A Forecasting Approach to Improve Control and Management for 5G Networks

Diogo Ferreira, André Reis, Carlos Senna, Susana Sargento

Abstract—**In 5G networks, time-series data will be omnipresent for the monitoring and management of network performance metrics. With the increase in the number of Internet of Things (IoT) devices, it is expected that the number of real-time time-series data streams will increase at a fast pace, making forecasting essential for the proactive successful management of the network.**

In this paper, we discuss to use both linear and non-linear forecasting methods, including machine learning, deep learning, and neural networks to improve 5G networks' management. For this purpose, we design and implement a real-time distributed forecasting framework, used to make simultaneous predictions of different network performance metrics, and with different learning algorithms. By using our framework, we compare the use of forecasting methods in two network scenarios, in a real vehicular network and in a 4G network, representing two different slices in a 5G network. We also integrate our framework in a 5G architecture.

Using the best forecasting models assessed previously, we propose a dynamic threshold algorithm for multi-slice management, to ensure that the resources of each slice are updated according to the slices' needs, while avoiding congestion and saving resources for other slices. The experimental results show that it is possible to forecast the slices' needs and congestion probability, selecting the best forecasting approach or an ensemble of the best ones, and act accordingly in the network to optimize its management.

Index Terms—**5G Management, Forecasting, Machine Learning, Neural Networks, NFV, SDN, Slicing Optimization.**

I. Introduction

The effort made in recent years by the research community and the Telecom industry in defining a new network architecture in the 5th generation of communications that supports the new set of requirements is finally reaching the market [1]. Due to the networks' dynamic load and flexible topology, forecasting of the network state is a must to ensure that the user requirements are met. Automation of the network management is also mandatory, here made only possible thanks to advances in virtualization, mainly in Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) [2].

An accurate and autonomous traffic forecasting is essential for many traffic management decisions in a network, such as traffic accounting, short-time traffic scheduling, long-term capacity planning, network design or network anomaly detection, and even multi-slice management in 5G networks [3][4]. With the scale and complexity expected for 5G networks, where flexible networks and their services are instantiated on demand, there is no time to react to an anomaly on the network, a congestion situation, and other impairments. It is essential to predict the required resources, their anomalies, and prevent them from occurring or changing the network proactively. Forecasting methods are an important tool to predict the needs of dynamic networks, and provide optimization in a multi-slice approach.

Forecasting methods can be divided into linear and non-linear. Linear forecasting methods, such as Autoregressive Integrated Moving Average (ARIMA) and Holt-Winters algorithms, have been used by network operators to forecast network KPIs. However, with the recent advances in Machine Learning, mainly in Deep Learning [5], studies show that non-linear forecasting methods, such as neural networks, produce more accurate forecasts than linear forecasting methods [6], [7]. These approaches have already been used, very recently, in the new 5G network concepts of different verticals through multiple slices [8][9]; however, they have not considered an autonomous management approach.

In this work, we discuss the use of linear and non-linear forecasting algorithms to improve multi-slice management in 5G networks, with varying hyper-parameters. For this purpose, we design and implement a real-time distributed forecasting framework, used to make simultaneous real-time predictions of different performance metrics and through different algorithms, including an ensemble of the best forecasting algorithms.

We consider two vertical scenarios with real data as multiple slices in a 5G network: a real vehicular network [10], and a 4G mobile operator network. The best forecast achieved in both scenarios is used to study the autonomic multi-slicing, considering two different verticals with their 5G slice, by dynamically forecasting their needs, dynamically allocate resources for the slices, providing both multiplexing and independence between slices, and preventing network congestion. The results show that it is possible to forecast the congestion and act accordingly in the network to improve its performance through a dynamic and autonomous resource management. Using these scenarios, we demonstrated that the prediction of network metrics, besides helping the network administrator to understand the future trends of the network, can also be used to perform autonomic improvements in the network. This approach has been integrated in a 5G architecture.

The novel contributions of this article are the following:

- Extensive evaluation of prediction approaches for two different networks with real data: a vehicular network and a 4G operator network;
- Extensive search and test of hyper-parameters in machine learning and deep learning approaches;

D. Ferreira and S. Sargento are with DETI, University of Aveiro, Portugal ({diogodanielsoaresferreira, susana}@ua.pt).

C. Senna, A. Reis and S. Sargento are with Instituto de Telecomunicações, Aveiro, Portugal ({cr.senna, andrebragareis}@av.it.pt).

- Feature-based testing of a broad number of feature combinations;
- Real-time distributed forecasting framework, with simultaneous real-time predictions of different performance metrics, through different algorithms, including an ensemble of the best forecasting algorithms;
- Dynamic threshold algorithm that takes advantage of the forecasts of the next time slot to adapt the network resources in an multi-slice 5G management;
- Integration of the Predictor framework into a 5G architecture, consuming KPIs from the Monitor/Assurance component, evaluating these KPIs alone or in an aggregated way, to alert the management components such as Orchestrator and Policer, about unexpected behaviour. Thus, we show how our framework can support proactive congestion management in 5G slices.

The remainder of the article is organized as follows. Section II addresses the relevant related work. Section III presents the forecasting algorithms, while Section IV presents and discusses the results. Section V presents the prediction framework, and section VI depicts the integration of our framework in a 5G architecture. Finally, section VII presents the conclusions and the future work.

II. Related Work

Traffic forecasting, along with classification, were two of the earliest machine learning applications in the networking field. Forecasting is a sub-discipline of prediction in which the predictions are about the future, on the basis of time-series data. Depending on the data available and the sampling period (the time between samples), it may be adequate to use a different forecasting type. For example, if the data samples have a sampling period of a day, it may be useful to use middle-term or long-term forecasts. If the data has a sampling period of a few seconds and it is only available from a period of a month, it is more adequate to perform short-term or real-time forecasts [11]. In this work we use forecasts with a lead time of one hour (short-term forecasts).

The ARIMA model is a stationary stochastic process, and contains two polynomials: an autoregression polynomial and a moving average polynomial. The Holt-Winters algorithm is a forecasting technique from the family of Exponential Smoothing methods [12]. Non-linear forecasting methods usually involve an Artificial Neural Network (ANNs). An ANN is a machine learning model based on the human brain. The ability to learn non-linear functions in an efficient and stable manner makes ANNs one of the most used techniques in machine learning today. The most common type of ANNs are Feed-Forward Neural Networks [6], where the information flow is unidirectional and there are no feedback loops (Section III-B).

The work in [13] compared the forecasts of the transfer rate in a network link, through different types of ANNs and statistical prediction time-series algorithms. The ANNs that were applied used multi-task and multiresolution learning. To create multiple representations of the training data, the Haar wavelet [14] was applied to the network traffic, to perform multi-resolution decomposition. The results of the work showed that, again, non-linear traffic forecasting approaches based on ANN outperformed linear forecasting models.

An Echo State Network (ECN) [15] is used as the ANN model. An ECN is a recurrent neural network with three layers: the input layer, the reservoir layer and the output layer. In this study, five mobile network services were chosen to forecast the traffic volume in a city: MMS, web, streaming media, QQ (the most popular instant messaging application in China) and XunLei (the most popular P2P application in China).

In [16], a hidden Markov model based on algorithms such as Kernel Bayes Rule (KBR) and Recurrent Neural Network (RNN) with Long Short Term Memory unit (LSTM unit) is used to train the model and to apply the model to predict the future traffic. The metric that is collected and used to forecast future network traffic is the flow count. In the study, the KBR performed better than the forecast using recurrent neural networks. The paper showed promising results in the use of flow-level statistics to forecast the traffic volume.

Network traffic classification helps to manage network resources, such as bandwidth requirement and fault diagnosis. Machine learning traffic classification techniques can be interesting options, by recognizing statistical features (attributes) of the traffic (such as packet size, inter-packet arrival times and packet lengths), and clustering network traffic in clusters that have similar traffic patterns. In [17] the authors propose unsupervised K-means and Expectation Maximization algorithm to cluster the network traffic application based on similarity between them. However, they do not discuss the implementation of the proposed algorithms. In [18], the proposed approach takes advantage of the network traffic decomposition that can be done, in linear and non-linear components. That decomposition is done using a Discrete Wavelet Transform. After that, the linear component is forecasted using the ARIMA model, and the non-linear component is forecasted using a Recurrent Neural Network (RNN), more adequate to non-linear data. The forecasts of both components are then averaged to achieve the final forecasts. The proposed technique outperforms the forecasts of ARIMA and RNN individually.

RNN lies in the deep learning group. Deep learning appears to be a viable approach for the network operators to configure and manage their networks in a more intelligent and autonomous fashion [6]. Applications of Deep Learning in network traffic control are relatively recent and garnered minor attention [19]. In [20], the work focuses on reducing the training time of Long Short-Term Memory Neural Network (LSTM). Besides being the best approach for network traffic forecasting, it takes excessive and computational resources to train. The results showed that, even only 35% of neural connectivity shows a satisfactory performance in the traffic forecasting, while being much easier to train.

