FeD-TST: Federated Temporal Sparse Transformers for QoS prediction in Dynamic IoT Networks

Aroosa Hameed, John Violos, Nina Santi, Aris Leivadeas, Nathalie Mitton

FeD-TST: Federated Temporal Sparse Transformers for QoS prediction in Dynamic IoT Networks

Aroosa Hameed, John Violos, Nina Santi, Aris Leivadeas *(Senior Member, IEEE), Nathalie Mitton*

***Abstract*—Internet of Things (IoT) applications generate tremendous amounts of data streams which are characterized by varying Quality of Service (QoS) indicators. These indicators need to be accurately estimated in order to appropriately schedule the computational and communication resources of the access and Edge networks. Nonetheless, such types of IoT data may be produced at irregular time instances, while suffering from varying network conditions and from the mobility patterns of the edge devices. At the same time, the multipurpose nature of IoT networks may facilitate the co-existence of diverse applications, which however may need to be analyzed separately for confidentiality reasons. Hence, in this paper, we aim to forecast time series data of key QoS metrics, such as throughput, delay, packet delivery and loss ratio, under different network configuration settings. Additionally, to secure data ownership while performing the QoS forecasting, we propose the FeDerated Temporal Sparse Transformer (FeD-TST) framework, which allows local clients to train their local models with their own QoS dataset for each network configuration; subsequently, an associated global model can be updated through the aggregation of the local models. In particular, three IoT applications are deployed in a real testbed under eight different network configurations with varying parameters including the mobility of the gateways, the transmission power and the channel frequency. The results obtained indicate that our proposed approach is more accurate than the identified state-of-the-art solutions.**

***Index Terms*—Internet of Things, QoS forecasting, Edge Computing, Federated Learning**

I. Introduction

THE proliferation of Internet of Things (IoT) applications generated a continuous stream of time-stamped data of various granularity. These data need to be analyzed in order to take actions and add the necessary intelligence to the IoT applications. Edge Computing can invoke this intelligence much faster by placing communication and computational resources closer to the source of data [1]. However, besides the analysis of the content of the data, there is a second type of analysis, the network analysis, which is equally important [2]. Through this analysis, the traffic characteristics of the IoT applications can be learned and the network conditions can be estimated, in order to better schedule the resources of the access and edge networks and to take preventive actions in case of network performance deterioration.

However, this type of network analysis is a hectic and challenging process for various reasons. First of all, IoT devices belonging to different applications follow disparate data generation modes (i.e., poll-based, periodic, event-driven, etc.). Secondly, IoT access networks are usually wireless and lossy, and they operate in unlicensed bands. This makes them prone to interference and unstable connection. Lastly, the IoT gateways or the devices themselves can be mobile (i.e., robots, drones, etc.), which can also affect the performance of the communication. These complexity levels could lead to a varying Quality of Service (QoS) behavior for each application. Hence, it is important to propose an efficient QoS forecasting model, which will use the time series data of how packets are generated by the IoT devices (i.e., when packets are transmitted/received) along with the network characteristics of the access network (i.e., frequency channel, etc.).

Time series forecasting is the task of analyzing time-stamped data, to make accurate future predictions that can be used in strategic decision making. However, time series forecasting becomes challenging when working with data that contain variables that change frequently (as in the case of IoT access networks) and events that cannot be controlled [3] (as in the case of IoT data generation). In such scenarios, it is important to utilize appropriate techniques to reduce uncertainty and enhance the accuracy of the predictions.

However, due to the complexity and continuously shifting patterns of IoT data under different controllable and uncontrollable factors, it is difficult to apply traditional ML approaches while dealing with the non-stationary QoS forecasting problems. Additionally, the most prominent Machine Learning (ML) techniques used to predict QoS metrics, such as [4]–[14], lack the ability to handle both long and short term dependencies at the same time, as training over longer sequences of past data degrades the accuracy of the prediction [15]. Recently, transformer models have been applied for time series forecasting [16], because of their ability to capture long-term dependencies using the combination of positional encoding and multi-head self attention mechanisms. Nonetheless, the transformer model requires time and memory that grows quadratically with the sequence length, which excludes their use on long sequences. Consequently, this motivated us to introduce a sparse transformer model, that entails less complexity for long sequences without sacrificing performance.

Nonetheless, a typical problem of both transformer models (sparse and traditional) is that they require large amount of data to be trained. In the IoT context, this could lead to stressing the computing entity that has to gather the time series data and analyze them. Additionally, to increase local efficiency and to respect confidentiality requirements, the data streams of different IoT applications may have the exigency of being treated separately. Accordingly, these two reasons inspired us to resort to a more distributed machine learning approach

A. Hameed, J. Violos and A. Leivadeas are with the Department of Software and Information Technology Engineering, École de Technologie Supérieure, University of Quebec, Montreal, Canada. N. Santi and N. Mitton are with INRIA Lille-Nord, France. E-mails: {aroosa.hammed, ioannis.violos}.1@ens.etsmtl.ca, aris.leivadeas@etsmtl.ca, {nina.santi, nathalie.mitton}.inria.fr

leveraging the Federated Learning (FL) technique. FL allows multiple clients to train a shared global model, by aggregating the local updates from decentralized and distributed clients, in order to improve the global model's accuracy, while tackling data scarcity and preserving the privacy of clients data [17]. To the best of our knowledge, this is the first work that introduces an adaptation of federated sparse transformers to forecast the QoS metrics of an IoT communication network.

More specifically, in this work, we deploy three different IoT applications in a real testbed to predict typical QoS metrics. For this prediction, a sparse transformer-based architecture is introduced that efficiently leverages the time dependency by investigating both the short and long input sequence dependencies without any performance degradation. Finally, we explore the effectiveness of our QoS forecasting sparse transformer in a FL setting in order to enhance the accuracy while coping with the data heterogeneity, scarcity and privacy issues. Accordingly, the main contributions of this work can be summarized as follows:

- We consider three real time IoT applications that present different requirements in terms of number of devices and data generation distribution. The applications are deployed in a real testbed comprised of 100 IoT devices and 3 mobile robots connected over an IEEE 802.15.4 access network, creating a lossy communication under a random access network for different mobility, transmission power and frequency channel configurations (Section IV).
- We design and implement a QoS predictor based on the Sparse Temporal Transformer approach that models the temporal dependencies within long input sequences of data and computes the attention scores using only a subset of the input positions (Section V-C). Both the univariate and multivariate predictions of four major QoS metrics such as Throughput, Packet Delivery Ratio (PDR), Packet Loss Ratio (PLR) and Latency are provided (Section VI).
- We implement a FL scheme with multiple clients to enable the collaborative learning of our sparse transformer models by leveraging the distributed historical data of different IoT applications (Section V).

The remainder of the paper is structured as follows. Section II highlights relevant works on QoS forecasting in an IoT/Edge environment. Section III introduces the system model and problem statement. Section IV describes the testbed implementation and the dataset generation. In Section V, the detailed design of our proposed solution, named FeD-TST, is provided. Section VI discusses the attained results, while section VII summarizes our conclusions and offers some future directions.

II. Related Work

QoS forecasting is a key concern for IoT applications as it allows an efficient allocation and utilization of edge resources. By predicting the future demand of IoT applications, edge resources can be dynamically allocated to meet the specific QoS requirements of each application [18]. Accordingly, traditional ML approaches used to play a major role in QoS prediction for their simplicity and interpretability. For instance, the authors in [4] predicted the delay of IoT applications in a 802.15.4 access network using a Deep Neural Network (DNN) with forward and backward passes. Similarly, the authors in [5] proposed the use of several regression approaches to predict the throughput of six IoT applications in a 802.15.4 network. Additionally, the authors in [6] predicted the throughput of audio, video and sensor data of an IoT healthcare application using an ARIMA/GARCH model. For the throughput prediction, a Convolutional Neural Network (CNN) with a target vectorization technique was also used in [7].

In terms of both single and multi-step ahead prediction, the authors in [8] predicted the delay of a simulated dataset of an IoT environment using a nonlinear autoregressive exogenous (NARX) Recurent Neural Network (RNN). A different type of RNN called Echo State Network (ESN) was also employed in [9], for QoS forecasting at the edge of an IoT network. The authors also introduced random noise into the internal states of the network in order to provide more robust forecasts, while addressing the stability problem of ESNs.

Regarding QoS prediction at the Multi-Access Edge Computing (MEC), Liu et al. [10] provided a multi QoS prediction framework using contextual factors. Firstly, they introduced a workload prediction mechanism for future scheduling services using Support Vector Machine (SVM) and optimized it with an Artificial Bee Colony (ABC) algorithm. Secondly, they performed the multi QoS prediction using Case Based Reasoning (CBR), which is a problem-solving methodology that involves using past experiences (cases) to solve new problems. This work was further extended in [11] providing the QoS prediction for both real-time and future MEC services, using CBR and an optimized SVM. In contrast, the authors in [12] introduced a matrix factorization method that predicts unknown QoS values using a location based user cluster's information and user's reputation information.

So far, all of the above proposed methods are centralized, while the data privacy is not considered. A privacy-aware QoS forecasting model based on Long Short-Term Memory (LSTM) and attention called Edge-PMAM (Edge QoS forecasting with Public Model and Attention Mechanism), was proposed in [13]. This work consisted of public and private models for privacy aware and personalized QoS forecasting. The Edge regions were divided using the Miller projection with k means clustering and for each region the model consisted of an attention layer on top of a LSTM to improve the performance of both models. Furthermore, Zhang et al. [14] designed a QoS prediction model based on FL, by firstly identifying the untrustworthy users and then processing the unreliable data to predict the QoS in a MEC setting. Similarly, the authors in [19] proposed a Federated Hierarchical Clustering for Distributed QoS Prediction (FHC-DQP) in which the user extracted features are utilized for clustering them together. Following, a location aware neural network is used to forecast the QoS. Moreover, Li et al. [20] proposed a personalized QoS prediction using federated tensor factorization, employing tensors to represent QoS data and creating unique personalized models for each edge server. A summary of the related works reviewed in this section is also given in Table I.

However, the above mentioned studies have different limitations that can be summarized as follows:

TABLE I
COMPARISON OF RELATED WORKS

Category	Ref.	Technique	Predicted QoS
Centralized Methods	[4]	Deep Neural Network	Delay
	[5]	Regression methods	Throughput
	[6]	ARIMA/GARCH	Throughput
	[7]	CNN	Throughput
	[8]	NARX-Recurrent Network	Delay
	[9]	Echo State Network	RT & Throughput
	[10]	SVM+ABC+CBR	Response Time
	[12]	Matrix Factorization	Response Time
Decentralized Methods	[13]	LSTM+Attention	RT & Throughput
	[14]	Matrix Factorization	RT & Throughput
	[19]	Hierarchical Clustering	RT & Throughput
	[20]	Tensor Factorization	RT & Throughput

- The existing studies based on traditional ML approaches in [4]- [7] and [19] do not consider the temporal aspect of the prediction. In contrast, these studies employ static models that treat each input sample independently, while not considering the sequential or time-dependent nature of the data. Furthermore, they only forecast one of the two most typical QoS metrics i.e., throughput or delay.
- The RNN models used in [8], [9] and [13] can only handle the short term sequence prediction efficiently. Specifically, during back-propagation, where gradients are propagated from the output to the input, the gradients can diminish or vanish as they are repeatedly multiplied by weight matrices and cause the vanishing gradient problem. As a result, the network struggles to update the weights effectively, particularly for earlier time steps, limiting its ability to capture long-term dependencies.
- The CBR approach used in [10] and [11] is difficult to be applied in the context of IoT, where a vast amount of data is generated from various heterogeneous devices. This could make building a comprehensive case base challenging, while the selection of relevant cases from the case base is difficult in large and heterogeneous datasets.
- The work in [12], [14] and [20] used matrix or tensor factorization techniques, which heavily depend on the data sparsity at the current time slots. Furthermore, the proposed approach in [12] assumed that the user clusters and user reputations remain fixed over time. However, network conditions and user behavior can be dynamic, which may require the model to be updated or retrained. Moreover, the work in [14] used reputation mechanism which relies on assigning reputation scores to clients based on their trustworthiness. However, this process can be subjective and prone to biases.

