the data should be pre-processed in order to be fed to the local learning algorithm for the local training phase. This step is crucial as the data format can affect the prediction accuracy of the local learning algorithm. This step ensures that the data is on the same scale, format and includes all required features. So, formatting the data efficiently will enable learning algorithms to learn more in the training phase, and hence predict with high accuracy.

D. Federated Training Process

The process of training a global learning model is composed of three main phases:

1) Training Initialization Phase: In this phase, the central node specifies the training hyper parameters such as, the learning rate and the number of epochs. It also builds an initial global model GLM_0, which along with the training hyper parameters are broadcasted to selected clients (participants).

2) Local Models Training: Each in-slice manager i starts to update parameters of its local model LM_i^j, where j is the current iteration index. To do so, each learner splits its local data into many batches. Then, it performs the average gradient on each batch regarding the current model, with a given learning rate and during a number of epochs. The updated parameters are then periodically sent to the central node. **Algorithm 1** details the main tasks performed by in-slice manager i.

Algorithm 1 *FedAvg* on in-slice manager i

Require: Local epochs EP, Batches size S, learning rate η.
Ensure: Local model LM^{j+1}.

```
SliceUpdate(i, LM)
for ep from 1 to EP do
    batches ← (split data D_i into batches of size S)
    for Batch T ∈ batches do
        LM ← LM − η∇f(LM, T) (∇f(LM, T) is the average gradient on batch T at the current model LM)
    end for
end for
return LM to the global slice orchestrator.
```

3) Global Model Aggregation: The slice orchestrator first aggregates the receiving local model. It then sends back the updated model parameters to the in-slice managers. We note that the main objective of the slice orchestrator is to minimize the average global loss function Loss(GLM^j).

$$Loss(GLM^j) = \frac{1}{M} \sum_{i=1}^{i=N} Loss(LM_i^j) \quad (1)$$

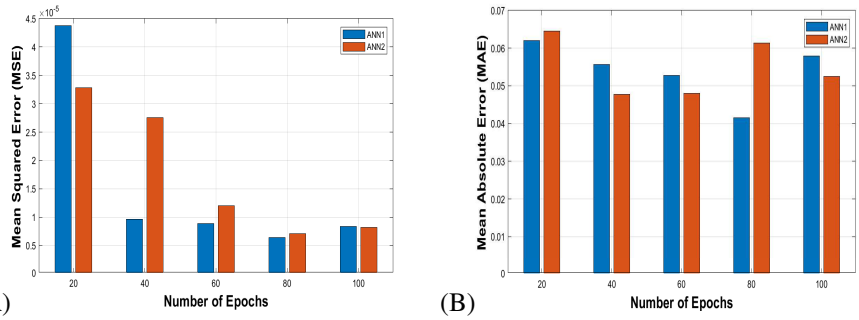
Algorithm 2 gives details on the main tasks of the global model runs at the slice orchestrator.

IV. Implementation and Results

In this section, we validate our FL-based scheme through two implementation prototypes.

A. Implementation Setting

Using federated Tensorflow tool [15], we implement two artificial neural networks (ANN). The first one (ANN1) comprises 2 hidden layers while the second one (ANN2) 3 hidden layers. We also note that both ANN share the same activation functions



(A) (B)

Fig. 2. Performance evaluation of centralized scheme. (A) MSE. (B) MAE.

Algorithm 2 *FedAvg* on the slice orchestrator

Require: Number of rounds $Round$, number of network slices M, Batches size B, Learning rate η, Local epochs E.

Ensure: Global model GLM^{k+1}.

Initialize GLM_0
for $k = 1$ **to** $Round$ **do**
  P = set of network slices
  for Each in-slice manager $i \in P$ in parallel **do**
    $L_i^{k+1} \leftarrow SliceUpdate(i, L^k)$
  end for
  $GLM^{k+1} \leftarrow \frac{1}{|P|} \sum_{i=1}^{i=|P|} L_i^{k+1}$
end for
return GLM^{k+1} to in-slice managers.

and optimizer (SGD). Besides evaluating the performances of our FL-based scheme, we compare it with a centralized learning scheme that implements the same ANNs and uses both local dataset as one central dataset. Moreover, we evaluate both schemes using three main metrics:

- Mean Squared Error (MSE): it shows the average squared error of predictions. It measures the square difference between predictions and real values and then average them, as follows:

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \widehat{y}_i)^2}{n} \quad (2)$$

  Where n is the length of the test set, $\widehat{y}_i$ indicates the predicted value, and y_i is the real value of the test data sample i.
- Mean Absolute Error (MAE): it shows how wrong the predictions were by averaging the absolute differences between prediction and real values, using the following formula:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \widehat{y}_i|}{n} \quad (3)$$

- Communication Overhead (samples): this metric reflects all exchanged information between the centralized node and participants, in terms of data samples (centralized scheme), local and global learning models (federated scheme).

TABLE I
IMPLEMENTATION PARAMETERS.

