A Learning-Based Fast Uplink Grant for Massive IoT via Support Vector Machines and Long Short-Term Memory

Eslam Eldeeb, Mohammad Shehab, and Hirley Alves, *Member, IEEE*

Abstract—The current random access (RA) allocation techniques suffer from congestion and high signaling overhead while serving massive machine-type communication (mMTC) applications. To this end, third-generation partnership project introduced the need to use fast uplink grant (FUG) allocation in order to reduce latency and increase reliability for smart Internet of Things (IoT) applications with strict Quality-of-Service constraints. We propose a novel FUG allocation based on support vector machine (SVM). First, machine-type communication (MTC) devices are prioritized using an SVM classifier. Second, a long short-term memory architecture is used for traffic prediction and correction techniques to overcome prediction errors. Both results are used to achieve an efficient resource scheduler in terms of the average latency and total throughput. A coupled Markov modulated Poisson process (CMMPP) traffic model with mixed alarm and regular traffic is applied to compare the proposed FUG allocation to other existing allocation techniques. In addition, an extended traffic model-based CMMPP is used to evaluate the proposed algorithm in a more dense network. We test the proposed scheme using real-time measurement data collected from the Numenta anomaly benchmark (NAB) database. Our simulation results show the proposed model outperforms the existing RA allocation schemes by achieving the highest throughput and the lowest access delay of the order of 1 ms by achieving prediction accuracy of 98 % when serving the target massive and critical MTC applications with a limited number of resources.

Index Terms—Alarm traffic, fast uplink grant (FUG), Internet of Things (IoT), machine learning, machine-type communications, resource allocation, support vector machines (SVMs).

I. INTRODUCTION

CELLULAR communications have experienced a paradigm shift in recent years by introducing service modes dedicated to machine-type communication (MTC), namely, massive MTC (mMTC), and ultrareliable low latency communications (URLLCs) [1]. Many smart Internet of Things (IoT) applications, such as traffic control, autonomous vehicles, environmental monitoring, surveillance, and crowdsensing, are enabled by MTC [2], [3]. Due to the diversity among MTC scenarios, the Quality-of-Service (QoS) requirements vary. Therefore, understanding their heterogeneous traffic behavior becomes an essential part of any communication system [4]. In this context, traffic modeling aims to capture the behavior of the traffic using a probabilistic model that can be implemented easily and provide feasible means for efficient allocation of network resources.

According to [5], MTC traffic, which is representative of many IoT applications, is classified into three different classes: 1) periodic update (PU), which describes periodic traffic characterized by a small number of short packets; 2) event driven (ED), which describes nonperiodic traffic due to a certain random trigger at an unknown time; and 3) payload exchange (PE), which describes bursty traffic that usually comes after PU or ED traffic. Traffic models are classified into source traffic models and aggregated traffic models [6]. Source traffic models, which treat every machine-type communications device (MTD) as a single separate entity, are very accurate, though become extremely complex when modeling highly dense networks. Aggregated traffic models treat all MTDs within the network as one entity by simply accumulating all the traffic as one stream. When compared to the source traffic approach, the aggregated traffic models are less complex at cost of lower accuracy. Laner *et al.* [4] designed a coupled Markov modulated Poisson process (CMMPP) such that it captures the traffic behavior efficiently based on a master node, called a background process, that describes the event. Grigoreva *et al.* [7] showed that modeling CMMPP using multiple background processes is computationally expensive in time. They introduce the coupled Markovian arrival process (CMAP) model based on unicast and multicast distributions describing the regular traffic and traffic affected by events, respectively.

In the long-term evolution (LTE) systems, devices attempt to access the medium via random access (RA) procedures. Thus, to obtain a radio resource and transmit a packet, user equipment (UE) undergoes a four-handshake procedure. This procedure suffers from high signaling overhead, which causes longer delays that prevent achieving the URLLC QoS requirements [8]. It also fails to meet one of the most challenging requirements of smart IoT, namely, is real-time performance due to high latency [9]. Additionally, highly dense MTC deployments with hundreds of devices competing for meager resources will suffer from a large number of

preamble collisions, which cause long delays and even packets drop [10]. To handle such issues, many solutions have been implemented as complementary features to the existing RA resource allocation algorithm, but they fail to guarantee MTC demands. Other researchers proposed credible techniques such as grant-free (GF) transmission, where each device chooses randomly a resource to transmit its packets without requests [11]. Although GF solutions overcome the exchange of messages that causes signaling overhead problems, highly dense mMTC scenarios, where the number of devices is usually larger than the number of available resources, still suffer from a large number of collisions and resulting in longer delays.