In [21], the authors introduce an automated Network Resource Allocator (NRA) system. It is engined by Machine Learning algorithms to predict traffic demands, identify a topology to deliver incoming traffic according to an SLA and operational costs. The NRA takes measurements from GStreamer players and network probes, and predicts network KPIs such as path bandwidth and latency directly related to QoE. The work in [22] presents and compares different machine learning models for the analysis of cellular network traffic, addressing two different problems: detection of anomalies generated by smartphone apps and prediction of Quality of Experience (QoE) for popular apps. The work considers an extensive battery of machine learning models, including single models as well as machine learning ensembles such as bagging, boosting and stacking, and are evaluated using real cellular traffic measurements captured at operational networks and at the end devices. It is an interesting work but is specific for cellular traffic, while the work we proposed in this article is applicable to any type of network.

Considering multi-slice resource management, recently there has been a strong focus on this subject in the literature. The article in [4] discusses the potential and critical role of artificial intelligence for the management and operation of mobile networks that implement network slicing. This work describes practical deep learning architectures that can solve multi-slices' resource problems in different case studies, and illustrates the high typical gain that can be expected from integrating AI in network slicing. It concludes that AI has a clear potential to become a cardinal technology for future-generation zero-touch mobile networks. The work in [23] describes a network slicing management framework based on complex network theory, which is referred to be appropriate to massive slices environment for 5G UDN. This work considers the management of massive slices and infrastructure resource domains, control of resources competition and attack prevention. However, this work is not implemented nor tested, so that more insights can be retrieved from such approach.

Examples of multi-slice and end-to-end network slicing algorithms are very recent and are presented in [24], [25], [26], [27], [9], and [8]. The works in [24] and [9] use reinforcement learning to assess the resources of the multiple slices. The proposed algorithms are tested with simulated results in [24] and with application-based traces available in the Internet in [9]. The first algorithm is compared with base BS coverage algorithms, considering RAN-based approaches. The algorithms in [25] and [8] are based on Deep Neural Networks to forecast the resources in each slice. The work in [25] tests the algorithm with an aggregation of hourly traffics of OTTs for a period of 5 days and compares with LSTM, while [8] considers simulated traffic and compares the results with other deep learning approaches. Finally, the work in [26] proposes a multi-resource allocation framework based on the Ordered Weighed Average (OWA) operator, and tests it with a subset of Amazon EC2 instances available in the Internet; and [27] proposes an optimization model based on Markov decision processes, with tests provided with simulated data. These algorithms have a strong complexity and cannot be implemented in real-time. Moreover, none of the previous works are able to forecast simultaneously different algorithms and metrics in a distributed approach, considering online real traffic from different real networks, and integrated in a 5G architecture approach.

Given the results obtained in the previous works, in this work several forecasting methods will be tested in different vertical slices in a 5G environment. Compared to previous works, we assess the performance of different forecasting approaches, using ARIMA as the benchmark, but we consider a multi-slice 5G network scenario with both vehicular and cellular networks. Moreover, we propose a framework to test different approaches and performance metrics simultaneously, including the ensemble of the best approaches, to provide a real-time monitoring and management of the 5G network. Finally, we propose dynamic and autonomous approaches for the resource management of the multiple slices, while improving resources and avoiding congestion.

III. Forecasting Methods

In this section we present linear and non-linear forecasting methods, including Machine Learning, Deep Learning and Neural Networks [6], as well as its advantages and disadvantages as tools to support network intelligence mechanisms.

A. Classical Machine Learning Methods using Time-lagged Inputs

Another approach for building a model with the task of predicting values is to use Classical Machine Learning (CML) algorithms, an approach in which any type of neural network or Deep Learning algorithm is excluded. In this paper, the tested algorithms were: Linear Regression; Lasso; ElasticNet; Stochastic Gradient Descent (SGD) Regressor; Support Vector Regression; Random Forest; Gradient Boosting Regressor; AdaBoosting Regressor (AdaBoost); and eXtreme Gradient Boosting Regressor (XGBoost) [28].

The Linear Regression, Lasso and ElasticNet are linear approaches for modeling the relationship between a dependent variable and independent variables. The Stochastic Gradient Descent (SGD) regressor models a linear relationship minimizing a loss function with the use of SGD, an iterative method for optimizing an objective function. Support Vector Regression is based on Support Vector Machines (SVMs), an algorithm designed to create a set of hyperplanes to be used for classification or prediction. It is also possible to create non-linear classifiers or regressors applying the kernel trick. In the tests, the Radial Basis Function kernel was used. Random Forest, Gradient Boosting Regressor, AdaBoost Regressor and XGBoost are ensemble methods that take advantage of the construction of multiple models to obtain better predictive performance. Particularly, while Random Forest is a bagging algorithm that trains multiple decision trees with different subsets of the training set and each one vote with equal weight, Gradient Boosting Regressor, AdaBoost Regressor and XGBoost are boosting algorithms that train new model instances focused on the prediction error of the previous instances. These methods allow for the inclusion of external features to the model, such as the hour of the day, which can improve the overall accuracy of the algorithm.

B. Neural Networks

Neural networks are widely used for modeling and predicting time-series data due to their capacity of learning complex patterns. The parameters of the neural network are determined only by the dataset and are not limited to any analytical model, making this approach able to predict non-linear relationships in the data and capable of achieving state of the art performance with a high amount of data, when compared with classical machine learning algorithms.

The neural network architecture is composed of nodes, called neurons, each one with an activation function, that defines the output of the neuron. In a Feed-forward Neural Network (FNN), the neurons of each layer are connected with all the neurons in the previous and next layer.

Neural networks use a set of variables to determine the network structure, and another set of variables to determine how the network is trained. These two sets of variables, known as hyper-parameters, must be set before training. Two of them are essential in the training phase: batch size and number of epochs. The batch size is the number of training examples in one forward and backward pass in the network. The number of epochs defines the number of times that the learning algorithm will work through the whole dataset.

For each model configuration several options can be set, such as the number of hidden feed-forward layers, the number of layers per neuron, the activation function of each neuron or the Dropout value for each layer. Dropout is a neural network mechanism to regularize the network and prevent overfitting [29]. Just like for the classical machine learning methods, it is possible to include external features. This method has the ability to outperform the previous methods for most real-world time-series forecasts. However, its biggest disadvantage is that it is time-consuming and computationally-expensive to train and forecast, when compared with the previous methods.

A Recurrent Neural Network (RNN) is a type of ANN that takes advantage of internal memory to maintain a state with information about the previous inputs. It is possible to maintain or to reset the internal state between the processing of two different inputs. If the state is reset, the recurrent network will only maintain a state depending on the input values in the input sequence.

With the use of RNNs it is possible to store information from arbitrarily long time ago. This is an advantage over the FNNs if there are any long-term dependencies that cannot be learned by time-lagging the data and using it as input. However, if that is not the case, the RNNs will perform the same or worse than the feed-forward neural networks.

The Long Short-Term Network (LSTM) is a particular type of RNN that prevents the exploding/vanishing problem, a difficulty when training general RNNs that makes the update of the network weights too big or non-existing. However, the LSTM is more expensive and hard to train the network due to the complex internal architecture of each neuron, making it more time consuming and harder to find an optimal solution.

A stateful LSTM network stores not only the state of the current batch, but also the state of the previous batches to predict the output, while a non-stateful LSTM network resets the internal states at every batch. Theoretically, a stateful LSTM is able to learn long-term dependencies, even if they are not fed directly in the same batch, while a non-stateful LSTM only learns the dependencies for each input batch.

C. 1-D Convolutional Neural Networks

Convolutional layers have proven to be very effective in the computer vision area, where the spatial locality of the pixels can be used to reduce the computations needed to achieve state-of-the-art performance in image recognition and image classification tasks. Similarly, Convolutional Neural Networks (convnets - ANN with convolutional layers) can be used in sequence processing, extracting features from local input patches and allowing for representation modularity and data efficiency. The property of locality used in computer vision problems can also be helpful in sequence processing, with time being treated as a spatial dimension, like the height or the width of a 2D image. Depending of the dataset, 1-dimensional convnets can be competitive with RNNs or FNNs in the prediction task, with a much cheaper computational cost.

Compared with the previous methods, the hyper-parameters are the same, as well as the model configuration options, but CNNs have two other types of layers. A pooling layer can be used to reduce the dimensionality of each feature map. For regularization of the network, besides Dropout layers, Batch Normalization [30] can also be used. Batch Normalization regularizes the network by reducing the internal covariance shift, normalizing the output of a layer by subtracting the batch mean and dividing by the batch standard deviation. The main goal of the Batch Normalization, however, is not regularization, but to accelerate the training of neural networks. The convolutional layers are characterized, among other things, by the number of output filters, the kernel size, the length of the stride and the activation function.

D. Feature-based Methods

Instead of using the raw values from the time-series directly, a set of features can be extracted from them. Those features can be chosen by a domain expert to provide better insights about the data. Using features extracted from the time-series data for the forecasting instead of using raw lagged values has some advantages: (1) the features can be adjusted and optimized by a domain expert; (2) the number of features is variable and sometimes it is possible to achieve similar or better results with a much lower number of features; (3) usually, it provides better results when there is no high autocorrelation between lags of values in the time-series data (when there is high variation in the data and it is hard to visually detect patterns). However, this approach also has drawbacks, mainly: (1) it has worse results when there is a high autocorrelation between lags of values in the time-series data (when it is possible to visually detect patterns in the data); (2) it depends heavily on the ability of the domain expert to extract the correct features for the prediction; (3) the task of choosing the best features can be time-consuming and non-optimal. The time-series features extracted from the data to be the input of the learning algorithms are the following:

- maximum, minimum, average and standard deviation of the values in the previous time interval;
- values above and below the average in the previous time interval;
- median of the values in the previous time interval;

TABLE I
SUMMARY OF THE MAJOR DIFFERENCES AMONG THE PRESENTED APPROACHES FOR TIME-SERIES FORECASTING.