Herein, we solve the above mentioned challenges as follows: (i) Firstly, we provide the detailed forecasting of four QoS metrics such as throughput, PDR, PLR, and latency by considering the temporal aspects in both univariate and multivariate settings; (ii) Secondly, to overcome the vanishing gradient problem in the training of long QoS data sequences, we are introducing a temporal transformer with sparse attention. The particular architecture can efficiently model long sequences by capturing dependencies between different positions without suffering from the vanishing gradient problem. Additionally, the sparse attention mechanism further optimizes the computational efficiency of our approach, particularly for resource-constrained IoT environments. (iii) Thirdly, we employ a FL approach that trains the client models on diverse IoT devices and sensors data, resulting in a more comprehensive QoS forecasting. Furthermore, our FL model continuously updates the model using distributed data from IoT devices ensuring that the model adapts to changing network conditions and application dynamics. Finally, the FL aggregates model updates from multiple clients providing a more objective and collective perspective on client behavior and trustworthiness.

III. System Model and Problem Statement

A. System Model

We consider a set A of IoT applications such that, $A = \{a_1, a_2, ..., a_k\}$. For each application $a_k \in A$, there is a number of n static sensor nodes denoted by the set $S^k = \{s_1^k, s_2^k, ..., s_n^k\}$ that generate data following a data distribution D^k, with $S^1 \cup S^2 \cup ... \cup S^k = S$. For each application, the dataset is generated for a specific duration T. Each dataset can be represented by a sequence of data points $x_{n,t}^k$ that describe the data generated by sensor n, belonging to application k at time step $t \in T$. To lighten the notation, the k superscript that denotes the application which a sensor node n is associated with will be dropped in the rest of the paper.

Similarly, we can represent the set of access points (APs), that form the edge computing environment, as a set of mobile robots $R = \{r_1, r_2, ..., r_m\}$. Each mobile robot $r_m \in R$ has a set of coordinates $G_m = \{(x_{m,1}, y_{m,1}), (x_{m,2}, y_{m,2}), ..., (x_{m,T}, y_{m,T})\}$ that represent its movement trajectory over a period of time.

The sensors belonging to an application $a_k \in A$, send their data to a unique robot r_m for all time steps T, which is represented as a binary function $z : A \times R \times T \rightarrow \{0, 1\}$. The function $z(a_k, r_m, t) = 1$ if application a_k sends data to robot r_m at timestamp t, and $z(a_k, r_m, t) = 0$ otherwise.

B. Modeling of Network Uncertainties

One of the contributions of this work, is to find an appropriate prediction model that will be able to accurately estimate major QoS metrics, under dynamic network conditions. Thus, four different network dynamics/uncertainties are considered:

1) **Interference:** At each time slot that a pair of sensor nodes $s_n, s_{n'} \in S$ utilizes the same frequency channel, an interference level $I_{n,n'} = f(P_n, d_{n,n'}, \epsilon)$ is created, where P_n represents the transmission power of sensor node s_n, $d_{n,n'}$ is the distance between sensors s_n and $s_{n'}$ and ϵ represents the characteristics of the wireless channel. In this work, we have tested two channel allocation techniques: i) all applications can use the same frequency channel (inter and intra-application interference) or ii) each application is associated with a different frequency channel (intra-application interference).
2) **Transmission power:** Each sensor $s_n \in S$ can be set with a different transmission power P_n to find a balance between transmission range and interference. Herein, the transmission power can be either 0 or $12dBm$.

3) **Mobility:** The robots acting as the mobile access points can also create another level of communication uncertainty. To evaluate the impact of mobility in the QoS prediction, two mobility settings are available: i) static robots (i.e. $G_m = \emptyset$) and ii) mobile robots ($G_m \neq \emptyset$).
4) **Heterogeneity of data:** Finally, the QoS estimation can be affected by the way the data are generated (i.e., event-based, periodic, or hybrid). Accordingly, during the generation of the datasets all the above three data generation distribution have been used.

Based on the above described complexities, there are two interference models I, two power models P and two mobility patters G. Hence, κ network configurations are examined, with $|\kappa| = |I| \times |P| \times |G| = 2 \times 2 \times 2 = 8$. For all these network configurations, the applications follow a heterogeneous data generation with the three distributions mentioned above.

C. Problem Formulation of QoS Forecasting

In our proposed cross-device FL setting, the set $C = \{c_1, c_2, ..., c_l\}$ represents the number of clients, which are connected to a global server $\mathcal{GS}$. Furthermore, the set $Q^\kappa = \{q_1^t, q_2^t, q_3^t, q_4^t\}^T$ represents the QoS metrics for all network configurations κ, where q_1^t is the throughput, q_2^t is the PDR, q_3^t is the PLR and q_4^t is the latency metric at timestamp t for the different IoT applications. Then each client $c_l \in C$ has its own local QoS dataset, which can be represented as $D_l \subseteq Q^\kappa$.

Specifically, D_l is a multivariate QoS dataset, such as $D_l = \{(q_1^1, q_2^1, q_3^1, q_4^1), (q_1^2, q_2^2, q_3^2, q_4^2), ..., (q_i^t, q_i^t, q_i^t, q_i^t)\}$ where each tuple $(q_1^t, q_2^t, q_3^t, q_4^t)$ represents a single observation in the dataset at time t. For a univariate setting (i.e., only throughput), the QoS dataset is represented as $D_l = \{q_1^1, q_1^2, ..., q_1^t\}$ and similarly for the other metrics.

For each QoS dataset (either univariate or multivariate time series), a fixed length input sequence of window size τ such that $\tau \in \mathbb{N}$, is used to predict the next values. To generalize, for any time series X the input sequence with a window size τ can be represented as $\{(x_{t+i_1}, x_{t+i_2}, ..., x_{t+i_\tau}) \mid 1 \leq i_1 \leq T-\tau+1, 1 \leq i_2 \leq T-\tau+2, ..., 1 \leq i_\tau \leq T-\tau+\tau\}$. Given this input sequence, we aim at performing a multi-step forecasting of Δ QoS values in the future, i.e., $\tilde{X} = \{\tilde{x}_1^t, \tilde{x}_2^{t+1}, ..., \tilde{x}_{\Delta-1}^{T-1}, \tilde{x}_\Delta^T\}$. If $\Delta = 1$ it becomes a one-step ahead problem. Hence, the forecasting is performed using a learning model $\mathcal{M}$, such as $\mathcal{M} : X \rightarrow \tilde{X}$ which is parameterized by some weights $\mathcal{W}$. Thus, the goal is to minimize a loss function, between the real X and predicted $\tilde{X}$ values. Hence, in the FL context, each client c_l will iteratively use its local training dataset D_l and training model $\mathcal{M}_l$ in order to minimize the loss function, such as:

$$\arg\min_{\mathcal{W}_l^k} \mathcal{L}_l^k(\mathcal{M}_l^k, D_l^k, \mathcal{W}_l^k) \tag{1}$$

where $\mathcal{L}_l^k(.)$ represents the loss function at round/iteration k, with $K = \{1, 2, 3, ..., k | k \in \mathbb{N}\}$, with the $\mathcal{M}_l^k$, D_l^k, and $\mathcal{W}_l^k$ as its parameters. Each local weight $\mathcal{W}_l^k$ is then transmitted to the server for updating the server's global weight parameter $\mathcal{W}_k^*$ at iteration k, which is called the global update. Then, the learning problem tries to find an optimal model parameter vector $\mathcal{W}_k^*$ and the goal is to minimize the global loss function $\mathcal{L}^*$ over the number of iterations k using a given dataset.

TABLE II
IoT APPLICATIONS PARAMETERS FOR DATA GENERATION

Application	#Nodes	Generation type	Lambda	Period(s)
VoIP	50	Periodic	0	0.0635
Surveillance	25	Exponential	196.74	1
Emergency Response	25	Hybrid	0.0333	30

IV. Use Case and Dataset Generation

In order to create the QoS datasets, we implemented an IoT/Edge Computing use case with 100 static transmitting sensor nodes that offload their data. Each sensor node belongs to one of the three available IoT applications: i) VoIP, that emulates an automatic help desk virtual assistant, ii) Surveillance, which includes a set of cameras for security, and iii) Emergency Response, which monitors critical areas of a building. The three applications produce data according to different distribution modes as shown in Table Table II. Specifically, the periodic mode generates data every i^{th} time instance, the event-based mode follows an exponential law with an occurrence rate of λ, and the hybrid mode which is a combination of both modes. Furthermore, each application is associated with a unique mobile robot that receives this data while moving according to a configured path.

The particular use case was implemented in the FIT IoT-LAB [21], an open testbed that allows large-scale experiments and provides openly programmable IoT nodes located on several sites. Certainly, the generation of datasets through a testbed may have limitations in capturing unforeseen events of real-life scenarios. Additionally, there are not currently available similar datasets from other testbeds or real IoT network locations, to allow us to make direct comparisons. Nevertheless, we selected the FIT IoT-LAB due to its provision of diverse deployment settings for dedicated IoT sensors. This enables us to produce a large volume of IoT data within a controlled environment, essential for machine learning training. Finally, as will be explained below, the selection of the access network, the IoT devices, and the mobile robots create a close to reality experimentation setting.

In particular, we used the IoT-LAB M3 as sensor nodes because they offer a diverse range of sensors and actuators to simulate various IoT scenarios. Additionally, their low-power consumption aligns with the energy constraints common in IoT devices, providing an accurate representation of real-world conditions. Additionally, we employed Turtlebots as mobile access points due to their versatility and ease of integration into the testbed, since they offer the necessary mobility customization for our experiment. All nodes communicate with each other through the 802.15.4 wireless protocol, which is widely used in IoT applications. This protocol provides low-power, short-range wireless communication capabilities, making it well-suited for energy-efficient and resource-constrained devices in IoT deployments. The locations of each sensor node in the experimental setup are shown in Fig. 1.