Parameters	Values
Dataset	
Number of client (network slices)	2
Number of samples by a client	500
Number of input variables	4 variables
Number of output variables	1 variable
Percentage of training set	80% of the dataset
Percentage of test set	20% of the dataset
Deep Learning	
Deep learning Tool	Tensorflow
FDL Tool	Federated Tensorflow
Aggregation algorithm	FedAvg
Deep learning algorithm	Artificial Neural Networks (ANN)
Number of ANN1's hidden layers	2 hidden layers
Number of ANN2's hidden layers	3 hidden layers
Number of hidden layers' neurons	6 neurons
Activation functions	Relu (hidden layers) Linear (output layer)
Optimizer	Stochastic Gradient Descent (SGD)
Loss functions	MSE, MAE
Learning rate	0.01
Batch size	5 samples
Number of epoch	$[20, 130]$ epochs

Table I summarizes the parameters and their settings in our simulation.

B. Results

Fig.2-(A) and (B) depict, respectively, the MSE and MAE of the centralized learning scheme ac-

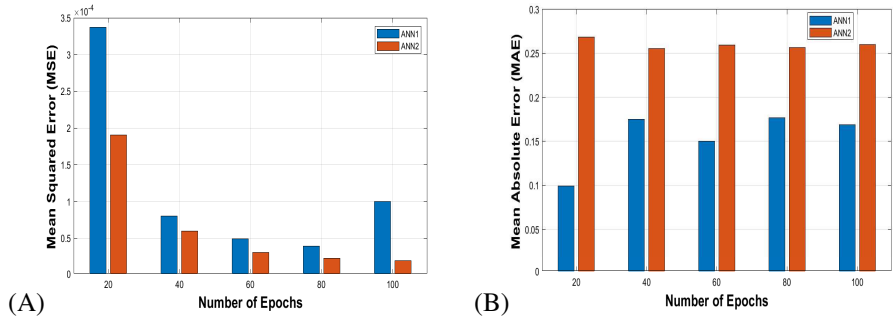


Fig. 3. Performance evaluation of federated scheme. (A) MSE. (B) MAE.

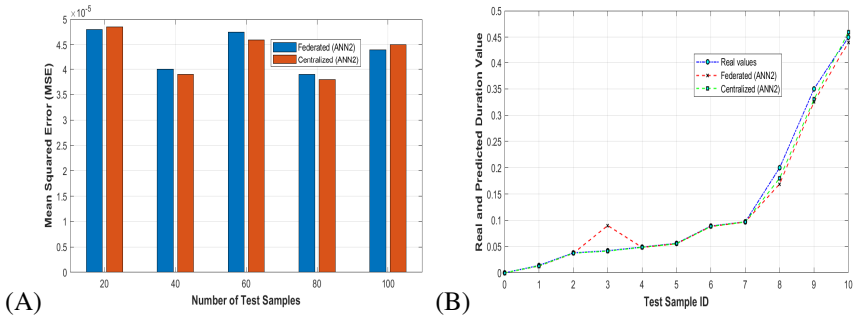


Fig. 4. Performances comparison between centralized and federated schemes. (A) MSE of ANN2 and on test dataset. (B) Real and Predicted average delay values.

cording to the number of epoch. We observe that the ANN2 improves prediction accuracy by minimizing both loss functions (MSE and MAE), when increasing the number of epochs. In addition, we remark that both neural networks generate a lower loss function using MSE, when compared to MAE. Hence, for centralized scheme, the ANN2 using the MSE is more adequate for our dataset as it gives a good prediction accuracy.

On the other hand, Fig.3-(A) and (B) show the MSE and MAE of FL scheme, respectively, while varying the number of epoch. As we can see, the ANN2 outperforms ANN1 when considering the MSE, whatever the number of epoch. However, the ANN1 improves prediction accuracy when using the MAE. Also, we observe that the accuracy loss is lower when using the MSE than MAE for both ANN1 and ANN2. Like the centralized scheme, our FL scheme is more accurate when implementing the ANN2 and using the MSE as accuracy metric.

Fig. 4-(A) shows a comparison between both schemes in term of MSE on the test dataset, which allowed us to evaluate the performances of our model on data that it has not seen before. We focus only on ANN2 results, since it allows both mechanisms to achieve the best performances. We observe that both schemes exhibit almost the same performances, even when the number of test samples increases. To validate these results, we drew in Fig. 4-(B) comparison between the real and predicted average delay values of ten test samples.

Mostly, we notice that the predicted delay values, by both schemes, are the same and closer to the real delay values. These results confirm Fig. 4-(A) results; showing the efficiency and the accuracy of the FL scheme in predicting the average delay.

Fig. 5 shows the generated communication overhead between both schemes. Based on the frequency of local model updates, we consider two federated schemes: in the first one, the local learners update their local models every one learning epoch (Federated Scheme(1-Epoch)), while the local models are updated each ten epochs (Federated Scheme(10-Epochs)). We remark that the federated learning scheme significantly reduces the communication overhead as compared to the centralized scheme. We also observe that the centralized scheme generates a stable overhead whatever the number of epoch. This is mainly due to the fact that FL avoids transferring data samples to the central node and sends only the local models periodically. Thus, the communication overhead will mainly depend on

the transfer frequency (as discussed in the next subsection). However, the learners in the centralized scheme must send their data sample towards the central node in order to generate the learning model. This transfer is performed only once which causes the communication overhead to be stable.