Approach	**AR**	**ML**	**NN**	**CNN**	**FBM**
Non-linear forecasting	No	-	Yes	Yes	Yes
Fast	Yes	-	No	No	-
Interpretable	Yes	-	No	No	-
Pre-processing Dependent	Yes	No	No	No	Yes
Uses past values	Yes	Yes	Yes	Yes	No
Uses external knowledge	No	No	No	No	Yes

- sum of all values in the previous time interval;
- 0.25, 0.50 and 0.75 quantiles of the values in the previous time interval;
- number of values in ranges in the previous time interval;
- position of the highest and lowest value in the previous time interval;
- different variety of time periods, such as hour, day and week.

The time intervals chosen to extract the features are the previous day and the previous week. The forecasting algorithms chosen were the previously presented algorithms in the classical machine learning methods subsection (Section III-A) and Neural Networks (Section III-B).

E. Summary

A comparison of the major differences among the presented approaches for time-series forecasting (AR - ARIMA; ML - Classical Machine Learning Methods using Time-lagged Inputs; NN - Neural Networks; CNN - 1-D Convolutional Neural Networks; and FBM - Feature-based methods) can be seen in Table I (the symbol '-' means that it depends on the scenario).

The ARIMA approach is the only one that does not perform non-linear forecasting, being ideal for periodical time-series or time-series with a clear trend. The Neural Networks and 1-D convolutional Neural Networks generally have bigger modelling capabilities than classical machine learning algorithms, at the cost of being slower and more difficult to interpret. The Feature-based approach is the only one that uses external knowledge, which is critical when the time-series values are heavily influenced by external events.

Although the model with the best predictions in the test set should be chosen, there are some practicalities that can rule out some methods for being implemented in a forecasting framework. If the scenario requires frequent retrainings, then the ARIMA method can have difficulties in the long-term, because it demands a full retraining (with all the data) everytime, while other methods can be incrementally trained. If the model needs to train fast with low resources, then ARIMA or classical machine learning methods should be preferred over Neural Networks.

IV. Evaluation of forecasting methods applied in network scenarios

In order to evaluate the methods previously presented to produce forecasts, we used two network scenarios. In the first scenario, we consider the number of sessions from a real vehicular network. Our second scenario is a 4G network where the dataset contains important Key Performance Indicators (KPIs). The experiments carried out in different scenarios aim to show the breadth and applicability of the proposed work, and the extended support of these techniques for the multi-slice resource management in 5G scenarios, where each network represents a slice.

After an analysis of both datasets and the identification of the metrics in both scenarios that correlate more to the required network resources, the forecasting results will be presented for the various methods with different hyper-parameters. An analysis is also made to the hyper-parameters that produce the best forecasts. The tests done on these two scenarios, addressing congestion management use cases and resource sharing in a resource-constrained network, are based on the prediction models for 5G network (Section VII), which may assist the network operator to act more effectively and proactively.

To generate the predictions, it will be used the Python programming language, using the *statsmodel* library for the ARIMA implementation, the *Scikit-learn* library for the implementations of classical machine learning algorithms, and the *TensorFlow* library for the neural network implementations. The metric to measure the generated predictions will be the Root Mean Squared Error (RMSE), as described in Equation 1, where y corresponds to the actual observations, $\hat{y}$ to the predicted values and n is the number of predicted values.

$$RMSE(y,\hat{y}) = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2} \qquad (1)$$

The time required to run the algorithms lies between tens of seconds to a few minutes (up to 5 minutes), using a personal laptop with 8 GB of RAM, an Intel Core i7-6500U processor, and GeForce 940M as a Graphics card and implemented with the Tensorflow library. This time is compatible to the predictions required in a multi-slice 5G management.

A. Vehicular Network Dataset Analysis

Vehicle networks play an important role in a smart cities' communication infrastructure. By analyzing the information collected in VANETs, it will be possible to identify behavioral patterns and pave the way for autonomous driving in the advent of the 5G networks [31]. Recently, free public Internet access has been in an upward trend in public transportations of developed cities. In Porto, in the largest mesh network of connected vehicles in the world, more than 200 000 people enjoy free Wi-Fi every day in buses, taxis and municipal service vehicles [10].

The vehicular network dataset used in our experiments contains the number of sessions per hour in the network between the 17th of September to the 8th of December 2018 in the Porto vehicular network. A user session is considered from the moment that a User Equipment (UE) has access to the Internet until its disassociation. The number of user sessions is the metric chosen to forecast, because it can be used as a rough estimate of network traffic volume.

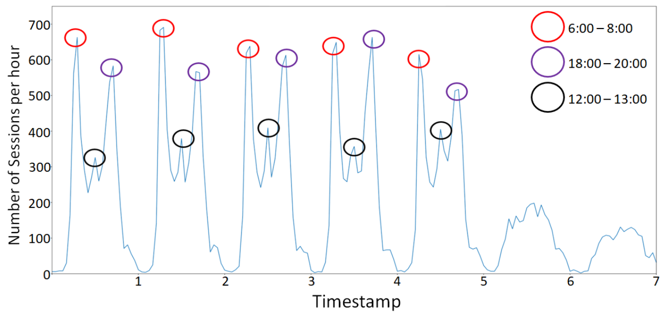


Fig. 1. Number of sessions per hour of one week in the vehicular network.

Figure 1 shows various data patterns in the 22-29 October week. The weekend has significantly fewer sessions than weekdays. In a day, there are two major peaks in the number of sessions: the first is around 7h00 and the second is around 17h00, which is understandable due to the workers' and students' schedules. There is a third minor peak around 12:00, at lunch time. Our auto-correlation function calculates the Pearson correlation coefficient [32] between the same data distribution, separated by various time lags.

Analyzing the dataset, we identified peaks in lags that are multiple of 24 hours, indicating that there is a high correlation with the lag of 24 hours (a day). The highest peak has a lag of 168 hours (a week). These observations help to detect patterns in the data: the peaks in the autocorrelation function indicate similarity with the time interval of the number of lags. The number of sessions is correlated with the number of sessions in the previous days at the same hour in the previous week.

Because the data has daily and weekly patterns, it is not a stationary process. Stationary processes are easier to model, especially for linear forecasting models such as ARIMA. If the data is not directly represented by a stationary process, various types of differentiation can be made to approximate a non-stationary process to a stationary process. For this dataset, according to the time-series analysis previously done, differentiation of one hour, a day and a week will be used in the tests.

B. 4G network dataset analysis

The 5G networks are still far from wide availability, and the process of migrating the current 4G to a 5G network infrastructure will take years. Meanwhile, it is necessary to monitor not only the 5G slices that will be used to support the different verticals, for example vehicular and 4G operator networks, but also 4G traffic, to be able to effectively provide quality of service to the users of both networks. The dataset used for the 4G forecasting has a set of KPIs that describe the operation of a set of cells of an ISP in Portugal during 6 – 27 March of 2019. The interval period between each measurement of a KPI is 1 hour, which means that there are 24 values for each KPI for each cell every day.

The chosen KPI to forecast the 4G traffic is the maximum number of connected users per hour (Figure 2). The data in Figure 2 was normalized in percentage to remove sensitive information about absolute values. There are several patterns in the data that are important to mention. There is a daily pattern: the number of maximum connected users increases every day until around 12 a.m., and then decreases in the afternoon. There is also a weekly pattern, where the weekends typically have a lower maximum number of connected users than the same hour in the workweek. Finally, there is a pattern in the data that can be explained by the monitoring of the KPI of the cells. Everyday, the maximum number of connected users at midnight is zero. This indicates that, at that time of the day, no monitoring measurement is being done of this KPI on the cells.

A seasonal decomposition using moving averages was done using an additive model with a frequency of 1 week. Figure 2 shows 4 plots with the information of the original data, captured trend, seasonality, and residuals from day 6 to day 27. The seasonality plot shows a trend of a lower maximum number of users on the weekends (days 9 and 10, 16 and 17, and 23 and 24). As with to the previous dataset, differentiation of one hour, one day and a week will be used in the tests.

TABLE II
FIVE BEST RESULTS OF THE PREDICTION TASK WITH THE ARIMA APPROACH.

	VANET				4G			
	p	**d**	**q**	**RMSE**	**p**	**d**	**q**	**RMSE**
1	3	168	0	49.50	2	0	4	260.39
2	2	168	2	49.62	3	0	2	261.43
3	1	168	3	49.62	2	0	3	263.88
4	2	168	3	49.63	4	0	2	264.08
5	4	168	1	49.71	2	0	2	271.06

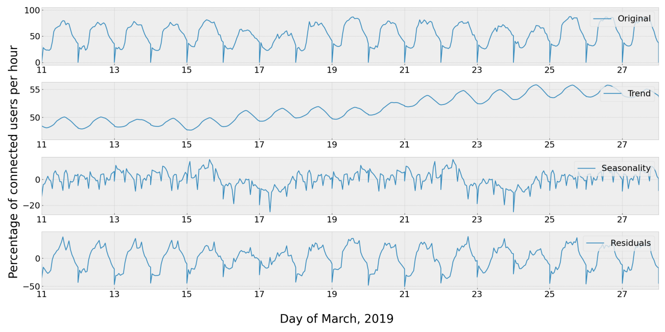


Fig. 2. Seasonal decomposition using moving averages with an additive model, normalized in percentage.