Recently, learning-based solutions have gained more attention to solve RA existing problems. In this context, fast uplink grant (FUG) was first introduced with the 2015 third-generation partnership project (3GPP) technical report [12], and further discussed in [10]. It is a learning-based resource allocation technique, where the base station (BS) preemptively allocates resources to devices on a predictive basis. FUG reduces signaling overhead and completely removes collision. Introducing FUG in smart IoT applications allows for installing a large number of IoT devices within a limited number of frequency resources while satisfying their demands owing to efficient predictive algorithms [9], [13]. Moreover, as FUG reduces signaling overhead and eliminates collisions, it economizes the energy consumption of IoT devices.

II. RELATED WORKS AND CONTRIBUTIONS

In the following paragraphs, we discuss the recent works in the literature that address the resource allocation problem in IoT networks.

To address the high number of collisions in the GF allocation, Ali *et al.* [14] suggested a dynamic resource allocation scheme, which resolves the preamble collisions rather than avoiding them. Abbas *et al.* [15] presented a GF solution based on nonorthogonal multiple access (NOMA). Unfortunately, the proposed solutions suffer from undesired signaling overhead and collisions [16]. Shehab *et al.* [17] presented a traffic prediction framework for IoT devices, which are influenced by binary Markovian events. A distributed NOMA solution based on reinforcement learning is used in [18], while Ali *et al.* [19] presented a sleeping multiarmed bandits (MABs) FUG solution. However, they focus only on achieving optimal resource allocation based on the QoS requirements of each device by employing an existing source traffic predictor and capturing the traffic behavior efficiently.

Kwasinski *et al.* [20] discussed the use of reinforcement learning and Q-tables to perform resource allocation. Shah-Mohammadi and Kwasinski [21] proposed a multiagent deep reinforcement learning technique, which performs distributed joint multiresource allocation. The authors applied transfer learning to the proposed model to speed up the convergence compared to applying deep Q-networks without transfer learning. Despite the advantages of applying reinforcement learning algorithms in resource allocation problems, it has two major disadvantages: 1) RL algorithms do not use any prior data about the network and only rely on reward signals and 2) the convergence can be time consuming.

Our contribution relies on the CMMPP traffic model as in [4] to evaluate the performance of our proposed model compared to RA procedures. Different from [7], we introduce a model that handles multiple background processes, called the M-background processes CMMPP (M-CMMPP) model. This allows us to address congested network scenarios with a large number of active devices and a low number of resources without being computationally expensive. Furthermore, we propose a novel FUG algorithm based on three steps.

1) The first step is the device classification, whereby the BS prioritizes the devices according to their QoS requirements. To classify the devices, we employ a large-margin classifier based on support vector machines (SVMs), which search for the optimal decision boundary that is maximally far away from the closest points of each class in the training set [22].

2) The second step is the traffic prediction, which is typically a time-series problem. The BS monitors the activity behavior of each MTD and forecasts which devices are active or inactive in a given transmission instant. In this context, long short-term memory (LSTM) is a special kind of recurrent neural networks (RNNs), which comprises four artificial neural networks (ANNs) layers, instead of just one, and acts as gates along with some pointwise operations [23]. These gates are trained to learn what to use from short and long past and what to forget. We apply LSTM for traffic prediction in our proposed FUG model. Moreover, we discuss the rationale for using LSTM and how it can be applied in real time.

3) The last step is to exploit device classification and traffic prediction results to grant resources to devices that are worth. CMMPP and M-CMMPP models are introduced in the problem formulation, where we apply the proposed FUG algorithm and compare it to existing resource allocation techniques. Testing and simulation are performed using real-time measurement data collected from the numenta anomaly benchmark (NAB) database.

The remainder of this article is organized as follows. Section III presents the system model and problem formulation. In Section IV, we introduce the details of the proposed FUG solution. In Section V, we present the simulation results and discussion. Finally, conclusion is presented in Section VI.