It is to be noted that other access protocols could be used, either IoT-specific, such as LoRaWAN, NB-IoT, etc. or more traditional ones such as Wi-Fi, cellular, etc. The choice of 802.15.4 was made due to its acceptance as the de

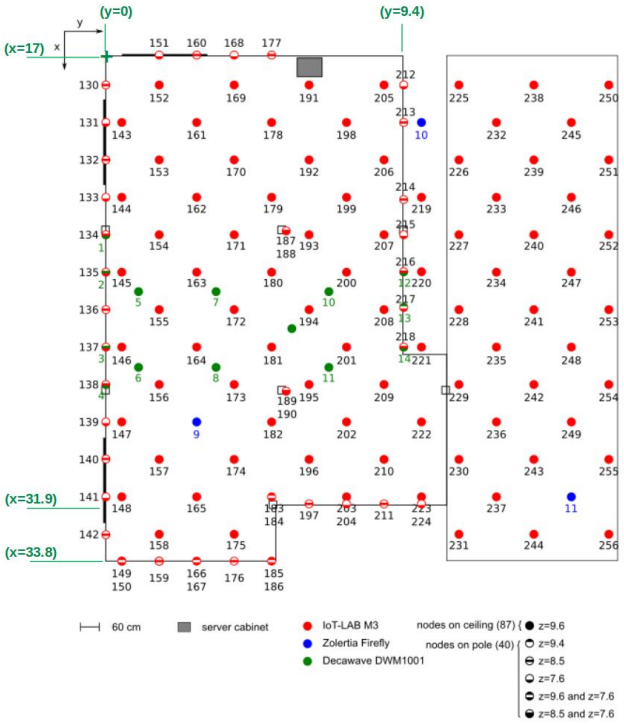


Fig. 1. Sensor nodes location deployed in FIT IoT-LAB

TABLE III
NETWORK CONFIGURATIONS WITH DIFFERENT COMPLEXITIES

Configurations	Mobility	tx_power	Channel	Channel value
1	Mobile	0dBm	Different	{11,16,21}
2	Mobile	12dBm	Different	{11,16,21}
3	Mobile	0dBm	Same	{11}
4	Mobile	12dBm	Same	{11}
5	Static	0dBm	Different	{11,16,21}
6	Static	12dBm	Different	{11,16,21}
7	Static	0dBm	Same	{11}
8	Static	12dBm	Same	{11}

facto short to medium range protocol for IoT applications, especially for those considered in this paper. In different settings necessitating longer transmission ranges LoRaWAN could be considered as a viable alternative.

As stipulated by the selected access protocol (i.e., 802.15.4), the payload size was set to $127B$ for all applications in all configurations. For the latter, and as explained in Section III-B there are eight network configurations, as shown in Table III.

We collected all the data transmitted by the sensor nodes and the data received by the robots. Specifically, at the sending side we collected the timestamp at which a message is transmitted; the name of the sensor node that transmits the message; the id of the transmitted message; the transmission power; the frequency channel; the receiving robot; whether the message was transmitted successfully or not; and the x and y coordinates of the robot. At the receiving side, the following features were collected: timestamp; the name of the receiving robot; the message id; the name of the transmitting sensor; the delay; the channel; and the robot coordinates.

Following, a feature engineer process was followed that resulted into the final QoS time series datasets used for our experimentation. Specifically, the features engineered using the initial raw data are: timestamp; robot name; total transmitted messages at specific timestamp; total received messages at the specific timestamp; the time when the first message is transmitted; the time when the last message is transmitted; PLR; Delay; PDR; and Throughput.

V. PROPOSED MODEL

A. Overview of FeD-TST

To forecast the QoS metrics of varying network uncertainties, we present FeD-TST, which combines the Federated Learning with Temporal Sparse Transformer (TST). FeD-TST is a secure distributed system for both univariate and multivariate Time Series Forecasting (TSF) as shown in Fig.2.

In this framework, there is a central server and multiple client nodes. The clients will first train their models locally based on the TST and then share their model weights with the other clients via the central server. For each connected client and for each network configuration, the hidden layer weights and parameters of the client model (i.e., TST) are represented as ω. After the uploading of this information, the global aggregation is performed at the FeD-TST server and the global model parameters are sent back to each client. Following, each client loads these global weights to its TST model's hidden layers.

The TST model of each local client is illustrated in Fig. 3. In the proposed TST, we first generate the real time IoT data of the three different applications and eight different network configurations as discussed in Section IV. Following, these datasets are pre-processed using normalization, down sampling and cleaning procedures and are prepared as univariate or multivariate inputs. The third step is to divide the dataset into training, validation and testing. The training involves several components, such as the encoder module (Fig. 3a), the decoder module (Fig. 3b), the multihead sparse attention module (Fig. 3c) and a sparse attention module (Fig. 3d). Both the encoder and decoder modules consist of multiple identical sub-layers that are stacked to each other. Finally, uni/multivariate forecasts are produced along with the evaluation based on the test dataset. In the rest of this section, we provide the details of each of the component of our proposed model.

B. Sparse Scaled Dot-Product Attention

The traditional transformer model utilizes the self-attention which is a form of a global attention to model the long-term dependencies [22]. However, global attention attends all the positions in the input QoS sequence leading to attend irrelevant or noisy information that might not be beneficial for the QoS forecasting task. In this work, the sparse scaled dot-product attention is employed to overcome this shortcoming by introducing sparsity, which allows each position to attend only to a subset of positions within a certain input QoS sequence. In this way, it reduces the computational complexity from quadratic to linear with respect to the QoS sequence length and

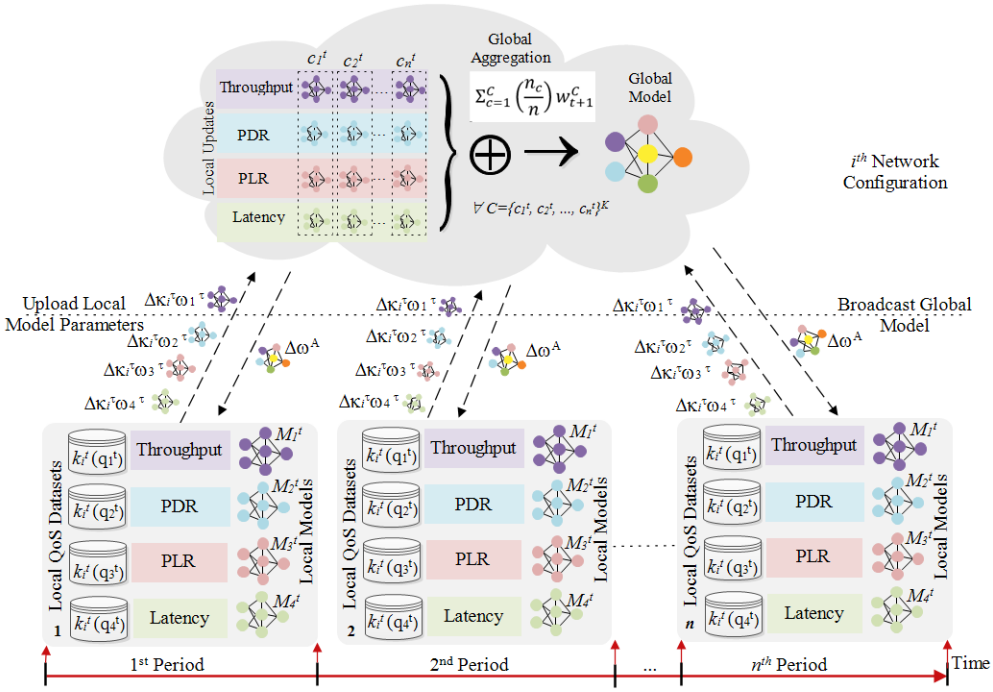


Fig. 2. Federated learning driven IoT QoS Forecasting System

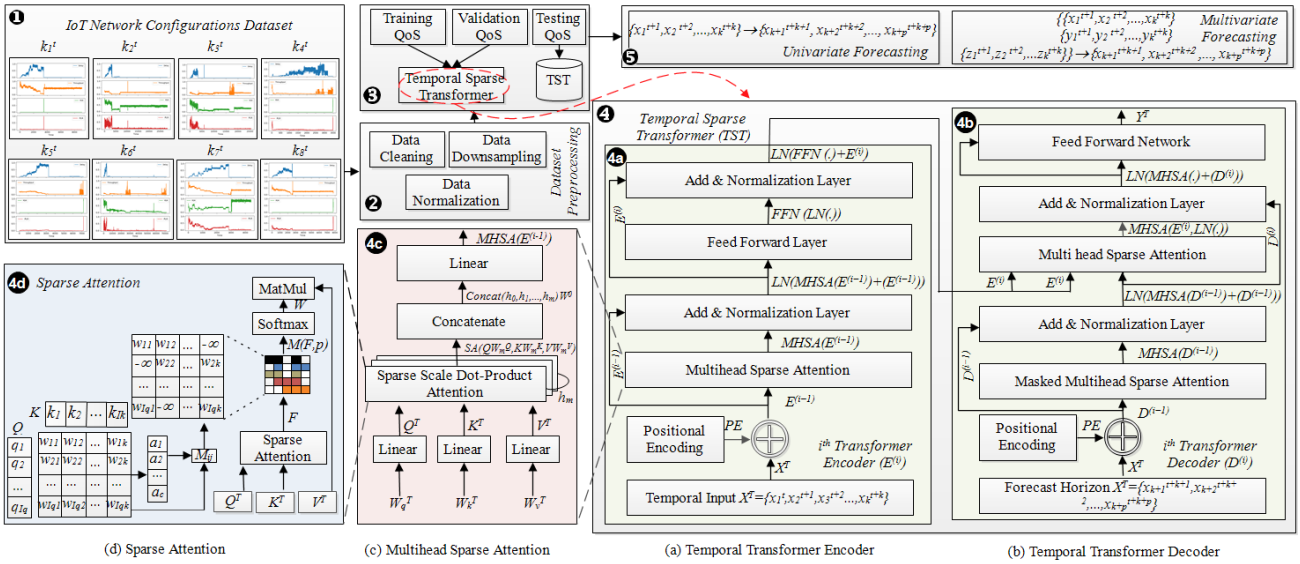


Fig. 3. Proposed Temporal Sparse Transformer (TST) Model for IoT QoS Forecasting

enables more efficient memory usage. This makes our model more suitable for QoS forecasting of real time IoT applications compared to traditional attentions.

As shown in Fig. 3d, a sparse scaled dot-product attention based on top-p selection is used to reserve the most important QoS segments. The attention mechanism consists of three main parameters named query, key and value, which consist the input of the sparse attention. Specifically, the query, $\mathcal{Q} \in \mathbb{R}^{m \times d_q}$ is a query vector of a given input sequence of a certain QoS metric at a specific time step to retrieve information from the key-value pairs. Each query vector is a transformed version of the corresponding time step in the QoS input sequence and is computed using a linear transformation. The key, $\mathcal{K} \in \mathbb{R}^{n \times d_k}$ is a vector for each time step in the QoS sequence to determine the relevance of each element in the sequence with respect to the query. Finally, the value, $\mathcal{V} \in \mathbb{R}^{n \times d_v}$ is a vector containing information associated with each time step in the QoS sequence to compute the output of attention mechanism. The n denotes the length of the key-value pairs, m is the length of the query vector, d is the dimension of the corresponding QoS vector, and $d_q = d_k$. To be more specific, $\mathcal{Q}$, $\mathcal{K}$ and $\mathcal{V}$ are the linear mappings of the input QoS sequence, such as $\{x_1^1, x_2^2, x_3^3, ..., x_i^t\}$, so that, $\mathcal{Q} = W_{\mathcal{Q}} x_i^t$, $\mathcal{K} = W_{\mathcal{K}} x_i^t$, $\mathcal{V} = W_{\mathcal{V}} x_i^t$, where $W_{\mathcal{Q}}$, $W_{\mathcal{K}}$ and $W_{\mathcal{V}}$

denote the learned weight matrices. To generate the attention scores, $\mathcal{F}$, the dot product of $\mathcal{Q}$ and $\mathcal{K}$ is performed which is then divided by $\sqrt{d_k}$ as follows:

$$\mathcal{F} = \mathcal{Q}\mathcal{K}^T/\sqrt{d_k} \tag{2}$$

The attention scores computed using the above equation represent the relation between different QoS values e.g., x_1^1 and x_2^2 in the input sequence. For example, the larger the attention score values are, the higher the relevance of the QoS values will be at a specific time step. After computing the attention scores, the sparse attention performs a masking operation $M(.)$ on the scores to select the top-p contributing values as follows:

$$M(\mathcal{F}, p)_{ij} = \begin{cases} \mathcal{F}_{ij} & if \mathcal{F}_{ij} \geq \theta_i \\ -\infty & otherwise \end{cases} \tag{3}$$

The θ_i represents the p^{th} largest element of each row i of the score matrix $\mathcal{F}$. After obtaining the highest attention values through the top-p selection, the softmax operation is applied to normalize the scores as:

$$SA\_scores = SoftMax(M(\mathcal{F}, p)) \tag{4}$$

where SA_scores denotes the normalized sparse attention values and $M(.)$ is assigned a negative infinity value if the attention scores are less than the threshold value.