C. Results with ARIMA and CML Methods

For the ARIMA approach, it is possible to set the hyper-parameters p (number of lags included in the model), d (degree of differentiation) and q (size of the moving average window). The parameters p and q will vary between 0 and 5; the parameter d will vary among the values 0 (none), 1, 24 and 168, because of the hourly, daily and weekly patterns of the data. For both datasets, the higher the parameter p, the lower the average RMSE. For the parameter q, the same can be said for the vehicular network dataset. However, for the 4G

TABLE III
FIVE BEST RESULTS FOR EACH SET OF TESTS USING DIFFERENT ADDITIONAL FEATURES FOR BOTH NETWORKS.

	VANET			
Test	**Algorithm**	**Diff.**	**Lag**	**RMSE**
1	Random Forest	0	48	35.66
2	XGBoost Regressor	0	24	36.36
3	Random Forest	0	12	36.43
4	Random Forest	0	48	36.57
5	XGBoost Regressor	0	24	36.70
	4G			
Test	**Algorithm**	**Diff.**	**Lag**	**RMSE**
1	SVR	1	12	127.49
2	Random Forest	1	24	132.56
3	Random Forest	0	12	132.85
4	XGBRegressor	1	24	134.91
5	Gradient Boosting R.	1	24	135.44

network, the value with lowest RMSE is when $q = 2$. The five best prediction models for both networks are the ones in Table II. For the vehicular network, it can be seen that the differentiation parameter is 168 for the five best results, and for the 4G network it is 0. The differentiation seems to have higher importance for an accurate ARIMA prediction model than p or q parameters.

In the comparison with classical machine learning methods, the parameter d will be tested with the values 0, 1, 24 and 168 (the same as for the ARIMA differentiation) and the l (lag number) will vary among 1, 12, 24, 48 and 168. Additional features have been added to the models. Three features of the data (day of the week, workday, and hour of the day) were considered useful to forecast the number of sessions in the next hour, and their effect on the results was tested.

For the vehicular network, most algorithms have better results when the differentiation lag is 168. However, three ensemble algorithms (Random Forest, Gradient Boosting Regressor and XGB Regressor) have better results when there is no differentiation. These ensemble algorithms are able to learn non-linearities in the data with the use of several machine learning models, instead of models that are only able to capture linear relations, where the data stationarity is needed, like Linear Regression. The five best results with this approach are presented in Table III. For the 4G network, the best results have one hour as differentiation. The same happens for other combinations of additional features. On average, the higher the number of lags, the better the performance, as expected. There is only slight improvement in going from 12 to 24. In the vehicular network, the best results have no differentiation. The impact of the additional features is not much, because the RMSE does not vary much. The best result (RMSE of 35.66 session per hour) was achieved with only the day of the week as additional feature, with the Random Forest algorithm, no differentiation lag and with 48 lags as input. In the 4G network, the best result (RMSE of 127.49 users per hour) is achieved with only the workday as feature, with the SVR algorithm. Furthermore, most of the best results are achieved with one-lag differentiation and 24 lags as input. The algorithms with the best results overall are the SVR, the XGB Regressor and the Gradient Boosting Regressor.

TABLE IV
FIVE BEST RESULTS ON THE TESTS USING A FEED-FORWARD NEURAL NETWORK FOR THE VEHICULAR NETWORK.

Best Results	d	l	**Hidden Layers**	**Neurons per Layer**	**Dropout Value**	**RMSE**
1	1	12	4	200	0.2	31.44
2	0	24	2	200	1.0	31.58
3	1	12	3	200	0.2	32.30
4	1	24	4	200	0.2	32.34
5	0	12	2	200	1.0	32.74

D. Results with Neural Networks

In the tests with the *Feed-forward Neural Networks* (FNNs) for the vehicular network, the hyper-parameter d varied among 0, 1, 24 and 168, and the l (lag number) varied among 4, 12, 24 and 168. 24 neural network configurations were tested, which varied in the number of feed-forward hidden layers (1 to 4), in the number of neurons per layer (20 or 200) and in the Dropout value (1 (None), 0.2 or 0.5). All the neurons have Rectified Linear Unit (ReLU) as the activation function, with the exception of the neuron of the last layer, that has no activation function. The batch size will be set to one, and the Optimizer will be Adam [33]. The neural network will be trained for 500 epochs, and the presented RMSE will be the lower RMSE in the cross-validation set along 500 epochs. In the vehicular network all the five best results have either one lag differentiation or no differentiation at all. Moreover, all the five best results have as number of lags 12 or 24, and 200 neurons per layer. The same set of additional features as in the previous subsection was tested with the best five results.

The five best results for the vehicular network with no additional features are presented in Table IV, along with their hyper-parameters. All the five best results have either one lag differentiation or no differentiation at all. Moreover, all the five best results have as number of lags 12 or 24, and 200 neurons per layer.

For the 4G network prediction task, similar hyper-parameters were tested, and the workday feature has the lowest average RMSE and also the best result overall (115.54 connected users per hour). It is an improvement from the result of RMSE of 127.49 users per hour in the previous section.

The Recurrent Neural Networks (RNNs) tests are similar to the FNNs tests with the exception of some hyper-parameters. Due to the time that LSTMs take to train, the hyper-parameter space must be reduced. For both network datasets, the d will be fixed to 0, due to in the previous subsection, the best values for the differentiation were 0 or 1, with little variation between them. The l will vary between 12, 24 and 168. The models tested for the vehicular network have between one and three LSTM hidden layers, with 20 or 200 neurons each layer. The Dropout mechanism will have Recurrent Dropout enabled (varying its values between 1 (no Dropout) and 0.2), and stateful/non-stateful LSTMs will also be tested.

In the five best results for the vehicular network, tests with the lag number of 12 had much better results than the tests with the lag number 24. The stateful forecast results are identical to the non-stateful forecast results, which indicates that all the information needed for the forecast of the next value is in the previous 12 lag values. Another indication is the fact that the results are very similar to the results with the feed-forward neural networks. In terms of features, only the feature of the workday alone improved the results, in contrast with previous subsections, where the best results come from the tests with only the workday feature. All other tests showed a degradation of the RMSE. The lowest RMSE achieved was of 31.36 sessions per hour, worse than the best result achieved with the FNNs.

The models tested for the 4G network have between one and four LSTM hidden layers, with 20 or 200 neurons each layer. The Dropout mechanism will have Recurrent Dropout fixed at 0.2 and the tests were done only with non-stateful LSTM layers, due to the small effect of their variation in the previous tests, and to reduce the hyper-parameter space. For additional features tests, it was used 5 sets. In the first test there were no additional features, and on the following tests, the additional features were varied. The results for RMSE were 113.70, 122.91, 122.85, 118.48, and 121.56 respectively. The best result of 113.70 was achieved without additional features.

In the Convolutional Neural Network (CNN) methods, d was set to 1, and the l varied among 12, 24 and 168. The stride length of all convolutional layers used is of size one, and the activation function used is ReLU. The number of output filters varied between 20 and 200 and the kernel size was set to four. The tests varied the number of convolutional hidden layers from one to four, with and without max pooling layers between convolutional layers. Networks with and without batch normalization between convolutional layers were also tested. Finally, two types of layers were tested: Flatten, to reshape the output of a convolutional layer from a 3D in a 2D array by concatenating the result of multiple filters, and Global Average Pooling 2D, that does the same transformation from a 3D array into a 2D array by selecting the maximum value for each array in the third dimension.

As an example, for the vehicular network (Table V), the results with Global Max Pooling layer have the worst results, due to its inability to learn patterns in the data. On the other hand, the tests with the Flatten layer and without batch normalization have the best results from the test set. Analyzing the lag number, the higher the lag number, the worse the overall results, with the best lag number being 12. Given that this method is adequate only for sequences, no additional features were tested. These tests are very time-consuming (taking approximately ten times more the time to train than a FNN for the same number of epochs and same number of neurons) and the results were also not satisfactory in the 4G network (the lowest RMSE was 38.59, much bigger than most RMSEs achieved using FNNs (Table IV)).

E. Other approaches with ensemble models

We approached the improvement of the prediction accuracy using ensemble models. An ensemble combines the result of several models to achieve more accurate forecasts. The best feed-forward network model with only the previous number of sessions per hour as input and external features (feed-forward neural network with one-lag differentiation, a lag number of 12 and the weekday as additional feature) was combined with the best model that has as input only time series features and external features (Random Forest with all features except quantile features). The combination of the two models was done with an average of the forecasts. The vehicular network' RMSE was of 28.44 sessions per hour, a new best record.

TABLE V
BEST RMSE RESULTS IN THE CONVNET TESTS AND ITS HYPER-PARAMETERS, IN DESCENDING ORDER, FOR THE VEHICULAR DATASET.