III. PRELIMINARIES

Consider a cellular network composed of a set $\mathcal{D}$ of static MTDs served by one BS with a limited number of frequency resources. The total number of devices is $D = |\mathcal{D}|$, where $|.|$ returns the length of a vector, such that each device $d_i \in \mathcal{D}$ can be either: 1) *active* or 2) *silent*. When active, devices can transmit either data packets with lower priority or alarm packets with higher priority. Each device $d_i \in D$ has fixed coordinate locations x_i and y_i, which are known by the BS. In this work, for each one of the devices, we aim to first predict its state,
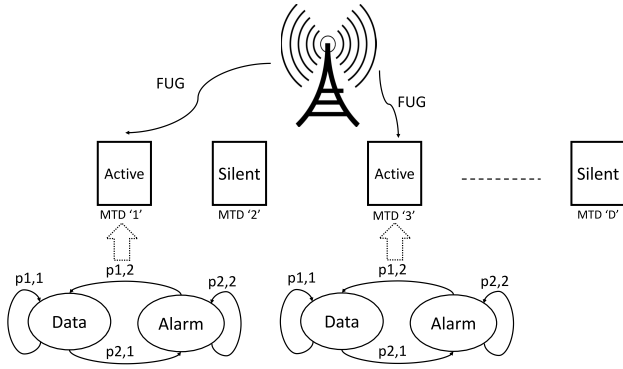
Fig. 1. Considered system model. A set of D devices, which can be active or silent. Active devices can be in either data or alarm state.

namely, silent or active, as well as the corresponding traffic priority. Thereafter, the serving BS schedules the set of active devices according to their priorities. We formulate the system model using an efficient traffic model called CMMPP [4]. Then, we extend the baseline model to account for a more dense scenario using the M-CMMPP model, which consists of several background CMMPP processes. Fig. 1 presents the system model.

A. CMMPP Traffic Model

The Markov processes and Poisson processes are very popular traffic and queuing models [24], [25]. Recently, the Markov modulated Poisson processes (MMPPs) have been developed for traffic modeling scenarios, where a Poisson process with a rate $\lambda_i[t]$ changes according to the state of Markov chains $s_n[t]$, where n is the number of MTDs. Consider a two-state MMPP, where the data state describes regular packet transmission, whereas the alarm state describes longer and higher priority packet transmission. Each device is transiting between these two states according to the respective probabilities based on the MMPP baseline model. Coupling Markov chains means that multiple chains mutually influence their transition probability matrices [4]. Given the state transition probabilities, we build a state transition matrix P_s, while the state probability vector π is defined using the respective state probabilities as follows:

$$P_s = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,j} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ p_{i,1} & p_{i,2} & \cdots & p_{i,j} \end{pmatrix}, \quad \pi = \begin{pmatrix} \pi_1 \\ \pi_2 \\ \vdots \end{pmatrix}. \quad (1)$$

In addition, the MMPP-based source traffic can be simplified by considering only one background process to modulate all MTDs in case of a sudden event such as fire or high temperature. Thus, the background process influences all the MTDs in both space and time, but with different strengths according to their distances from the epicenter (position of the background process) and time of that background process. Moreover, it causes some devices to transition from state 1 (data) to state 2 (alarm), while others remain in their current state. The state probability matrix is composed of two submatrices, namely, the coordinated matrix P_C and the uncoordinated matrix P_U.

The former describes the behavior of the devices near to the event and its main characteristic is to issue an alarm at a time (alarm state), and then go back to the data state again. On the other hand, the latter describes the behavior of the devices away from the background process and its main characteristic is to remain at the data state and never switch to the alarm state. The background process generates samples $\theta_n[t]$, which are a function of time and space for all devices. The state probability matrix is calculated as follows:

$$P_n[t] = \theta_n[t].P_C + (1 - \theta_n[t]).P_U \quad (2)$$

$$\theta_n[t] = \delta_n.\theta[t] \quad (3)$$

where $\theta[t] \in [0, 1]$ consists of samples, uniformly distributed over time, which describe how the devices are affected by the background process in time. δ_n follows standard normal distribution whose samples describe how the devices are affected by the background process in space. In addition, the mean and variance of δ_n describe the epicenter of the background process and how strong is the background process (high variance, means strong event, and further devices are affected), respectively. Devices near the epicenter of the background process are highly affected by the process and transit to state 2 (alarm). Modeling δ_n as normal distribution describes the behavior of sudden events (for example, fire or high temperature) in real-time applications, where devices near to the center of the event have higher probabilities to be affected by that event than far devices. Multiplying $\theta[t]$ and δ_n results in $\theta_n[t]$, which describes how the devices are affected by the background process jointly in space and time. The same idea is repeated with the state probability vector $\pi_n[t]$, which will be composed of π_C and π_U. Therefore, θ and δ_n are distributed as $\theta \sim \mathcal{U}(T, T + \tau)$ and $\delta_n \sim \mathrm{N}(\mu, \Sigma)$, where $\mathcal{U}$ is the uniform distribution, N is the normal distribution, T is the start instant of the background process, τ is the duration of the background process, while μ and Σ represent the mean and the covariance matrix of the background process in x and y coordinates, respectively. Space and time distributions are independent. Therefore, the probability density function (PDF) of $\theta_n[t]$ is defined in (4), where $x, y \in \mathbb{R}_0^+$

$$f_{\theta_n}(x, y, t) = \begin{cases} \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{1}{2}\left[\left(\frac{x-\mu_x}{\sigma_x}\right)^2 + \left(\frac{y-\mu_y}{\sigma_y}\right)^2\right]\right) \\ \qquad\qquad , T < t < T + \tau; \\ 0, \quad \text{otherwise.} \end{cases} \quad (4)$$

Simulating the CMMPP traffic model using the derived equations results in four plots describing the behavior of MTDs in space and time as shown in Fig. 2. Fig. 2 shows: 1) the startup state where all the devices are likely to transmit data; 2) the data state in which some devices transmit data at different instants and using different packet lengths; 3) the alarm state where devices near to the epicenter transmit alarm signal in a correlated behavior with large packet lengths due to the activation of the background process; and finally, 4) the silent state where the devices that transmitted alarm tend to be silent [4].
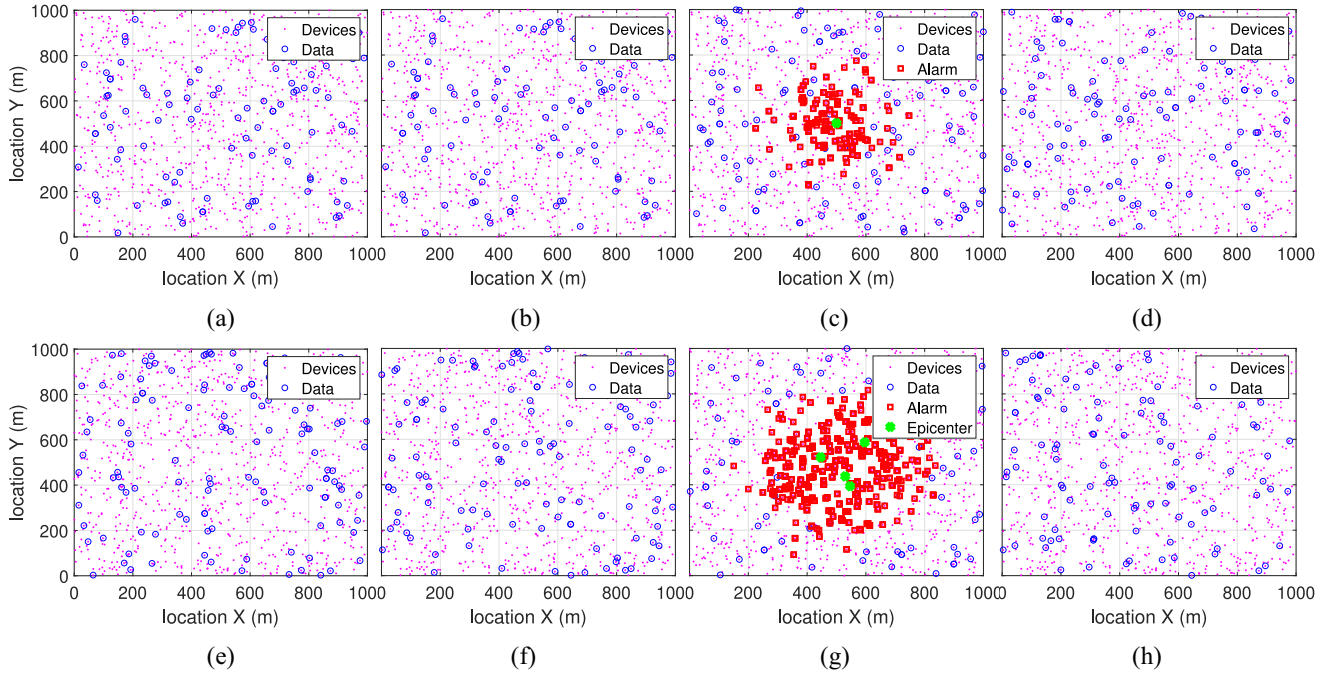
Fig. 2. Traffic modeling results: 1000 MTD, 60-s runtime, 1000 m × 1000 m area, λ_{Data} = 1/3600, λ_{Alarm} = 1, and uniformly distributed processes epicenters, intervals, and variances. CMMPP results: (a)–(d). M-CMMPP results (M = 4): (e)–(h). (a) Startup. (b) Data. (c) Alarm. (d) Silent. (e) Startup. (f) Data. (g) Alarm. (h) Silent.