C. Temporal Sparse Transformer

1) Input and output of the TST Model: The QoS values are both the input and output of the TST, since past QoS values (input) will be used to predict the future ones (output). Additionally, some contextual features are added only to the input to help with the prediction of the future QoS values. Specifically, during the training of the TST model, the input to the encoder module consists of n contextual features, such as $\{F_1, F_2, ..F_n\}$, along with the QoS values, both forming a set of X^T input samples. Similarly, the input samples for the decoder module consists of the same contextual features with however, the actual (target) QoS values for a defined forecast horizon. In contrast, the output of the decoder consists of the predicted QoS values.

If it is a univariate forecasting setting, then the input sample consists of only one contextual feature i.e., timestamp and a forecasting feature (i.e., throughput), thus, $n = 2$. In case of multivariate forecasting, the QoS metric (i.e., throughput) will be forecasted based on the previous time steps of all contextual features, namely the timestamp (F_1), total transmitted messages (F_2), total received messages (F_3), time first message transmitted (F_4), time last message transmitted (F_5), PDR (F_6), PLR (F_7), latency (F_8) and throughput (F_9) itself, and thus, $n > 2$. However, the final forecasting output generated by the TST model will be the throughput values for the specific forecasting horizon. The same approach will be applied for the other three QoS metrics, such as, PDR, PLR and latency.

The contextual features along with the QoS values form the time series vectors, and they need to be converted into a suitable format to be used by the TST model. For this, the sliding window technique is used where the time series vectors are divided into smaller windows of fixed length, also called input samples, and each window is passed as an input to the TST model. Related to these input samples/windows, there are two important parameters i) the rolling window size, which is the length of the sliding window that determines the number of time series data values in each window and ii) the forecast horizon, which is the number of future steps to be predicted.

2) QoS Temporal Positional Embedding: The TST model encodes the temporal position information to extract the long-term temporal dependencies from the time series QoS input sequences. The traditional attention mechanism can also capture the relations among the QoS input sequence [23], however, it neglects the ordering information in such time-series data. Therefore, in order to make use of the order of the input sequence, there is a positional encoding added in the TST model to add the location information to each input sequence value. The positional encoding can be described as follows:

$$PE_{pos,2i} = \sin\left(\frac{pos}{\alpha^{2i/d}}\right) \tag{5}$$

$$PE_{pos,2i+1} = \cos\left(\frac{pos}{\alpha^{2i/d}}\right) \tag{6}$$

where pos is the positional index of the QoS value in the input sequence, i is the length of the QoS input sequence, α is a user defined scalar and d is the dimensionality of the encoded position in the input sequence.

3) Long-Term Temporal QoS Extraction: To solve the long-term QoS forecasting problem, we propose a TST network to extract the long-term temporal relations from the entire QoS input sequence. The TST model consists of ρ number of QoS TST encoders and the corresponding ρ number of QoS TST decoders, and its architecture is shown in Fig. 3a and Fig. 3b. Each QoS forecasting encoder consists of the multi-head attention, fully-connected feed forward network, dropout [24] and layer normalisation [25] components. To express the i^{th} TST encoder, the abstract form is given as follows:

$$E^{(i)} = TST(E^{(i-1)}) = \mathcal{LN}(\mathcal{FFN}(\tilde{E}^{(i)}) + \tilde{E}^{(i)}) \tag{7}$$

where $E^{(i)}$ is the output of the i^{th} TST encoder, $\mathcal{LN}(.)$ is the operation of normalization layer, $\mathcal{FFN}(.)$ represents the fully connected feed-forward network and it can be expanded into:

$$\mathcal{FFN}(\tilde{E}^{(i)}) = max\left(0, \tilde{E}^{(i)}W^1 + b^1\right)W^2 + b^2 \tag{8}$$

where $\mathcal{FFN}(.)$ contains two fully-connected layers with dropout and a ReLU activation with W^1, b^1, W^2 and b^2 as its corresponding learnable parameters. The $\mathcal{FFN}(.)$ thus gives the nonlinear transformation of its input. Additionally, $\tilde{E}^{(i)}$ denotes the intermediate feature of the QoS encoder and can be represented as follows:

$$\tilde{E}^{(i)} = \mathcal{LN}(MHSA(E^{(i-1)}) + E^{(i-1)}) \tag{9}$$

where $MHSA(.)$ is the multi-head sparse attention, which uses m different linear transformations and analyzes the previous QoS encoder layer features i.e., $E^{(i-1)}$. The multi-head sparse attention process can be written as:

$$MHSA(E^{(i-1)}) = Concat\left(h_0, h_1, h_2, ..., h_m\right)W^0 \tag{10}$$

where $Concat$ is the concatenation operation performed on all attention heads, W^0 is the linear transformation of the concatenated output, and h_m is the m^{th} attention head:

$$h_m = \mathcal{SA}\left(QW_m^Q, KW_m^K, VW_m^V\right) \tag{11}$$

where $\mathcal{SA}$ is the sparse attention as discussed in section V-A and is computed using Equations 2-4.

The QoS decoder of the TST has the same structure as the QoS encoder, however, two additional operations are added. Firstly, a subsequent mask to the first attention block is added, which ensures that the forecasting of the position i in the input sequence can only rely on the known outputs of positions which are less than i, thus avoiding the auto-regressive behavior. This addition is given as:

$$\tilde{D}^{(i)} = \mathcal{LN}(M\tilde{H}SA(D^{(i-1)}) + D^{(i-1)}) \tag{12}$$

where $M\tilde{H}SA$ is the masked multi-head attention for the decoder and $D^{(i-1)}$ is the output from the previous decoder.

Secondly, an attention block is added to the decoder that performs the attention operation on the output of the encoder block. Based on these discussed additions, the QoS decoder is given as:

$$\begin{aligned} D^{(i)} &= \mathcal{LN}(MHSA(\tilde{D}^{(i)}, E^{(i)}) + \tilde{D}^{(i)}) \\ \hat{D}^{(i)} &= \mathcal{LN}(FFN(D^{(i)}) + D^{(i)}) \end{aligned} \tag{13}$$

where $E^{(i)}$ is the output from the i^{th} encoder and $\hat{D}^{(i)}$ is the output from the i^{th} decoder.

D. Proposed FeD-TST

The key idea of FeD-TST scheme is that the clients co-train the TST model presented above, while keeping the QoS data locally for each network configuration. To be more specific, FeD-TST, as shown in Fig. 2 consists of the following steps:

1) Initialization: The secure client and server connection is first established, along with the server's selection of a subset of clients from all connected clients to participate in the training for the following round/iteration. After all clients are enumerated, the model parameters are initialized and broadcast to each client. These parameters include the number of encoder and decoder blocks ρ, the number of communication rounds $\mathcal{K}$, the number of local epochs $\mathcal{E}$, the learning rate η, the loss function $\mathcal{L}$, the initial weight w^0, which is the starting point of the global model's parameters before the training process begins, and the time step size s.

2) Training of the local TST models on Clients: After the initialization step, each local client trains a TST model using their respective local QoS data for the k^{th} round, and the training results will be used for the next $k+1$ round. The same process is repeated at each round.

3) Uploading of the local TST model weights: When the training of each client's local TST model is completed, each client extracts the respective TST weights and then uploads them to the server for the next round $k+1$.

4) Aggregation of the model weights: According to the received TST model weights from each client, the server then performs the aggregation mechanism, called Federated Averaging (FeDAvg), which aggregates the local model updates from the participating clients and computes a global model update. FeDAvg actually combines stochastic gradient descent (SGD) of each client (local gradients) with a server that performs global model averaging (aggregation). The aggregated results are then broadcast to each client. This iterative process is repeated until the loss function converges or a maximum number of rounds is reached. The algorithmic implementation of all the components presented in this Section along with their complexity analysis is provided in the Appendix A.

VI. Experimental Evaluation

In this section, the evaluation of our proposed FeD-TST framework is provided. First, a brief overview of the network configurations is provided and their respective datasets. Following, the baseline comparative methods and the performance metrics are outlined . Lastly, the results of all experiments are provided along with their the comparative analysis.

A. Model Implementation and Frameworks

1) Network Configurations and Datasets: Under the time series forecasting settings, we forecast the four following QoS metrics as: (i) Throughput; (ii) PDR; (iii) PLR and (iv) Latency in both univariate and multivariate settings. The window size for both settings is set to be 30. For the network configurations with mobility as a parameter, the duration of the corresponding QoS datasets is 6 hours stipulated by the battery duration of the robots. In contrast, for static access points, the duration of the respective QoS datasets is 10 hours. Finally, it is noted that we use 60% of the data for training, 20% of the data for validation and the remaining 20% for testing purposes.

2) Baseline Methods: For comparison purposes, we evaluate our proposed FeD-TST model against the most popular centralized and decentralized learning models that are appropriate for time series forecasting. Regarding the centralized baselines we have used the i) LSTM, ii) Bidirectional LSTM (Bi-LSTM), iii) Attention + BiLSTM, iv) Sequence to Sequence (Seq2Seq), v) Seq2Seq + Attention [26], vi) Temporal Convolutional Networks (TCN) + BiLSTM [27], and vii) Time2Vec + Transformer [28]. Finally, for the decentralized comparison we have employed the i) FL + LSTM, and the ii) FL + Multi-head Attention (MHA), which is a temporal transformer model [16] adapted to the FL setting. The description of all these approaches along with the hyperparameters used are provided in the Appendix B.

3) Performance Metrics: In order to evaluate the performance of each method, three widely used metrics have been selected: Mean Absolute Error (MAE), Mean Square Error (MSE), and Root Mean Square Error (RMSE). Additionally, to evaluate the effectiveness of the proposed approach, we use the communication cost model as proposed in [29]. The particular cost is modeled as $2 \cdot \mathcal{K} \cdot \mathcal{N} \cdot \Omega \cdot \mathcal{M}_{size}$ where $\mathcal{K}$ is the total number of communication rounds, $|\mathcal{LC}|$ is the total number of client nodes, Ω is the fraction of client nodes to be

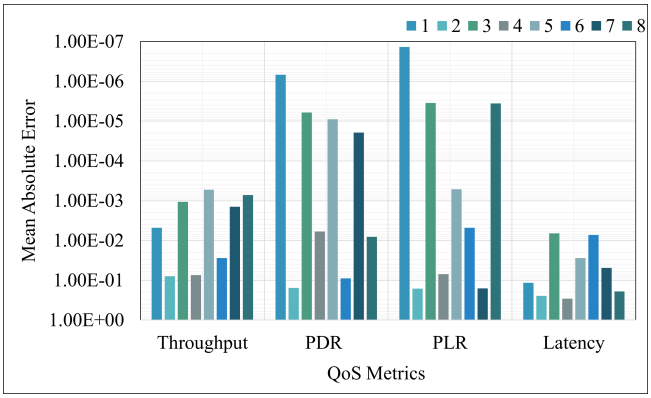


Fig. 4. Network configuration impact on QoS forecasting

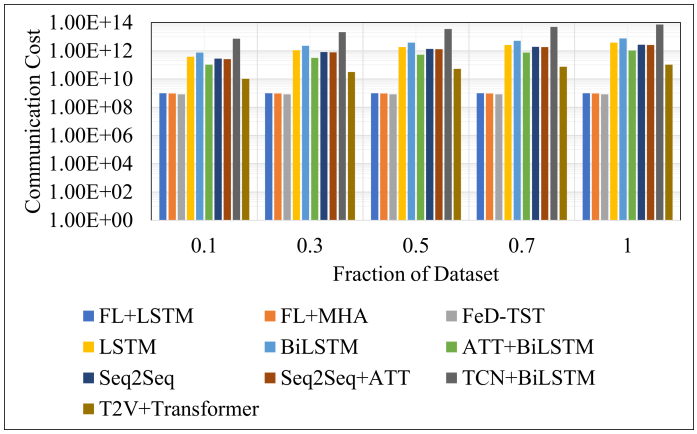


Fig. 5. Communication Cost

selected, $\mathcal{M}_{size}$ is the size of the ML model. The latter is equal to the total number of trainable parameters $P_{\mathcal{M}}$ multiplied by the size of the parameters in bits Υ i.e., (4, 8, 16, 32 bits).