Best Results	Flatten/ Global Max Pooling	Batch Norm	l	RMSE
1	Flatten	FALSE	12	38.59
2	Flatten	FALSE	12	38.87
3	Flatten	FALSE	12	39.60
4	Flatten	FALSE	12	41.41
5	Global Max Pooling	FALSE	12	41.71

One a way to improve the results is to add another predictor. Another test was made with both predictors used before, and with the best model using the recurrent neural networks (recurrent neural network not stateful, with a lag number of 12 and with the workday as additional feature). The output is now the average of the forecasts of the three models. The RMSE for the vehicular network was 26.22 sessions per hour, which surpasses the previous best result.

In the 4G network, the models used in these tests will be the best feed-forward neural network model (RMSE of 115.54 users per hour), the best LSTM model (RMSE of 113.70 users per hour) and the best result with the Classical Machine Learning models (SVR algorithm, with an RMSE of 127.49). Because the neural network models have better results than the SVR algorithm, an ensemble of only both neural networks will be tested. Then, an ensemble of the three models will also be created to check if the SVR forecasts improve the final result.

The forecasts with the SVR algorithm increase the RMSE error. The best result in the cross-validation set is achieved with an ensemble of the feed-forward neural network and the LSTM network, where it is possible to achieve an RMSE as low as 104.94 connected users per hour.

The final tests are performed by training the models with the best results, considering the training and cross-validation sets, and testing with the test set. Five different models will be used for the tests for both networks:

1) best feed-forward neural network (FFNN) configuration;
2) best recurrent neural network (LSTM) configuration;
3) algorithm with the best result of the classical machine learning approach;
4) ensemble with the best feed-forward neural network and the best recurrent neural network;
5) ensemble with the best feed-forward neural network and, the best recurrent neural network and the best result of the classical machine learning approach.

TABLE VI
RMSE RESULTS WITH THE TEST SET OF THE BEST FIVE TESTS IN BOTH NETWORKS.

Test	Models	VANET RMSE	4G RMSE
1	FFNN	26.59	153.47
2	LSTM	23.40	177.54
3	SVR	29.10	197.13
4	FFNN + LSTM	24.10	152.41
5	FFNN, LSTM, SVR	22.11	164.53

The results of the tests for the vehicular dataset (26 Nov - 8 Dec, 2018) can be seen in Table VI, and they show that the RMSE improved from the previous best tests (previously RMSE of 29.99, 31.36, 38.45, 28.44 and 26.82 sessions per hour, respectively), due to the larger set of training data. For a small margin, the recurrent neural network performs better than the feed-forward network in the RMSE, as opposite to the tests in the cross-validation set. The model ensembles, just like in the tests done with the cross-validation set, outperform the models separately. The best result was achieved with an ensemble of the feed-forward network, the recurrent network and the random forest algorithm, with an RMSE of 22.11 (Figure 3).

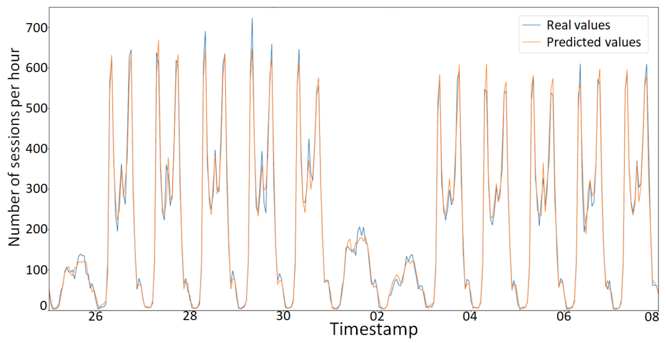


Fig. 3. Ensemble model forecasts with three predictors in vehicular network.

In the results of the 4G network, on average, the error is higher than in the cross-validation set, due to the data distribution being less similar to the training set (26-31 March, 2019). The ensemble with the feed-forward neural network and the recurrent neural network has the best result, just like in cross-validation set (Figure 4). The data in Figure 4 was normalized in percentage to remove sensitive information about absolute values.

Therefore, we deployed the best models for each scenario - FFNN, LSTM and SVR for the vehicular dataset and FFNN and LSTM for the 4G network dataset. It is important to note that the models with the best score may not be the chosen models to deploy in the framework. As noted in Section III-E, there are other important metrics to take into account when choosing a model, such as training time, prediction time, resources needed to use the model, interpretability, etc. In the case of ensemble models, like those that we have chosen, the prediction time and the heavy use of resources must be taken

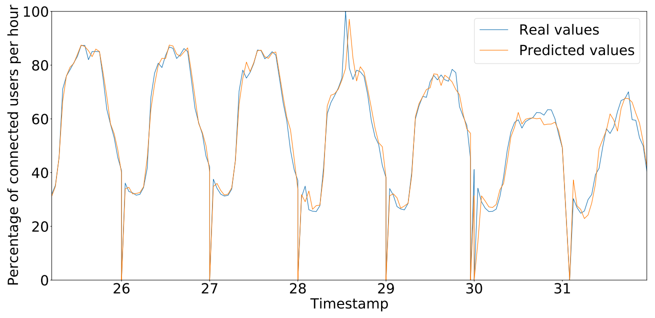


Fig. 4. Ensemble model forecasts with two predictors in 4G network.

into account. The proposed approach is able to configure the metrics chosen to select the best model, and it is important to leave it open to the network manager for custom configuration.

V. Predictor Framework

To be able to execute experiments with forecasting methods to evaluate, simultaneously, the best alternatives to support network management, we design and implement a Predictor Framework (PF) [34]. The PF is able to test, in real-time, different algorithms with different metrics, including ensembles of the best algorithms. Its proposed architecture has the following characteristics to enable a resilient and distributed framework:

- simple and flexible API to be used for clients that do not need to understand the internals of the predictor;
- easy addition and removal of predictors, according to the needs of the clients;
- creation of ensemble predictions by taking into account the predictions made by multiple models;
- distributed prediction and training – it is possible to distribute the predictors in different systems, or even to distribute the training phase and the prediction phase for the same predictor, to allow for fast prediction and fine tuning of the model;
- horizontal scaling - the predictor API and the message broker can be replicated to handle the growing number of predictors;
- real-time prediction - the predictions are distributed and performed in real-time, providing a constant prediction of the gathered metrics.

In our context, “clients” can be from policy or monitoring instances doing management of 5G slices to dashboards for visual monitoring of behavior.

The high-level view of the PF, depicted in Figure 5, is composed of five main components: the *Prediction API*, the *Predictor Message Broker* (PMB), the *Metric Prediction Component* (MPC), the *Metric Message Transformation Component* (MMTC) and the *Metric Prediction Orchestrator* (MPO).

The *Prediction API* is the interface that allows a client to: (*t1*) train the predictor for a given metric; (*t2*) predict a value for a time series system/metric; (*t3*) save new time series data for later training; and (*t4*) change the training parameters. The Prediction API is a REST microservice that encapsulates in a

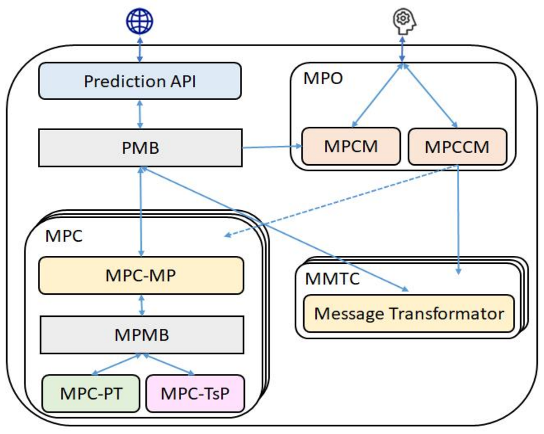


Fig. 5. Macro Architecture of the Predictor Framework.

simple and convenient way the functionalities of the predictors without any operability loss, to be used by end-users.

All tasks are available as REST endpoints, and they receive the name of the predictor as a parameter. For the task (*t1*), the client should send the current value of the time series to be available for the predictor to use it, along with its timestamp and a list of additional attributes, if needed. The list of additional attributes is optional and depends only on the specific predictor. The API will obtain the result of the predictor and return it to the client, along with the timestamp of the predicted point and any additional attributes. The same input attributes are needed for task (*t3*), but the API saves the point and does not return data to the client. The parameters that must be sent for the task (*t2*) are the start and end date of the data to consider for the training phase, besides any additional attributes. Finally, for the task (*t4*), the parameters are the timestamp of the next training, the period between two training phases, the data period that should be considered for each training and optional additional attributes. Other endpoints are also accessible to rebuild or clear the internal state of the predictor (only applicable when the predictor depends on previous values), and to obtain information about the predictor configurations. In contrast with the internals of the architecture, which uses a publish-subscribe system, the interaction between the external world and the API is done using a request-response system. This is more adequate for the client, since it makes single requests for predictions and does not need to subscribe to any type of messages besides the response to its predictions.

The Predictor API receives the requests from external clients and, according to an internal message protocol, sends a message to the *Predictor Message Broker (PMB)*. This component is responsible to, according to the recipient of the message, forward the message to the corresponding *metric predictor* or *message transformation component*. The broker enables the multicast distribution of messages, allowing it to build several prediction components for the same metric, each one with a different prediction model. That is, the information that comes from the monitoring system is distributed internally without any overhead for the slice (network) or other services running on the slice. Besides, these predictions can be combined to build an ensemble predictor that performs better than the individual prediction components. To address these requirements, the Predictor API and Message Broker (PMB) provide fault tolerance, disaster recovery, low response times and high scalability. These functionalities are important in high throughput scenarios, where failures or high latency may happen.