B. M-CMMPP Traffic Model

We introduced the CMMPP model based on one background process to represent a regular MTC use case. Some applications can have more than one background process at a time causing a bursty scenario and network congestion [7]. Assume $M \geq 1$ background processes affect the transition of devices from the data state to the alarm state as follows:

$$\theta_{n_m}[t] = \delta_{n_m}\theta_m[t], \quad m = 1, 2, 3, \ldots, M. \tag{5}$$

As the background processes are independent, we start by calculating the probability of no alarms (P_{NA}), which is a function of the probability of alarm at a given process (P_{m_A})

$$P_{NA} = \left(1 - P_{1_A}\right)\left(1 - P_{2_A}\right)\cdots\left(1 - P_{M_A}\right)$$
$$= \prod_{m=1}^{M}\left(1 - P_{m_A}\right), \quad m = 1, 2, 3, \ldots, M \tag{6}$$

$$P_A = 1 - P_{NA} = 1 - \prod_{m=1}^{M}\left(1 - P_{m_A}\right), \quad m = 1, 2, 3, \ldots, M. \tag{7}$$

The overall $\theta_n[t]$, which is the probability of having alarms due to M background processes, can be derived as follows:

$$\theta_n[t] = 1 - \prod_{m=1}^{M}\left(1 - \left(\delta_{n_m}.\theta_m[t]\right)\right). \tag{8}$$

Then, the overall $\theta_n[t]$ from (8) is used in (2) to form the transition probability matrix. Therefore, θ_m and δ_{n_m} are distributed, respectively, as $\theta_m \sim \mathcal{U}(T_m, T_m + \tau_m)$ and $\delta_{n_m} \sim N(\mu_{\mathbf{m}}, \Sigma_{\mathbf{m}})$, where $\mathcal{U}$ is the uniform distribution, N is the normal distribution, $\mu_{\mathbf{m}}$, T_m is the start instant of the background process m, τ_m is the duration of the background process m, and $\mu_{\mathbf{m}}$ and $\Sigma_{\mathbf{m}}$ represent the mean and the covariance matrix of the background process m in x and y coordinates, respectively. Space and time distributions are independent. Therefore, the overall $\theta_n[t]$ is defined as in (9), shown at the bottom of the page, where $x, y \in \mathbb{R}_0^+$, and $m = 1, 2, \ldots, M$.

Simulating the proposed M-CMMPP traffic model using 4-background processes shows similar results to the original CMMPP model, but with an extremely larger number of data and alarm packets as shown in Fig. 2(e)–(h). This can cause network congestion, which is one of the most challenging problems in MTC applications while using current cellular network technologies.

$$f_{\theta_n}(x, y, t) = \begin{cases} \dfrac{1}{\tau_m}\left[1 - \left(\Pi_{n=1}^{M}\left(1 - \dfrac{\exp\left(-\dfrac{1}{2}\left[\left(\dfrac{x-\mu_{n_x}}{\sigma_{n_x}}\right)^2 + \left(\dfrac{y-\mu_{n_y}}{\sigma_{n_y}}\right)^2\right]\right)}{2\pi\sigma_{n_x}\sigma_{n_y}}\right)\right)\right], & T_m < t < T_m + \tau_m \\ 0, \text{otherwise} \end{cases} \tag{9}$$

IV. PROPOSED FAST UPLINK GRANT ALGORITHM

The proposed FUG model is divided into three main parts: 1) *device classification*; 2) *traffic prediction*; and 3) *resource allocation*. First, the BS predicts which devices, within a certain application under its coverage area, have a higher priority compared to other devices and need to be served first. The second step is estimating the time, which each device needs to access the network. Finally, the BS uses the results from the device classification and traffic prediction to allocate resources to the selected devices at specific time instants. The data collection, preprocessing, and computations done on it are performed on a server on the BS end, where it has better computational and energy resources that can limit the latency of the computations.