4) Implementation Details: We implemented the proposed FeD-TST model and all baseline federated approaches using the Flower Federated Framework [30] with 512 local clients for four QoS datasets under eight network configurations, respectively. For each of the QoS forecasting, we trained every algorithm for 30 communication rounds. All the centralized and decentralized models were executed in a Python environment with open-source TensorFlow libraries. All models were trained on high-performance Linux clusters offered by Compute Canada namely, Cedar and Beluga. For the Beluga cluster, we trained, validated and tested the models on a NVIDIA V100 with 16GB GPU and for the Cedar cluster, we utilized the NVIDIA P100 with 16GB GPU respectively.

5) Results: For both of the time series forecasting settings, we present the accuracy efficiency of all four different QoS metrics, while considering the eight different network configurations. In particular, the results will be assessed with respect to the impact of network configurations, the communication cost, and the forecasting efficiency of the FL model.

B. Impact of Network Configurations

To evaluate the impact of the network conditions on the prediction accuracy, we first plot the MAE when using our proposed FeD-TST solution under all eight network configurations. As shown in Fig. 4, the forecasting error for the throughput is the least for the fifth network configuration (i.e., *Static_diff_channel_0dbm*) followed by network configuration 8 (i.e., *Static_same_channel_0dbm*). This is because the robot access points remained stationary, which reduced the variability in the network conditions leading to more stable transmissions. Thus, the deep learning model can learn more easily such scenarios.

Next for the PDR and PLR, the FeD-TST provides better forecasting for the first network configuration (i.e., *Mobile_diff_channel_0dbm*). The reason is that PDR and PLR can be severely affected by the level of the interference. Thus, when using different frequency channels, the likelihood of interference between the applications is reduced, leading to more reliable transmissions and improved forecast accuracy. In contrast, when the interference becomes more important the PDR and PLR change more significantly, affecting the overall accuracy of the forecast. The same observation is drawn when the transmission power increases, which also leads to higher interference and the forecasting accuracy decreases for such configurations.

Lastly for the latency, FeD-TST achieves the best forecasting for the third network configuration (i.e., *Mobile_same_channel_0dbm*). In the particular configuration, we expect to see an increased level of interference, which leads to more retransmissions and thus longer delays. However, the proposed model managed to learn such complex behaviors providing good forecasting accuracy.

C. Communication Cost

In terms of communications cost, Fig. 5 provides the comparison of the proposed FeD-TST with the rest of the benchmarks, when varying the fraction of the dataset used by the training process, from 10 to 100%. It should be noted, that the particular Figure illustrates the communication of only one network configuration (i.e., *Mobile_same_channel_0dbm*), due to space limitations. Nonetheless, similar results were obtained for the rest of the configurations as well.

The key observation of the communication cost, is that for all federated learning based methods, such as *FL+LSTM*, *FL+MHA* and our proposed *FeD-TST*, the cost remains constant for all different fractions of the dataset used for the training. This is attributed to the fact that there is no actual datasets transferred between the clients and server. Instead, only the updates of the global and local agents are exchanged, resulting in a consistent and constant communication cost. However, the communication costs of the centralized models increase with the fraction of the dataset used in the training process. Among the centralized models, TCN+BiLSTM shows the highest communication cost, as it has the highest number of trainable parameters compared to the other models. Finally, the proposed FeD-TST gives the least communication cost of $8.2056E+08$, compared to the other FL approaches, as FL+MHA yields a communication cost of $9.43992E+08$ and FL+LSTM of $9.81624E+08$.

TABLE IV
UNIVARIATE FORECASTING RESULTS FOR THROUGHPUT, BEST RESULTS ARE HIGHLIGHTED IN BOLD

Configurations		1	2	3	4	5	6	7	8
Methods	Metrics	Mobile	Mobile	Mobile	Mobile	Static	Static	Static	Static
LSTM	MSE	0.002180008	0.058064374	0.017221582	0.019661525	0.007008886	0.000043146	0.028044095	0.007652375
	MAE	0.031486913	0.199266738	0.082996892	0.073941417	0.075111779	0.001767353	0.130296992	0.053803794
	RMSE	0.046690557	0.240965503	0.131231025	0.140219554	0.08371909	0.006568581	0.167463712	0.087477852
BiLSTM	MSE	0.002149963	0.058196797	0.017138673	0.008286225	0.006855734	0.000059553	0.027234415	0.007399871
	MAE	0.03109576	0.201700344	0.08100321	0.043326636	0.073171018	0.001759178	0.12784306	0.051513432
	RMSE	0.046367696	0.241240122	0.130914755	0.091028704	0.08279936	0.007717055	0.165028528	0.086022505
ATT+BiLSTM	MSE	0.002113405	0.06045324	0.016387171	0.008071625	0.007019641	0.000026737	0.028635743	0.007840934
	MAE	0.03286028	0.203475403	0.078651295	0.043766762	0.074738003	0.001766987	0.132077915	0.055272665
	RMSE	0.045971786	0.245872406	0.128012387	0.089842222	0.083783299	0.005170763	0.169220988	0.088549048
Seq2Seq	MSE	0.002172816	0.058766967	0.017565562	0.007910387	0.006828733	0.000036975	0.028332603	0.007925499
	MAE	0.034427306	0.202370203	0.082538805	0.045117619	0.074294335	0.001800946	0.131756	0.053090038
	RMSE	0.046613471	0.24241899	0.132535134	0.088940356	0.082636146	0.006080671	0.168322913	0.089025274
Seq2Seq+ATT	MSE	0.002108766	0.057875243	0.017857116	0.007336685	0.007259331	**0.000011513**	0.027428818	0.007885314
	MAE	0.031982373	0.201074426	0.094457217	0.042848431	0.072002338	**0.00172142**	0.125179741	0.054124594
	RMSE	0.045921299	0.24057274	0.13363052	0.085654454	0.085201709	**0.003393085**	0.165616478	0.08879929
TCN+BiLSTM	MSE	0.00217524	0.05769644	0.016973583	0.007007257	0.007166868	0.000020477	0.02818864	0.008333666
	MAE	0.035093832	0.198256349	0.081892429	0.045072616	0.076047148	0.001763019	0.133661025	0.062342796
	RMSE	0.046639469	0.240200832	0.130282705	0.083709361	0.084657358	0.004525111	0.16789473	0.091288916
T2V+Transformer	MSE	0.002094785	0.059257085	0.016875335	0.006954281	0.006820699	0.000048149	0.03231086	0.008300878
	MAE	0.032487425	0.204504672	0.080115092	0.0402296	0.074963033	0.001738727	0.139963191	0.053888586
	RMSE	0.045768823	0.243427782	0.129905099	0.083392333	0.082587524	0.006938978	0.179752218	0.091109152
FL+LSTM	MSE	0.026109913	0.198509365	0.019059159	0.009396685	0.007740845	0.000105776	0.036035966	0.008500859
	MAE	0.121700913	0.245523065	0.083601423	0.048500054	0.073999718	0.004875254	0.153384194	0.05503843
	RMSE	0.161543608	0.444212198	0.13799575	0.096909925	0.087978683	0.010284722	0.189783856	0.092178144
FL+MHA	MSE	0.002752693	0.093995392	0.029476583	0.013653285	0.015197324	0.000150662	0.063143753	0.035793204
	MAE	0.038101345	0.248628452	0.130680636	0.061146908	0.101738915	0.007402018	0.198087156	0.147313178
	RMSE	0.052466109	0.306586683	0.171687454	0.116847269	0.123277426	0.01227443	0.251284212	0.189190924
FeD-TST	MSE	**3.67E-05**	**0.007006747**	**0.000226781**	**0.006597863**	**3.48511E-05**	0.001039662	**0.003347858**	**0.001060865**
	MAE	**0.004760905**	**0.079123348**	**0.001064445**	**0.040155964**	**0.000532004**	0.027471544	**0.001423224**	**0.000721199**
	RMSE	**0.006055055**	**0.083706312**	**0.001505924**	**0.081227229**	**0.005090348**	0.032243785	**0.018297152**	**0.001002998**

D. Univariate Results

In Table IV, the three error metrics i.e., MSE, MAE and RMSE, when forecasting the throughput for all methods and network configurations, are presented. From these results, the following observations can be drawn. Firstly, the proposed FeD-TST model outperforms the other two distributed solutions (i.e., FL+LSTM and FL+MSA) in almost all network configurations. There are two main reasons for this efficiency: i) FL+LSTM can suffer from the vanishing gradients when processing long sequences, which can make it difficult to learn long-term dependencies. However, FeD-TST can avoid this problem by using the attention module, which is used to capture the dependencies between different throughput values in the input sequence. This allows the FeD-TST model to learn which parts of the input sequence are more relevant for the throughput forecasting at each time step. By propagating this information across the entire sequence using the attention module, FeD-TST avoids the problem of vanishing gradients. Furthermore, the attention module in FeD-TST is more effective than the gating mechanism in FL+LSTM in terms of capturing long-term dependencies and patterns in the data, especially for longer throughput input sequences. Moreover, FeD-TST uses the positional encoding mechanism which encodes the position of each throughput value in the input sequence. This empowers the FeD-TST to differentiate between various throughput observations in the sequence, even if they have the same value and better handle the longer throughput sequences; ii) FL+MSA uses a self-attention module, which computes the importance of each throughput value in the input sequence based on its relationship with all the other throughput values in the sequence. This means that the self-attention considers all throughput values in the sequence when computing the attention weights. Thus, this can lead the self-attention to be affected by noisy or irrelevant throughput values in the sequence, even if they are not relevant for the forecasting. In contrast, the sparse attention in FeD-TST only considers a subset of the throughput values in the input sequence when computing the attention weights. This subset of values is typically chosen based on their relevance for the forecasting, which means that noisy or irrelevant QoS values in the sequence may be ignored by the model making FeD-TST more robust than FL-MSA.

The second observation from Table IV is that FeD-TST also outperforms all centralized solutions and for almost all network configurations. The reason is that the FeD-TST enables the model to learn from multiple clients with potentially different data distributions. This helps our model to better adapt to new data and be more robust to changes in the data distribution over time. Furthermore, FeD-TST can leverage the collective knowledge of multiple clients at the same time, in order to improve its performance on throughput forecasting.