The core component of the framework is the *Metric Predictor Component (MPC)*. Each instance of the MPC is responsible for predicting the time series data for a given dynamic system/metric: it receives the messages directly from the message broker and predicts the time series points. The MPC is composed by four sub-components: Metric Predictor (MPC-MP), Metric Predictor Message Broker (MPMB), Predictor Training (MPC-PT), and Time-series Persistence (MPC-TsP).

The Metric Predictor Message Broker is the internal message broker of the predictor component, and it routes messages between its sub-components. The Predictor Training does online or offline training of the model to update it with the newly received data. For offline training, the training task parameters (date of the next training, training data window, etc.) can be adjusted via the Prediction API. The new model will be uploaded to the Metric Prediction component via the internal message broker. The Time-series Persistence stores the data to optimize the read and write operations for time-series data, to be used when training the model.

To enable an ensemble prediction, a reduction step is also required to join the predictor results and perform the adequate transformation. The *Metric Message Transformation Component* performs that transformation by receiving the messages in the prediction message broker and applying custom functions for each predictor component message, before inserting the transformed messages into the broker. The transformer can also be used for other data transformations, such as data units transformations, data anonymization or other scenarios. In the use cases presented in this article, the transformer will be used to create an ensemble prediction.

The instances of the metric predictor and the transformation components are orchestrated by the *Metric Prediction Orchestration Component (MPO)*, which has two internal components: the Monitoring (MPCM) and the Configuration and Management (MPCCM). The configuration and management component is able to instantiate and manage the life cycle of the predictor and the transformation instances. These instances can be dynamically added or removed when ordered by the prediction framework manager, ensuring their availability and reliability, restarting them when they crash. The instances of metric predictors and transformations (MMTC) start by establishing a connection with the message broker (PMB) and make the link with the monitoring component (MPCM) to report their activity. MPCM accesses the logs and the exchanged messages, assessing the current state of operation of the instances, to flag MPCCM about anomalies (an instance crash for example).

The process of training and testing with the framework will be explained:

- Firstly, it is needed to create the MPCs, one for each approach used, with its specific train and prediction algorithms;

- If it will be used more than one MPC at the same time (ensemble approaches), it is also needed to create an MMTC to calculate a single prediction based on the several predictions calculated by the MPC;
- To perform the training of the models, it is needed to send a request to the Prediction API with the start and end date of the training data. Each MPC will perform its training separately and store the new model;
- To perform a prediction, it is made a request to the Prediction API, that will send it to the PMB;
- The PMB will send the message to all the MPCs of that metric, that will perform the prediction and send it to the PMB;
- If there is an MMTC (for ensemble algorithms), it will fetch the predictions and calculate a single result, that will be sent to the Prediction API to create a response to the user with the requested prediction.

The PF was designed to be easy to use (no requirements on programming knowledge), easy to deploy, can be used in automated deployment workflows, and lightweight to work in slices. By enabling the execution of several containers, the framework allows to predict different metrics of the network simultaneously, such as the number of sessions, bandwidth, delays, and others, being able to configure according to the operators' needs. The results that will be presented in the next section, which depicts the integration of the prediction framework in a 5G architecture, were obtained through the use of our framework.

VI. Distributed Intelligence in 5G environments

The 5G architecture is defined to support connectivity and data services, enabling virtualization of network functions and software-defined networks, among other techniques. The 3rd Generation Partnership Project[1] (3GPP) proposes a 5G system architecture to leverage service-based interactions between the network functions of the control plane [35]. Some of the main components of this reference 5G architecture are the following:

- Policy Control Function (PCF): uses its information about the slice usage (bandwidth, active services, clients, etc) to manage the slice/network behaviour. PCF provides and maintains the policy rules to control plane functions;
- Session Management Function (SMF): supports session management, selection and control of user plane functions, downlink data notification and roaming;
- User Plane Functions (UPFs): handles the user plane path of Packet Data Unit (PDUs) Sessions. UPF selection is performed by the SMF;
- Network Data Analytics Function (NWDAF): represents operator managed network analytics logical control, and provides slice specific network data analytics.

These components are directly related to our framework (Section V), here called by *Predictor Framework* (PF). Our PF can be seen as an implementation of NWDAF.

[1]https://www.3gpp.org/

This architecture has been implemented in Mobilizer 5G Project[2] (Figure 6), a portuguese 5G project that aims to foster 5G implementation and deployment in Portugal. Our prediction framework has been deployed in this architecture and has been integrated with both Assurance and PCF modules. These modules are implemented in Docker containers, and orchestration, management, and automation of network and edge computing services is performed through ONAP[3].

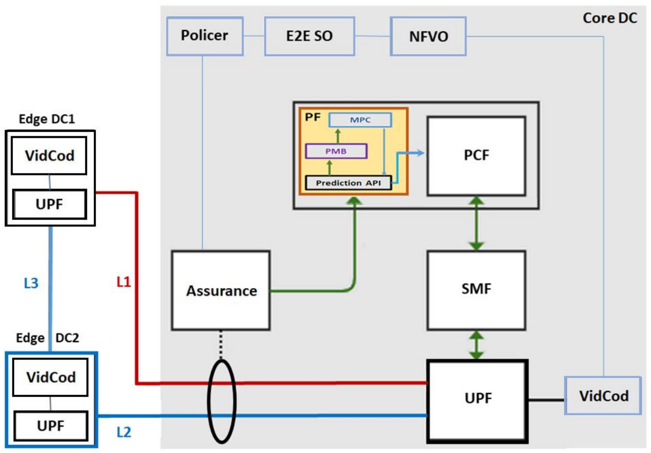


Fig. 6. Prediction framework integration in a 5G Architecture.

Figure 6 depicts a simplified version of the 5G architecture deployed in our premises. To illustrate the prediction framework functionalities, we exemplify a link congestion use case of a virtual Video Coding (VidCod) scenario.

Policer, end-2-end Service Orchestrator (E2E SO) and Network Function Virtualization Orchestrator (NFVO) are the entities needed to orchestrate the VidCod services in Edge DC1 and DC2.

The Prediction API of the PF receives KPIs from the Assurance module, and activates the Metric Predictor Component (MPC) through the Predictor Message Broker (PMB). MPC evaluates the received KPIs against the expected behavior model (learned over time), and if it recognizes a distortion above the expected threshold (through the forecasting approaches), it signals the Prediction API to send an advertisement about the predicted anomaly to the PCF. By predicting that link 1 (L1 - red line in Figure 6) tends to become congested, the PF signals the PCF when it is expected that L1 overload will occur. The PCF and SMF do all the preparation for changing the traffic from L1 to L3-L2 (blue lines). The intelligence of this operation lies in the fact that the PF does not recognize anomalies when they are occurring, but rather tries to predict the occurrence of the anomaly in the near future. Upon receiving the prediction, the PCF, considering the current slice status (bandwidth, services, options) can act preventively by changing the slice policy, or even indicating the need for more resources to the orchestrator. The result of this joint operation Assurance-PF-PCF-SMF can prevent the

[2]https://5go.pt
[3]https://www.onap.org/

anomaly from occurring and consequently maintain the slice's QoS and consequently the entire network.

With PF, it is possible to use a prediction instance for each KPI or combine KPIs per prediction instance. This combination can be used on all slices. In addition, by using microservice-based construction, PF can be easily instantiated in the creation of a new slice.

VII. Using the predictions to improve a 5G Network

The aim of this section is to use the previous approaches to act autonomously to improve the resource management of a 5G network with multiple slices, a slice for the vehicular network and a slice for the 4G operator network.

We consider two scenarios: in the first scenario, the network has limited resources, and must be divided between those resources for multiple slices as optimally as possible, maximizing the network utilization and minimizing the degraded network sessions; the second scenario has the aim of preventing network congestion while minimizing the use of resources for each slice, managing dynamically the resources needed. These approaches are tested considering the distributed intelligence described in the previous section, through the integration of the PF with the session and policy control functions, where the slices' resources are adjusted autonomously as a result of the predictions in the PF.

The implementation and testing of the scenarios will be done using the same data as described before, using the Python programming language to analyze the results.

A. Dividing a Resource for Multiple Slices

In this scenario, the chosen network resource is the maximum number of sessions per hour in the network, which will be limited to a maximum value. The maximum number of allowed sessions will have to be split across all slices to reduce the number of lost sessions in the network. This split can be made statically (setting a fixed threshold for each slice) or dynamically (adjusting in real-time the threshold for each slice, according to demand). Three splitting algorithms will be compared: a fixed-threshold algorithm, where the splitting of the maximum number of sessions between the slices is done with a fixed threshold for all slices, defined by an optimization search on the past values of that resource for that slice; an optimal dynamic threshold algorithm, that takes its decisions of the splitting based on the future values of the metrics of all slices (as if the forecasts were always exactly correct); and a best-possible dynamic threshold algorithm, that makes the threshold decisions based on the forecasts made for the next hour of the slices. Furthermore, the maximum number of sessions available for the network in the tests also vary from 0 to the maximum number of sessions needed for the network to serve all users (when the number of sessions lost is 0).