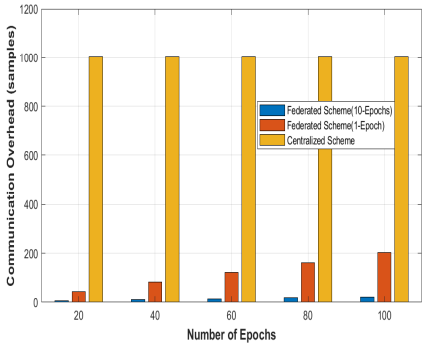


Fig. 5. Performance evaluation comparison in terms of generated communication overhead.

Consequently, we can deduce that the FL scheme is able to achieve the same performances as the centralized one, without explicitly accessing to privacy-sensitive information of network slices. We can also claim that the FL can help to preserve slices isolation especially in multi-tenants environment; while reducing significantly the data to transfer (i.e. less communication overhead) to the central node.

C. Discussion and Open Issues

Although the FL scheme can reach the same performances as the centralized one in terms of prediction accuracy, while preserving network slices' privacy-data and reducing communication overhead, two main challenges remain to predict slices' service-oriented KPIs. Firstly, in our study, we considered only two network slices as we were not able to run parallel network slices due to resource constraint. But, the performance of FL, in terms of prediction accuracy, depends highly on the number of network slices participating to the global learning model. Higher the number of network slices is, higher the global model accuracy is. But, considering a high number of network slices requires a high usage of network resources (communication and computation), which may affect the global performance of running network slices. Hence, more investigations are needed to study the trade-off between FL scalability and global learning performance. One potential solution to address such challenge is to divide network slices into clusters according to their type (low latency or high bandwidth), or their UEs locations. Then, only slices belonging to the same cluster are allowed to participate in the global learning model.
Secondly, according to our results, we deduced that the prediction accuracy increases as the number of epoch increases, which reflects directly the frequency of local model updates. However, updating local models with high frequency increases the network overhead and consumes more network and computing resources. Therefore, more studies are required to deal with the trade-off between capabilities of slices' local resources and performance of global learning model. The designed FL algorithm must also be robust to such challenge by, for instance, adjusting the frequency of local model updates according to the available network and computing resources.

V. Conclusion and Future Works

In this paper, we addressed the challenge of predicting network slices' service-oriented KPIs. Due to privacy-sensitivity of data on services' performances, we applied FL to predict slices' service-oriented KPIs, by locally keeping data in each network slice, and only slices' local learning models are aggregated in a central entity. To demonstrate the feasibility of our scheme, we focused on predicting one of the key service-oriented KPI of a VNF running inside a network slice, which is the response time of the MME. We generated two datasets using OpenAirInterface platform, while varying the CPU, memory dedicated to the VNF (MME) as well as the number of attach request per second. Then, we applied a federated learning algorithm to predict the attachment duration of UE by using the generated datasets locally at each network slice. We implemented the devised model using Tensorflow, and the obtained results showed the efficiency of our scheme in reaching a good prediction accuracy, while ensuring the privacy issues of such service-oriented KPI.

Acknowledgement

This work has been supported by the European Union's H2020 MonB5G (grant no. 871780) project.

REFERENCES

[1] A. Gupta and R. K. Jha, "A survey of 5g network: Architecture and emerging technologies," *IEEE Access*, vol. 3, pp. 1206–1232, 2015.

[2] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5g wireless networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 1617–1655, thirdquarter 2016.

[3] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2429–2453, thirdquarter 2018.

[4] "Zero touch network service management (zsm)," https://www.etsi.org/technologies/zero-touch-network-service-management, accessed: 01-16-2020.

[5] S. Kukliński and L. Tomaszewski, "Key performance indicators for 5g network slicing," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, June 2019, pp. 464–471.

[6] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," 2019.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, 2017, pp. 1273–1282.

[8] I. Afolabi, T. Taleb, P. A. Frangoudis, M. Bagaa, and A. Ksentini, "Network slicing-based customization of 5g mobile services," *IEEE Network*, vol. 33, no. 5, pp. 134–141, Sep. 2019.

[9] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "Openairinterface: A flexible platform for 5g research," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, p. 33–38, Oct. 2014.

[10] B. Brik, A. Ksentini, and M. Bouaziz, "Federated learning for uavs-enabled wireless networks: Use cases, challenges, and open problems," *IEEE Access*, vol. 8, pp. 53 841–53 849, 2020.

[11] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Transactions on Communications*, pp. 1–1, 2019.

[12] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, April 2019, pp. 1387–1395.

[13] S. Kukliński, L. Tomaszewski, T. Osiński, A. Ksentini, P. A. Frangoudis, E. Cau, and M. Corici, "A reference architecture for network slicing," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, June 2018, pp. 217–221.

[14] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2019.

[15] "Tensorflow federated," https://www.tensorflow.org/federated, accessed: 11/29/2019.