The final observation, is that for network configuration 6, the Seq2Seq+ATT model performed slightly better than FeD-TST. This can attributed to the fact that the dataset for configuration 6 may not have been diverse enough in terms of the throughput observations. This can adversely impact the learning, since FL models are trained on data that are distributed across multiple clients, which can help improve generalization. However, if

TABLE V
UNIVARIATE FORECASTING RESULTS FOR PDR, BEST RESULTS ARE HIGHLIGHTED IN BOLD

Configurations		1	2	3	4	5	6	7	8
Methods	Metrics	Mobile	Mobile	Mobile	Mobile	Static	Static	Static	Static
LSTM	MSE	0.018257154	0.037851512	0.000608792	0.001952939	0.001112278	0.01857656	0.076177719	0.001061075
	MAE	0.080767698	0.156077519	0.001516378	0.003028177	0.001499788	0.080090836	0.245758619	0.003359817
	RMSE	0.135119038	0.194554657	0.024673704	0.044192069	0.033350831	0.136295855	0.276003114	0.032574153
BiLSTM	MSE	0.018449913	0.025352443	0.001411486	0.001915232	0.000599673	0.019455916	0.074454278	0.002206909
	MAE	0.080295092	0.12848674	0.001740071	0.003789981	0.001141312	0.07880042	0.241632533	0.008116467
	RMSE	0.135830455	0.159224505	0.037569752	0.04376336	0.024488231	0.139484464	0.272863113	0.046977751
ATT+BiLSTM	MSE	0.018410313	0.024726183	0.00088561	0.001182298	0.000512866	0.018651737	0.076006	0.000577173
	MAE	0.080084426	0.125214189	0.001705488	0.003541321	0.001272777	0.079920708	0.245711249	0.002313198
	RMSE	0.135684608	0.157245613	0.029759198	0.03438456	0.02264655	0.136571364	0.275691856	0.024024419
Seq2Seq	MSE	0.018686683	0.02546426	0.000627586	0.002250746	0.000353211	0.019352935	0.074024439	0.000891181
	MAE	0.084082524	0.128587491	0.001882692	0.004034477	0.002321665	0.080620867	0.247234381	0.002561207
	RMSE	0.136699243	0.159575247	0.025051661	0.047442024	0.018793907	0.139114827	0.272074326	0.029852654
Seq2Seq+ATT	MSE	0.017895584	0.02488008	0.000994254	0.002043545	0.000256814	**0.018469424**	0.072522882	0.000677193
	MAE	0.079086621	0.125329378	0.001816939	0.004525415	0.001136588	**0.078296446**	0.235733602	0.002153011
	RMSE	0.133774376	0.157734206	0.031531789	0.045205586	0.016025408	**0.13590226**	0.269300728	0.026022931
TCN+BiLSTM	MSE	0.018551141	0.024655612	0.000600939	0.001797786	0.000350983	0.019236453	0.074489966	0.001268126
	MAE	0.082181067	0.125796429	0.00392316	0.003801887	0.001281348	0.081557265	0.2375388	0.007112032
	RMSE	0.136202572	0.157021055	0.024514057	0.04240031	0.01873454	0.13869554	0.272928499	0.035610752
T2V+Transformer	MSE	0.018813977	0.02562528	0.000637369	0.001975316	0.016922135	0.018765976	0.074159335	0.001057309
	MAE	0.086577267	0.130660705	0.001173224	0.005953995	0.055162054	0.079134315	0.243209761	0.006711487
	RMSE	0.137164053	0.16007898	0.025246166	0.044444524	0.130085105	0.136988963	0.272322117	0.032516286
FL+LSTM	MSE	0.019515449	0.098753192	0.000603801	0.135606289	0.001285934	0.025663415	0.107867897	0.000866407
	MAE	0.086151034	0.269703567	0.001251696	0.366929799	0.001705134	0.085382722	0.23624748	0.008104443
	RMSE	0.139641225	0.314214915	0.024572313	0.368222237	0.035859846	0.160190925	0.32837072	0.029434759
FL+MSA	MSE	0.02645552	0.048118532	0.006903093	0.022831427	0.003011749	0.020869514	0.095742323	0.001933012
	MAE	0.122321516	0.17536521	0.071824536	0.118473649	0.008171082	0.091781527	0.254539758	0.009358659
	RMSE	0.162651524	0.219359368	0.083084859	0.151100725	0.054879408	0.144462854	0.309422553	0.043966036
FeD-TST	MSE	**07.52E-13**	**0.024042883**	**4.86E-11**	**0.000736637**	**8.88E-11**	0.023981718	**5.03E-10**	**0.000481912**
	MAE	**6.82E-07**	**0.123524003**	**6.07E-06**	**0.002928413**	**9.01E-06**	0.089999422	**1.95E-05**	**0.002050021**
	RMSE	**8.67E-07**	**0.155057675**	**6.97E-06**	**0.027141048**	**9.42E-06**	0.154860318	**2.24E-05**	**0.021952493**

the data distributions across the clients are too similar, the FL model may not be able to effectively capture the underlying patterns in the data. Finally, the Seq2Seq+ATT model is a data-efficient model and it can achieve good performance even with fewer homogeneous training samples.

Similarly, Table V presents the error metric results for PDR. As it can be seen, the proposed FeD-TST model performs better than all other models for all configurations except for the network configuration 6 (i.e., static robots, different channel allocation, and $12dBm$ transmission power). Once more and for the reasons described above the Seq2Seq+ATT model exhibited the best performance. This means that the Seq2Seq+ATT model is able to well capture the temporal patterns that affect the PDR dataset, such as changes in network congestion or interference, and make more accurate future forecasts. Furthermore, for other seven network configurations, FeD-TST is better than all models as LSTMs with its variants struggled with capturing long-term dependencies efficiently, Seq2Seq and its variant models struggled handling complex and diverse PDR patterns. Regarding T2V+Transformer models they help to encode time-related information using Time2Vector representations, but they may not be as effective as the FeD-TST in capturing the nuanced temporal patterns for these seven network configurations.

Regarding PLR, and as seen in Table VI, FeD-TST provides the best results for 6 configurations. Specifically, for the second (i.e., mobile robots, different channel, $12dbm$) and fourth network configuration (i.e., mobile robots, same channel, $12dbm$), TCN+BiLSTM showcased the best performance. In both configurations the access points are mobile and the transmission power is high. Nonetheless, from the same Table and for the first (mobile robots, different channel, $0dbm$) and third configuration (mobile robots, same channel, $0dbm$), it can be seen that FeD-TST performed better. By looking close into these configurations, we can see that the transmission power may have an impact on how the learning models perform.

Specifically, when a network configuration contains mobile access points the network topology and data distribution can change over time. This means that the data may not be as sparse, as all access points may have relevant information at different times. Also the sparsity of PLR datasets can be influenced by the transmission power. When the transmission power is set to 0dBm, the signals do not travel far, resulting in a shorter propagation range, and higher PLR. Since the FeD-TST model can handle large amounts of data more efficiently, the model may have more information to work with, when there is a noticeable amount of PLR, allowing it to attend the subset of important PLR values using the sparse attention. Therefore, it can perform well on this type of PLR datasets. However, when the transmission power is set to 12dBm, the signals can travel further, resulting in a more sparse PLR dataset. Hence, TCN+BiLSTM, which uses convolutional layers to extract the features from the PLR input data, can capture well local patterns, while the BiLSTM layer can effectively capture the longer-term temporal relationships between the PLR values. Overall, TCN+BiLSTM performs better than the FeD-TST model in scenarios where the dataset is more sparse.

TABLE VI
UNIVARIATE FORECASTING RESULTS FOR PLR, BEST RESULTS ARE HIGHLIGHTED IN BOLD

Configurations		1	2	3	4	5	6	7	8
Methods	Metrics	Mobile	Mobile	Mobile	Mobile	Static	Static	Static	Static
LSTM	MSE	0.021792671	0.033600801	0.000170506	0.003800143	0.000898948	0.006792275	0.028396553	0.004032325
	MAE	0.082623192	0.150432311	0.007338254	0.023558733	0.013402833	0.030336409	0.13075646	0.032339007
	RMSE	0.14762341	0.183305213	0.013057785	0.061645296	0.029982454	0.082415258	0.168512768	0.063500594
BiLSTM	MSE	0.020697191	0.034122424	0.000401793	0.004283569	0.000542852	0.006855659	0.041219532	0.003004725
	MAE	0.080600034	0.151636813	0.008385715	0.024967082	0.012687792	0.027902375	0.158798501	0.031972674
	RMSE	0.143865183	0.184722559	0.020044774	0.06544898	0.023299188	0.08279891	0.203025937	0.054815371
ATT+BiLSTM	MSE	0.020245053	0.033294998	0.00014359	0.00421118	0.000547313	0.006935762	0.028205546	0.003443814
	MAE	0.080818832	0.151242344	0.007237762	0.023958626	0.013155007	0.028010578	0.132771995	0.032751093
	RMSE	0.142285113	0.182469171	0.011982922	0.064893606	0.023394728	0.083281221	0.167945069	0.058684019
Seq2Seq	MSE	0.021252736	0.033916195	0.000523421	0.003230895	0.000885887	0.00682344	0.027612433	0.002580039
	MAE	0.086687473	0.153022372	0.007574055	0.022760354	0.013292484	0.028000094	0.131795246	0.032047241
	RMSE	0.145783182	0.184163502	0.022878392	0.056840967	0.029763853	0.082604119	0.166169893	0.050794084
Seq2Seq+ATT	MSE	0.020900074	0.032672588	0.000460761	0.003551169	0.000549879	0.006830279	0.028939158	0.00355437
	MAE	0.085837735	0.148762972	0.009517111	0.022397076	0.012634716	0.032682136	0.131375186	0.032834303
	RMSE	0.14456858	0.180755604	0.021465339	0.059591688	0.023449497	0.0826455	0.170115131	0.059618536
TCN+BiLSTM	MSE	0.020198194	**0.031638767**	0.000394537	**0.002814738**	0.000232801	0.006485215	0.028564665	0.003535622
	MAE	0.078534621	**0.148136144**	0.008401207	**0.022391848**	0.012239431	0.02950272	0.132377563	0.033134242
	RMSE	0.142120352	**0.177872896**	0.019862957	**0.053054102**	0.015257805	0.080530831	0.169010844	0.0594611
T2V+Transformer	MSE	0.020911002	0.034417311	0.000920884	0.004394198	0.000889586	0.006719895	0.030964824	0.002978452
	MAE	0.085307551	0.153888459	0.009341288	0.02465749	0.013932025	0.031952691	0.137978923	0.030472741
	RMSE	0.14460637	0.18551903	0.030346072	0.066288749	0.029825929	0.081974968	0.175968248	0.054575195
FL+LSTM	MSE	0.022094503	0.041293263	0.046052642	0.004675237	0.810819387	0.006800786	69.65638733	0.003036108
	MAE	0.075020067	0.159862995	0.137425244	0.02439647	0.367824584	0.021475384	1.586332083	0.028901355
	RMSE	0.148626059	0.203149363	0.214597806	0.068371393	0.867065966	0.082466155	8.330111504	0.055099569
FL+MSA	MSE	0.026332522	0.053814277	0.002095943	0.009201947	0.144768745	0.008289964	0.042631045	0.006102875
	MAE	0.09593612	0.185807839	0.013791043	0.049586143	0.285889983	0.040396597	0.162858874	0.05424583
	RMSE	0.16227299	0.231979042	0.045781475	0.095926777	0.380484879	0.091049239	0.206472874	0.078120902
FeD-TST	MSE	**2.66E-14**	0.038911376	**1.91E-11**	0.008929118	**3.05E-05**	**3.67E-05**	**0.026158862**	**2.60E-11**
	MAE	**1.37E-07**	0.16107823	**3.48E-06**	0.070162557	**0.00051405**	**0.004760905**	**0.127036972**	**3.57E-06**
	RMSE	**1.63E-07**	0.197259665	**4.37E-06**	0.094494008	**0.005524271**	**0.006055055**	**0.161737015**	**5.10E-06**