The performance metric that is minimized for this problem is the number of sessions above the threshold for that slice, as represented in Equation 2 using Iverson bracket notation, where $n_s(x)$ is the number of sessions of a slice.

$$L(n_s(x), threshold(x), t) = \sum_{t=0}^{length(n_s(x))} n_s(x)[t] \\ -threshold(x)[t][n_s(x)[t] > threshold(x)[t]] \tag{2}$$

In the dynamic threshold algorithm, the value of the threshold will change according to the forecasts of the next hour, to better adjust to the network traffic fluctuations, for better utilization of the maximum number of available sessions in the network. It is possible to set the minimum number of sessions per slice, which prevents a forecast of a slice with a low number of sessions from blocking the slice for accepting new sessions.

To manage the resources of each slice, the dynamic threshold algorithm is presented below (Algorithm 1). In this algorithm: $p_s(x)$ is the prediction of required resources and sessions for slice x; min_s is the minimum resources and number of sessions guaranteed in each slice x; NoS is the number of active slices; R_{max} is the amount of physical resources that can be divided through the different slices; and $n_s(x)$ are the resources assigned to slice x.

Algorithm 1 Dynamic Threshold Algorithm
```
procedure DYNAMIC_THRESHOLD($p_s$,$min_s$,$NoS$,$R_{max}$)
    $free\_res = R_{max} - \sum_x min_s(x)$
    for $x$ in $NoS$ do
        if $p_s(x) \leq min_s(x)$ then
            $n_s(x) = min_s(x)$
        else
            $n_s(x) = min_s(x) + \frac{p_s(x) - min_s(x)}{\sum_x p_x(x)} * free\_res$
        end if
    end for
    Return $n_s$
end procedure
```

The dynamic threshold algorithm calculates the resources assigned for each slice following two conditions: 1) if all predictions for the next timeframe are lower than the minimum resources assigned for that slice, it is assigned to that slice its minimum resources; 2) otherwise, it is assigned to that slice its minimum resources plus a number of additional resources, based on the predicted required resources and the available resources in the network.

We also consider a dynamic threshold algorithm where the forecasts are the real values (to establish a baseline for the best possible result).

The evaluation is performed by running the fixed and dynamic algorithms. These algorithms are running by varying the number of sessions available in the network (starting with no session available to all the sessions available). We then evaluate the percentage of sessions in the network above the number of sessions allowed for each slice. Figure 7 shows the results: the percentage of lost sessions decreases as the number of sessions available in the network increases for all algorithms, as expected. The fixed-threshold algorithm performs worse than the dynamic threshold one, while the dynamic threshold algorithm performs only marginally worse

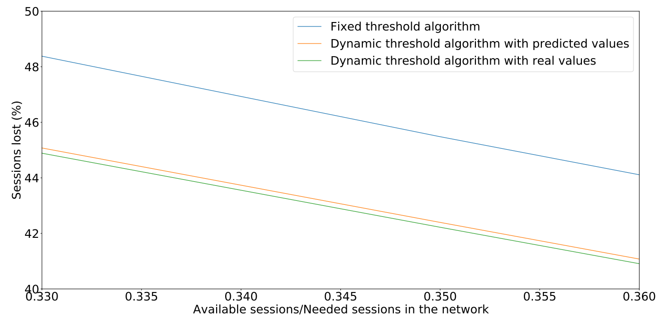


Fig. 7. Percentage of sessions lost for different algorithms.

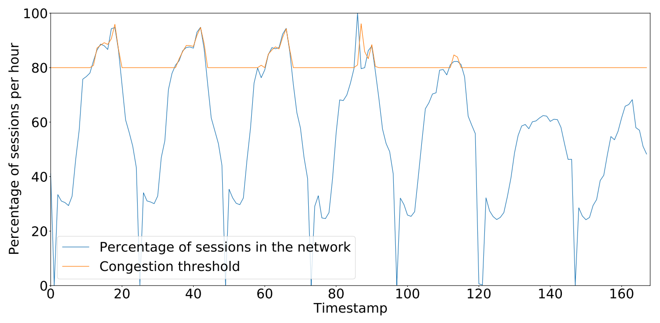


Fig. 8. Proactive management of the congestion threshold in the network.

with the forecasts than with the real values. This result shows the effectiveness of applying the dynamic threshold algorithm with forecasts for a multi-slice network.

B. Preventing Network Congestion

In large core networks, the goal of the management may be to give enough resources for each slice to provide to its users the minimum congestion possible, while saving as many resources as possible for other slices or for network emergencies. Thus, the chosen resource is the maximum number of sessions per slice. Because the available metrics do not allow one to derive network congestion, we assumed that the network is congested when the number of sessions per hour is higher than 80% of the maximum value of the number of sessions in the dataset. The test set used is the same week of the data from both slices. Three evaluation methods will be proposed to calculate the congestion in the network.

In the first method, all sessions above the threshold of 80% of the maximum number of sessions in the dataset are counted as congested sessions. The second method considers a reactive action in the slice upon congestion. If the number of sessions in the slice is above the threshold, the resources in the network are adapted in the next time-frame to accommodate the value of the number of sessions in the previous time-frame. Finally, the third method uses the forecasts from both slices to perform proactive management of the congestion. When the number of sessions forecasted in the next time-frame for each slice is above the 80% threshold, the network prematurely adapts its resources for each slice to accommodate the number of sessions forecasted, to prevent the congestion in the network. Figure 8 shows the proactive threshold adaptation.

The performance metric for the three methods is the percentage of sessions above the threshold, which is shown in Table VII. The reactive management of the congested threshold reduces the number of congested sessions for both slices, and the proactive management of the congested threshold reduces it even more, as expected.

An interesting fact is the high number of congested sessions in the entire network when compared with both slices, for the two first methods. This is due to the aggregated traffic having its mean value closer to the 80% threshold, with more sessions being above the threshold than before, and with lower peaks when compared with the mean value of the data. However, the proactive management of the network can adapt to it, because the aggregated congestion management has less congested sessions than the sum of the congested sessions in both slices. Therefore, the results show that the proactive management reduces even further the congestion in the network, through the utilization of accurate predictions in the network. Other than congestion reducing, the proactive management also reduces the resources needed for each slice, by accurately predicting what are the resources needed for all slices, allowing for a better resource allocation.

TABLE VII
PERCENTAGE OF CONGESTED SESSIONS IN BOTH SLICES, ACCORDING TO THE DIFFERENT METHODS OF CONGESTION MANAGEMENT USED.

	Vehicular Slice	**4G Slice**	**Entire Network (Aggregated)**
No threshold adaptation	2.53%	1.28%	2.51%
Reactive management	2.00%	0.55%	0.83%
Proactive management	1.02%	0.32%	0.38%

VIII. Conclusion

With the implementation of 5G networks, automated decisions and autonomous action on the network are essential for the management of 5G networks. In this paper, we present an overview of linear and non-linear forecasting methods, and discuss their use to improve management in 5G networks. By using our proposed forecasting framework, we evaluated an extended set of forecasting approaches, including ensembles of the best ones, and tested the use of statistical features extracted from the time-series to forecast the number of users' sessions in different networks.

Through these forecasting models, a multi-slice resource management approach has been proposed, to ensure that the resources needed for each slice are divided fairly according to the forecasts of all slices. Moreover, the goal was to optimize the usage of a resource for slices: each slice should have enough resources to provide to its users the minimum congestion possible, while saving the maximum of resources for other slices or for network emergencies. The proposed proactive congestion management is able to forecast the congestion and act accordingly in the network to increase the network resources for a slice. This approach has finally been integrated in a 5G architecture.

Future work aims to test more use cases with the prediction framework in the 5G architecture, mainly forecasting network problems and anomalies.

Acknowledgment

This work is supported by the European Regional Development Fund (FEDER), through the Regional Operational Programme of Lisbon (POR LISBOA 2020) and the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework [Project 5G with Nr. 024539 (POCI-01-0247-FEDER-024539)].

References

[1] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5g: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201–1221, June 2017.

[2] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5g network slicing using sdn and nfv: A survey of taxonomy, architectures and future challenges," *Computer Networks*, vol. 167, p. 106984, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128619304773

[3] D. Ferreira, C. Senna, P. Salvador, L. Cortesão, C. Pires, R. Pedro, and S. Sargento, "Root cause analysis of reduced accessibility in 4g networks," in *Machine Learning for Networking*, S. Boumerdassi, É. Renault, and P. Mühlethaler, Eds. Cham: Springer International Publishing, 2020, pp. 117–133. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-030-45778-5_9

[4] D. Bega, M. Gramaglia, A. Garcia-Saavedra, M. Fiore, A. Banchs, and X. Costa-Perez, "Network slicing meets artificial intelligence: An ai-based framework for slice management," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 32–38, 2020.

[5] F. Grando, L. Z. Granville, and L. C. Lamb, "Machine learning in network centrality measures: Tutorial and outlook," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 102:1–102:32, Oct. 2018. [Online]. Available: http://doi.acm.org/10.1145/3237192

[6] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 92:1–92:36, Sep. 2018. [Online]. Available: http://doi.acm.org/10.1145/3234150

[7] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, April 2020. [Online]. Available: https://link.springer.com/article/10.1007/s10462-020-09825-6

[8] N. Huynh, D. Hoang, D. Nguyen, and E. Dutkiewicz, "Optimal and fast real-time resource slicing with deep dueling neural networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1455–1470, 2019.

[9] J. Koo, V. Mendiratta, M. Rahman, and A. Walid, "Deep reinforcement learning for network slicing with heterogeneous resource requirements and time varying traffic dynamics," in *15th International Conference on Network and Service Management (CNSM 2019)*. IEEE, 2019.

[10] P. M. Santos *et al.*, "PortoLivingLab: An IoT-based sensing platform for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 523–532, apr 2018. [Online]. Available: https://doi.org/10.1109/jiot.2018.2791522

[11] X. Ding, S. Canu, T. Denoeux, T. Rue, and F. Pernant, "Neural network based models for forecasting," in *in Proceedings of ADT'95*. Wiley and Sons, 1995, pp. 243–252.

[12] A. Azzouni and G. Pujolle, "Neutm: A neural network-based framework for traffic matrix prediction in SDN," *CoRR*, vol. abs/1710.06799, 2017. [Online]. Available: http://arxiv.org/abs/1710.06799

[13] M. Barabas, G. Boanea, A. B. Rus, V. Dobrota, and J. Domingo-Pascual, "Evaluation of network traffic prediction based on neural networks with multi-task learning and multiresolution decomposition," in *2011 IEEE 7th International Conference on Intelligent Computer Communication and Processing*. IEEE, aug 2011. [Online]. Available: https://doi.org/10.1109/iccp.2011.6047849

[14] A. Haar, "Zur theorie der orthogonalen funktionensysteme," *Mathematische Annalen*, vol. 69, no. 3, pp. 331–371, Sep 1910. [Online]. Available: https://doi.org/10.1007/BF01456326

[15] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," GMD Report 148, 2001. [Online]. Available: http://www.faculty.jacobs-university.de/hjaeger/pubs/EchoStatesTechRep.pdf

[16] Z. Chen, J. Wen, and Y. Geng, "Predicting future traffic using hidden markov models," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*. IEEE, nov 2016. [Online]. Available: https://doi.org/10.1109/icnp.2016.7785328

[17] H. Singh, "Performance analysis of unsupervised machine learning techniques for network traffic classification," in *2015 Fifth International Conference on Advanced Computing Communication Technologies*, Feb 2015, pp. 401–404.

[18] R. Madan and P. S. Mangipudi, "Predicting computer network traffic: A time series forecasting approach using DWT, ARIMA and RNN," in *2018 Eleventh International Conference on Contemporary Computing (IC3)*. IEEE, Aug. 2018. [Online]. Available: https://doi.org/10.1109/ic3.2018.8530608

[19] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrows intelligent network traffic control systems," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2432–2455, Fourthquarter 2017.

[20] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Traffic prediction based on random connectivity in deep learning with long short-term memory," *CoRR*, vol. abs/1711.02833, 2017. [Online]. Available: http://arxiv.org/abs/1711.02833

[21] A. Martin, J. Egaãa, J. Flórez, J. Montalbán, I. G. Olaizola, M. Quartulli, R. Viola, and M. Zorrilla, "Network resource allocation system for qoe-aware delivery of media services in 5g networks," *IEEE Transactions on Broadcasting*, vol. 64, no. 2, pp. 561–574, 2018.

[22] P. Casas, "Machine learning models for wireless network monitoring and analysis," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, April 2018, pp. 242–247.

[23] W. Guan, X. Wen, L. Wang, and Z. Lu, "Network slicing management of 5g ultra-dense networks based on complex network theory," in *IEEE Globecom Workshops*. IEEE, 2017.

[24] T. Li, X. Zhu, and X. Liu, "An end-to-end network slicing algorithm based on deep q-learning for 5g network," *IEEE Access*, vol. 8, no. 1, pp. 122 229–122 240, 2020.

[25] H. Chergui and C. Verikoukis, "Offline sla-constrained deep learning for 5g networks reliable and dynamic end-to-end slicing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 350–360, 2020.

[26] F. Fossati, S. Moretti, P. Perny, and S. Secci, "Multi-resource allocation for network slicing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1311–1324, 2020.

[27] F. Song, J. Li, C. Ma, Y. Zhang, L. Shi, and D. N. K. Jayakody, "Dynamic virtual resource allocation for 5g and beyond network slicing," *IEEE Open Journal of Vehicular Technology*, vol. 1, pp. 215–226, 2020.

[28] T. D. Buskirk, A. Kirchner, A. Eck, and C. S. Signorino., "An introduction to machine learning methods for survey researchers," *Survey Practice*, vol. 11, 2018. [Online]. Available: https://doi.org/10.29115/SP-2018-0004

[29] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[31] H. Ye, L. Liang, G. Y. Li, J. Kim, L. Lu, and M. Wu, "Machine learning for vehicular networks: Recent advances and application examples," *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 94–101, June 2018.

[32] K. Pearson, "VII. note on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, vol. 58, no. 347-352, pp. 240–242, Jan. 1895. [Online]. Available: https://doi.org/10.1098/rspl.1895.0041

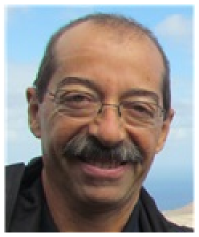
[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[34] D. Ferreira, C. Senna, and S. Sargento, "Distributed real-time forecasting framework for iot network and service management," in *Fifth IEEE/IFIP International Workshop on Analytics for Network and Service Management (AnNet 2020), IEEE/IFIP Network Operations and Management Symposium (NOMS)*, April 2020.

[35] ETSI, "3gpp ts 123 501 v15.9.0 (2020-03) – 5g; system architecture for the 5g system (5gs) (3gpp ts 23.501 version 15.9.0 release 15)," Tech. Rep., 2020.

Diogo Ferreira is currently working as Data Scientist at Talkdesk. He received the M.Sc. in Computers and Telematics Engineering from University of Aveiro, Portugal (2019). Over the last few years, he was involved with research in the area of 5G networks and Machine Learning. His main interests are intelligent networks, machine learning and deep learning.

Andre Braga Reis received the B.S. and M.Sc. degrees in electronics and telecommunications engineering from the University of Aveiro, Portugal, in 2009, in collaboration with the Eindhoven University of Technology, The Netherlands, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA. His research interests are in vehicular, ad-hoc, and mesh networks.



Dr. Carlos Senna is a Post-Doctoral Researcher at Instituto de Telecomunicaçẽs (IT) in Aveiro, Portugal. He received the Ph.D. (2014) degrees in Computer Science from the University of Campinas, Brazil, and he is working in research related to computer networks, cloud/edge computing and distributed systems. Over the last years he was involved in several research projects (Horizon Project: A New Horizon to The Internet, Support for Orchestration of Resilient and Reliable Services In the Fog (SORTS), and Europe - Brazil Collaboration of Big Data Scientific Research Through Cloud-Centric Applications (EUBRA BIGSEA)). Currently, he is involved in projects such as S2MovingCity: Sensing and Serving a Moving City, P2020 SAICT- PAC/0011/2015 MobiWise: from Mobile Sensing to Mobility Advising, 5G Mobilizer project "Components and services for 5G networks, and Efficient Information Centric Networks for IoT Infrastructure (InfoCent-IoT)". His main research interests are in ad-hoc and vehicular network mechanisms and protocols, network management, and cloud/edge/fog based solutions.

Susana Sargento is a Full Professor in the University of Aveiro and a senior researcher in the Institute of Telecommunications, where she is leading the Network Architectures and Protocols (NAP) group (https://www.it.pt/Groups/Index/62). She received her PhD in 2003 in Electrical Engineering in the University of Aveiro, being a visiting student at Rice University in 2000 and 2001. She joined the Department of Computer Science of the University of Porto between 2002 and 2004, and she was a Guest Faculty of the Department of Electrical and Computer Engineering from Carnegie Mellon University, USA, in August 2008, where she performed Faculty Exchange in 2010/2011. Since 2002 she has been leading many national and international projects, and worked closely with telecom operators and OEMs. She has been involved in several FP7 projects (4WARD, Euro-NF, C-Cast, WIP, Daidalos, C-Mobile), EU Coordinated Support Action 2012-316296 ”FUTURE-CITIES”, EU Horizon 2020 5GinFire, national projects, and CMU-Portugal projects (S2MovingCity, DRIVE-IN with the Carnegie Melon University) and MIT-Portugal Snob5G project. She has been TPC-Chair and organized several international conferences and workshops, such as ACM MobiCom, IEEE Globecom and IEEE ICC. She has also been a reviewer of numerous international conferences and journals, such as IEEE Wireless Communications, IEEE Networks, IEEE Communications. She was the founder of Veniam (www.veniam.com), which builds a seamless low-cost vehicle-based internet infrastructure, and she was the winner of the 2016 EU Prize for Women Innovators. Susana is also the co-coordinator of the national initiative of digital competences in the research axis (INCoDe.2030, http://www.incode2030.gov.pt/), belongs to the evaluation committee of the Fundo200M (www.200m.pt) government co-investment and funding), and she is one of the Scientific Directors of CMU-Portugal Programme (http://www.cmuportugal.org/). Her main research interests are in the areas of self-organized networks, in ad-hoc and vehicular network mechanisms and protocols, such as routing, mobility, security and delay-tolerant mechanisms, resource management, and content distribution networks. She regularly acts as an Expert for European Research Programmes.