TABLE VII
UNIVARIATE FORECASTING RESULTS FOR LATENCY, BEST RESULTS ARE HIGHLIGHTED IN BOLD

Configurations		1	2	3	4	5	6	7	8
Methods	Metrics	Mobile	Mobile	Mobile	Mobile	Static	Static	Static	Static
LSTM	MSE	0.024408949	0.06017804	0.088636216	0.056182637	0.017420559	0.009039901	0.100882114	0.040418583
	MAE	0.080140304	0.214223589	0.248948097	0.201964768	0.049288355	0.033133292	0.291552399	0.167005577
	RMSE	0.156233636	0.245312127	0.297718349	0.237028769	0.131986964	0.095078395	0.317619448	0.201043734
BiLSTM	MSE	0.024206015	0.0603612	0.088730471	0.084598273	0.01688086	0.009230823	0.101561174	0.041677659
	MAE	0.079947131	0.214269641	0.249370197	0.198644096	0.047570767	0.033800875	0.291890608	0.165094546
	RMSE	0.155582824	0.245685164	0.297876604	0.246165022	0.129926363	0.096077172	0.31868664	0.204151069
ATT+BiLSTM	MSE	0.024179881	0.084598273	0.089001176	0.057675521	0.01729584	0.009121923	0.102110922	**0.040280716**
	MAE	0.080108745	0.246165022	0.240791322	0.203930682	0.046483801	0.03072686	0.293887956	**0.14324601**
	RMSE	0.155498813	0.290857822	0.298330648	0.240157283	0.131513648	0.095508757	0.319547997	**0.200700563**
Seq2Seq	MSE	0.024997304	0.05980379	0.089389425	0.055409534	0.016375065	0.008770341	0.099581648	0.040978067
	MAE	0.078904381	0.214706046	0.256589036	0.201288012	0.050110525	0.033967249	0.281536809	0.168885758
	RMSE	0.158105358	0.244548135	0.298980642	0.235392298	0.127965093	0.093650097	0.315565599	0.202430399
Seq2Seq+ATT	MSE	0.024013291	0.60308408	0.086830182	0.057301211	0.017024882	0.008396982	0.103578993	0.041083115
	MAE	0.081703762	0.21346644	0.254700822	0.202772912	0.049581186	0.035931874	0.295579004	0.170103562
	RMSE	0.154962224	0.245577704	0.294669615	0.239376713	0.130479431	0.091635047	0.321836904	0.2026897
TCN+BiLSTM	MSE	0.032147098	0.062922512	0.09267886	0.054981224	0.016755645	0.009074548	0.103784816	0.04146944
	MAE	0.11613974	0.213559675	0.255984481	0.199181873	0.046739142	0.034320761	0.29798527	0.16241555
	RMSE	0.179296121	0.250843602	0.304432029	0.234480753	0.129443598	0.095260424	0.322156508	0.203640468
T2V+Transformer	MSE	0.024686465	0.064907747	0.09201476	0.055858478	0.016280688	0.008851462	0.103110644	0.040447623
	MAE	0.082032614	0.222181234	0.266272543	0.199272798	0.052664447	0.031812765	0.298888429	0.166716834
	RMSE	0.15711927	0.254769988	0.303339347	0.236343982	0.1275958	0.094082209	0.321108461	0.201115943
FL+LSTM	MSE	0.027547928	80687.98438	0.103246123	4.166488647	0.019056881	0.009888737	0.121394642	0.047525864
	MAE	0.077818892	18.25476456	0.230045661	0.455275744	0.053534083	0.036810573	0.274667084	0.170613229
	RMSE	0.165957242	283.6138611	0.32129097	2.016258478	0.138044104	0.099438779	0.348386973	0.217998505
FL+MSA	MSE	0.024906835	0.115047574	0.245384723	0.098320305	0.023086162	0.01547756	0.136887431	0.07021708
	MAE	0.077157654	0.27891928	0.433125585	0.257558107	0.077514783	0.060696498	0.318601936	0.196365908
	RMSE	0.157818988	0.339186639	0.495363235	0.313560694	0.151941314	0.124408841	0.369983017	0.264985055
FeD-TST	MSE	**0.023777915**	**0.059448318**	**4.40E-05**	**0.054488377**	**0.001039662**	**7.52E-05**	**0.002381638**	0.066853426
	MAE	**0.076379195**	**0.212089206**	**0.006626479**	**0.198644096**	**0.027471544**	**0.007280115**	**0.04880077**	0.190088391
	RMSE	**0.15420089**	**0.243820257**	**0.006636661**	**0.233427456**	**0.032243785**	**0.00867211**	**0.048802029**	0.2585603

For the latency dataset and as seen from Table VII, the FeD-TST is well suited for all network configurations except configuration 8 (static robots, same channel, $12dBm$), where the ATT+BiLSTM model gives the best performance. The

TABLE VIII
MULTIVARIATE FORECASTING RESULTS FOR THROUGHPUT, BEST RESULTS ARE HIGHLIGHTED IN BOLD

Configurations		1	2	3	4	5	6	7	8
Methods	Metrics	Mobile	Mobile	Mobile	Mobile	Static	Static	Static	Static
LSTM	MSE	0.028607518	0.098101639	0.013346768	0.06513989	0.027364537	0.009801793	0.019369793	0.009325607
	MAE	0.132310821	0.306853361	0.09260656	0.195485626	0.094224957	0.0679559	0.135783555	0.077761354
	RMSE	0.169137572	0.313211812	0.115528215	0.255225175	0.1654223	0.099004003	0.139175403	0.096569184
BiLSTM	MSE	0.02742663	0.101624222	0.013520551	0.070678015	0.027028323	0.00956266	0.011623326	0.009524192
	MAE	0.129706767	0.312485707	0.093181167	0.211890758	0.085585064	0.065992332	0.103169602	0.0786563
	RMSE	0.165609873	0.318785543	0.116277903	0.26585337	0.16440293	0.097788856	0.107811532	0.097591965
ATT+BiLSTM	MSE	0.029332515	0.037467551	0.012400092	0.0409477	0.027625441	0.00918847	0.02887653	0.008108388
	MAE	0.134291403	0.192225229	0.089390103	0.133535994	0.054213672	0.063080543	0.165644779	0.07195941
	RMSE	0.171267378	0.193565365	0.111355702	0.202355382	0.166209028	0.095856508	0.169930955	0.090046588
Seq2Seq	MSE	0.029094397	0.222091576	0.013109092	0.078138339	0.087997502	0.075132291	0.002492687	0.007884014
	MAE	0.133091282	0.469024161	0.091876588	0.265520788	0.2902785	0.216263738	0.043581509	0.070501437
	RMSE	0.170570797	0.471265929	0.114494943	0.279532357	0.296643729	0.274102701	0.049926819	0.08879197
Seq2Seq+ATT	MSE	0.029802806	0.197385415	0.013370054	0.076762567	0.09746884	0.072463298	**0.001641729**	0.008784528
	MAE	0.134996629	0.440116966	0.092956957	0.250004146	0.307147377	0.210497291	**0.033518686**	0.075263339
	RMSE	0.172634893	0.444280784	0.115628948	0.277060583	0.3122	0.269190078	**0.040518256**	0.093725811
TCN+BiLSTM	MSE	0.028206558	0.049226831	0.013811762	0.05078849	0.032273292	0.011554496	0.004679834	0.008682814
	MAE	0.131378731	0.219623558	0.093851078	0.145438506	0.061250018	0.078781507	0.061909672	0.074983242
	RMSE	0.167948081	0.221871203	0.117523452	0.225363019	0.179647688	0.10749184	0.068409315	0.09318162
FL+LSTM	MSE	0.022230878	0.026630133	0.02434326	0.030587779	0.047000714	0.03989619	0.058205191	0.020379072
	MAE	0.116344981	0.12812157	0.117841065	0.138585195	0.203050449	0.158704802	0.18743296	0.105792277
	RMSE	0.148784027	0.162836522	0.155774087	0.17474249	0.216674656	0.199541911	0.241099924	0.142648384
FL+MSA	MSE	0.055654656	0.219694048	0.123248108	0.834618747	0.290231615	0.079888575	0.062982544	0.022085747
	MAE	0.202963769	0.439056277	0.317067206	0.88365382	0.483849257	0.249765113	0.208570436	0.107883148
	RMSE	0.235912398	0.46871531	0.351067096	0.913574696	0.538731515	0.282645643	0.250963241	0.148612738
FeD-TST	MSE	**2.97E-05**	**0.000250346**	**0.002767468**	**1.40E-05**	**0.002059271**	**0.000360866**	0.003959762	**0.00255931**
	MAE	**0.0043**	**0.012030067**	**0.039089505**	**0.003005067**	**0.041864872**	**0.015221067**	0.050758883	**0.034195092**
	RMSE	**0.0054**	**0.015822316**	**0.052606728**	**0.003742877**	**0.045379192**	**0.018996468**	0.062926643	**0.050589621**

reason is that when multiple robots are operating on the same channel, they can interfere with each other's signals. This becomes more evident when the robots are static and they use high transmission power, increasing this way the common serving area, which however will suffer from more dense interference levels. Inevitably, this high interference introduces additional complexities in capturing the patterns associated with increased latency. Such temporal complexities become intricate as they are influenced by both short-term variations and longer-term trends. Therefore, forecasting the latency in such a network configuration is more challenging for federated learning models.

E. Multivariate Results

Table VIII presents the multivariate results for the throughput prediction for different network configurations. As in the case of univariate prediction, our proposed model, Fed-TST outperformed all of the rest methods and for almost all network configurations. Additionally, the prediction errors are reduced compared to the univariate setting, since in this second set of results, the FeD-TST model does not only capture the temporal dependencies of the forecasting variable e.g., throughput values but the temporal dependencies of all the other features at the same time (i.e., throughput, PDR, PLR, latency, time first packet transmitted, time last packet transmitted, total transmitted packets, total received packets). By modeling the temporal dependencies between these multiple features, the FeD-TST is able to better forecast the future forecasting variable e.g., throughput values. Furthermore, the sparse attention mechanism is more useful in multivariate settings, as these settings consist of multiple features which however, may or may not be relevant to the target forecasting variable. Thus, the sparse attention efficiently attends only relevant features and the FeD-TST model learns a more accurate representation of the input data, while providing more accurate forecasts than the other models.

Only for the seventh network configuration (static robots, same channel, $0dbm$), the Seq2Seq+ATT centralized model performed slightly better than the FeD-TST. By further analyzing this behavior, we found that the size of the input sequence had an impact on the final performance of the FeD-TST. Due to the complexity of this configuration, temporal patterns and dependencies might span across a larger number of data points which necessitates longer input sequences for the model to discern and capture meaningful patterns. However, the input sequence length is not providing enough information for the FeD-TST model to effectively capture the temporal relationships among the multiple features. Thus, an increase in the input sequence length may reduce the forecasting error metrics.

Similar observations were drawn for PDR, PRL, and Latency, where the FeD-TST consistently gave the best performance for the large majority of the network configurations. Nonetheless, for page limitations purposes, we have decided to include the Tables of these multivariate QoS forecasting in the Appendix C. Overall, FeD-TST managed to find the best accuracy performance with the minimum prediction error in 53 out of 64 univariate and multivariate experiments performed, which proves that it can uniformly outperform the rest of the centralized and distributed approaches in a dynamic and uncertain network environment.

F. Limitations of proposed FeD-TST

The sparse attention introduced in the proposed TST model may limit the model's ability to capture more global context information of QoS in scenarios where contextual information of QoS metrics is more important. However, this makes the TST model limited when applied in the centralized setting. In contrast, in this work, a federated learning based training of TST is performed. Thus, the TST model is trained collaboratively on decentralized clients and servers, each with its local QOS dataset. This allows the TST model to learn from diverse data sources, capturing a broader range of patterns and QoS context more accurately than a TST model trained on a single centralized dataset.

VII. Conclusion

In this work, we investigated the QoS forecasting problem by formulating it as a univariate and multivariate time series forecasting problem in a federated learning setting. In particular, a new framework, FeD-TST was introduced that promotes an efficient QoS forecasting for a number of heterogeneous network configurations with various IoT applications that stress the IoT access network creating several levels of QoS uncertainty. To evaluate our framework, we firstly generated real-time datasets for eight different network configurations that considered the mobility of access points, the frequency channel, and the transmission power allocation as configuration knobs. Following, we presented a novel Federated learning based sparse temporal transformer based architecture (FeD-TST), which learns temporal representations and their long term complex dependencies in a federated fashion, for the forecasting of four QoS metrics, namely, throughput, PDR, PLR and latency. Finally, we performed an extensive experimental evaluation in which we proved that our proposed FeD-TST outperforms several competitive benchmark methods in both univariate and multivariate settings. As a future work, we aim to explore alternative attention techniques, such as compressed attention and investigate their impact on the accuracy achieved. Furthermore, we would like to investigate the aggregation strategies other than the FeDAvg at the server side to improve future forecasts of several key QoS metrics.

Acknowledgments

This work was supported in part by the CHIST-ERA-2018-DRUID-NET project "Edge Computing Resource Allocation for Dynamic Networks".

References

[1] F. Saeik *et al.*, "Task offloading in edge and cloud computing: A survey on mathematical, artificial intelligence and control theory solutions," *Computer Networks*, vol. 195, p. 108177, 2021.

[2] D. Hanes, G. Salguiero, P. Grossetete, R. Barton, and J. Henry, *IoT Fundamentals: Networking Technologies, Protocol, and Use Cases for the Internet of Things*. Cisco Press, 2017.

[3] Y. Hahn, T. Langer, R. Meyes, and T. Meisen, "Time series dataset survey for forecasting with deep learning," *Forecasting*, vol. 5, no. 1, pp. 315–335, 2023.

[4] M. Ateeq, F. Ishmanov, M. K. Afzal, and M. Naeem, "Predicting delay in iot using deep learning: A multiparametric approach," *IEEE Access*, vol. 7, pp. 62 022–62 031, 2019.

[5] A. Hameed, J. Violos, N. Santi, A. Leivadeas, and N. Mitton, "A machine learning regression approach for throughput estimation in an iot environment," in *2021 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, 2021, pp. 29–36.

[6] P. Bardalai, H. Neog, P. E. Dutta, N. Medhi, and S. K. Deka, "Throughput prediction in smart healthcare network using machine learning approaches," in *2022 IEEE 19th India Council International Conference (INDICON)*, 2022, pp. 1–6.

[7] Y. Hou *et al.*, "A study of throughput prediction using convolutional neural network over factory environment," in *2021 23rd International Conference on Advanced Communication Technology (ICACT)*, 2021, pp. 429–434.

[8] A. R. Abdellah, O. Abdulkareem Mahmood, and A. Koucheryavy, "Delay prediction in iot using machine learning approach," in *2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2020, pp. 275–279.

[9] G. White and S. Clarke, "Short-term qos forecasting at the edge for reliable service applications," *IEEE Transactions on Services Computing*, vol. 15, no. 2, pp. 1089–1102, 2022.

[10] Z. Liu, Q. Z. Sheng, W. E. Zhang, D. Chu, and X. Xu, "Context-aware multi-qos prediction for services in mobile edge computing," in *2019 IEEE International Conference on Services Computing (SCC)*, 2019, pp. 72–79.

[11] Z. Liu, Q. Z. Sheng, X. Xu, D. Chu, and W. E. Zhang, "Context-aware and adaptive qos prediction for mobile edge computing services," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 400–413, 2022.

[12] S. Li, J. Wen, and X. Wang, "From reputation perspective: A hybrid matrix factorization for qos prediction in location-aware mobile service recommendation system," *Mobile Information Systems*, no. 1574-017X, p. 8950508, 2019.

[13] H. Jin, P. Zhang, H. Dong, Y. Zhu, and A. Bouguettaya, "Privacy-aware forecasting of quality of service in mobile edge computing," *IEEE Transactions on Services Computing*, vol. 16, no. 1, pp. 478–492, 2023.

[14] Y. Zhang, P. Zhang, Y. Luo, and L. Ji, "Towards efficient, credible and privacy-preserving service qos prediction in unreliable mobile edge environments," in *2020 International Symposium on Reliable Distributed Systems (SRDS)*, 2020, pp. 309–318.

[15] D. Bisht and M. Ram, *Recent advances in time series forecasting*. CRC Press, 2021.

[16] A. Hameed, J. Violos, A. Leivadeas, N. Santi, R. Grünblatt, and N. Mitton, "Toward qos prediction based on temporal transformers for iot applications," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4010–4027, 2022.

[17] U. Mangla, *Application of Federated Learning in Telecommunications and Edge Computing*. Springer International Publishing, 2022, pp. 523–534.

[18] K. Papadakis-Vlachopapadopoulos, I. Dimolitsas, D. Dechouniotis, E. E. Tsiropoulou, I. Roussaki, and S. Papavassiliou, "Blockchain-based slice orchestration for enabling cross-slice communication at the network edge," in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2020, pp. 140–147.

[19] G. Zou, S. Lin, S. Hu, S. Duan, Y. Gan, B. Zhang, and Y. Chen, "Fhc-dqp: Federated hierarchical clustering for distributed qos prediction," *IEEE Transactions on Services Computing*, pp. 1–14, 2023.

[20] X. Li, S. Li, Y. Li, Y. Zhou, C. Chen, and Z. Zheng, "A personalized federated tensor factorization framework for distributed iot services qos prediction from heterogeneous data," *IEEE Internet of Things Journal*, vol. 9, no. 24, pp. 25 460–25 473, 2022.

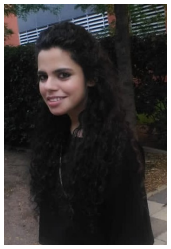
[21] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, "Fit iot-lab: A large scale open experimental iot testbed," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015, pp. 459–464.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[23] S. Reza, M. C. Ferreira, J. Machado, and J. M. R. Tavares, "A multi-head attention-based transformer model for traffic flow forecasting with a comparative analysis to recurrent neural networks," *Expert Systems with Applications*, vol. 202, p. 117275, 2022.

[24] X. Pan, R. Coen-Cagli, and O. Schwartz, "Modeling neural variability in deep networks with dropout," *bioRxiv*, 2021.

[25] F. Liu, X. Ren, Z. Zhang, X. Sun, and Y. Zou, "Rethinking skip connection with layer normalization in transformers and resnets," *ArXiv*, vol. abs/2105.07205, 2021.
[26] G. Li, F. Li, T. Ahmad, J. Liu, T. Li, X. Fang, and Y. Wu, "Performance evaluation of sequence-to-sequence-attention model for short-term multi-step ahead building energy predictions," *Energy*, vol. 259, p. 124915, 2022.
[27] Y. Chen, Y. Kang, Y. Chen, and Z. Wang, "Probabilistic forecasting with temporal convolutional neural network," *Neurocomputing*, vol. 399, pp. 491–501, 2020.
[28] M. Kazemi *et al.*, "Time2vec: Learning a vector representation of time," *arXiv e-prints*, pp. arXiv–1907, 2019.
[29] Q. Xia, W. Ye, Z. Tao, J. Wu, and Q. Li, "A survey of federated learning for edge computing: Research problems and solutions," *High-Confidence Computing*, vol. 1, no. 1, p. 100008, 2021.
[30] D. J. Beutel *et al.*, "Flower: A friendly federated learning research framework," 2022.



Aroosa Hameed is currently an Ericsson Postdoctoral Fellow at Carleton University, Ottawa, Canada. She received her MPhil degree in computer science from Quaid-i-Azam University, Islamabad, Pakistan, in 2018, and the Ph.D. degree from École de technologie Supérieure (ETS), Universite du Quebec, Montreal, Canada, in August 2023. She received the best paper award in the IEEE iThings 2021. Her primary research interests include Internet of Things (IoT), edge computing, machine learning, federated learning, and 5G communications.

John Violos is research associate in the Dept. of Software Engineering and Information Technology at ETS. His previous positions were research associate at National Technical University of Athens, sessional lecturer at Harokopio University of Athens and visiting lecturer at National and Kapodistrian University of Athens. He was a member in the European Commission's Digital Single Market working group on the code of conduct for switching and porting data between cloud service providers. His research interests include Deep Learning, Machine Learning, Cloud and Edge computing.

Nina Santi is a PhD student under the supervision of Nathalie Mitton in the Inria FUN team. Their focus is on small computing devices like electronic tags and sensor networks. She has received the MSc degrees in Computer Science from University of Lille, France, in 2020.

Aris Leivadeas (S'12-M'15-SM'21) is currently an Associate Professor with the Department of Software and Information Technology Engineering at the École de technologie Supérieure (ETS), Montreal, Canada. From 2015 to 2018 he was a postdoctoral fellow in the Department of Systems and Computer Engineering, at Carleton University, Ottawa Canada. In parallel, Aris worked as an intern at Ericsson and collaborated with Cisco in Ottawa, Canada. He received his diploma in Electrical and Computer Engineering from the University of Patras in 2008, the M.Sc. degree in Engineering from King's College London in 2009, and the Ph.D degree in Electrical and Computer Engineering from the National Technical University of Athens in 2015. His research interests include Network Function Virtualization, Cloud and Edge Computing, IoT, and network optimization and management. He received the best paper award in ACM ICPE'18 and '23 and IEEE iThings'21 and the best presentation award in IEEE HPSR'20.

Nathalie Mitton received the MSc and PhD. degrees in Computer Science from INSA Lyon in 2003 and 2006 respectively. She received her Habilitation à diriger des recherches (HDR) in 2011 from Université Lille 1. She is currently an Inria full researcher since 2006 and from 2012, she is the scientific head of the Inria FUN team which is focused on small computing devices like electronic tags and sensor networks. Her research interests focus on self-organization from PHY to routing for wireless constrained networks. She has published her research in more than 30 international revues and more than 100 international conferences. She is involved in the setup of the FIT IoT LAB platform (http://fit-equipex.fr/, https://www.iot-lab.info), the H2020 CyberSANE and VESSEDIA projects and in several program and organization committees such as Infocom 2021 & 2020 & 2019, PerCom 2020 & 2019, DCOSS 2021 & 2020 & 2019, Adhocnow (since 2015), ICC (since 2015), Globecom (since 2017), VTC (since 2016), etc. She also supervises several PhD students and engineers.