A. Device Classification Using SVM

According to the CMMPP model, the serving BS classifies devices into two groups. While devices in the first group transmit only data all the time, the second group transmit data and alarm. Devices that transmit alarms have higher priority than devices that transmit data. Specifically, our first step is a typical binary classification problem. Binary classification is a type of supervised learning, where a model is developed to classify some candidates among two classes. The first step in device classification is collecting an appropriate data set with desired features and labels, which reflects different scenarios of the application for a period of time. Features are the input data that the classifier should take into consideration and learn their pattern and how they are related to their corresponding labels. In MTDs classification, features are position coordinates of the devices, while labels are the data or alarm classes. Usually, binary classification problems are considered as finding the best decision boundary that separates the classes correctly.

The optimum decision boundary can be found using the SVM algorithm. Define training points $T = (\vec{x}_i, z_i)$, where $\vec{x}_i$ are the features, and z_i are the labels $(-1, 1)$. Then, define the decision hyperplane as b and a normal vector $\vec{w}$ perpendicular to the hyperplane. Finally, define a point $\vec{x}$ to be on the hyperplane, so that

$$\vec{w}^T \vec{x} = -b. \tag{10}$$

The SVM aims to maximize the width between the nearest features from one class to the other (support vectors of each class) [22]. Fortunately, this optimization problem is convex, which can be solved using the Lagrange multiplier theorem with any quadratic programming

$$L = \frac{1}{2}\|w\|^2 - \sum \alpha_i \big[z_i(\vec{w}.\vec{x}_i + b) - 1\big] \tag{11}$$

where α_i is the Lagrange multiplier. Classify class 1 if

$$\sum \alpha_i.y_i.\vec{x}_i.\vec{x} + b \leq 0. \tag{12}$$

By inserting features (device coordinates) and labels (class 1 or class 2) into the classifier, we find the optimal boundary between the two classes. However, this classifier is only applicable to linearly separable feature points. The CMMPP classes are nonlinearly separable. Thus, to avoid this problem, we use transformation kernels (ϕ) to map the feature points into higher dimensions, where they can be linearly separable

$$\phi : \vec{x} \rightarrow \phi(\vec{x}). \tag{13}$$

The radial basis function (RBF) maps the data into an infinite-dimensional Hilbert space [22]. It is very simple and fast to remap feature points using such kernel transformations; here, we employ the Gaussian RBF kernel given as

$$K\big(\vec{x}, \vec{x'}\big) = e^{-\frac{(\vec{x}-\vec{x'})^2}{2\sigma^2}} \tag{14}$$

where $\vec{x}$ and $\vec{x'}$ are two features. The RBF SVM can not only extract the pattern of devices efficiently but also classify them according to their priorities.

B. Traffic Prediction Using LSTM

After predicting the priorities of the devices, the serving BS needs to predict the activation time and silent time of these devices to implement an efficient resource scheduler. In a simple RNN architecture, input features, which are the past status of MTDs collected by the BS, are updated each instant t and fed into an ordinary ANN, where hidden layers are connected to form a feedback path. In such RNNs, there are three types of weights, which should describe the dependency between instants: 1) w_{xh} represents weights from input features to hidden layers; 2) w_{hh} corresponds to weights from hidden layers at time instant $t - 1$ to hidden layers at time instant t; and 3) w_{hy} yields the weights from hidden layers to output. Those weights are used in the underlying prediction as follows:

$$h_t = \tanh(w_{hh}h_{t-1} + w_{xh}x_t) \tag{15}$$
$$y_t = w_{hy}h_t \tag{16}$$

where $\tanh(\cdot)$ is the hyperbolic tangent activation function, h_{t-1} is the previous hidden layer at $t-1$ result from the same recurrent equation, and x_t is the input features vector at t. Afterward, a loss function and an optimizer are applied to find the correct weights that describe the relation between inputs and outputs [26].

Despite their astonishing ability to forecast the future, the RNNs still have a major weakness, namely, they have difficulty extracting relevant information located in the far past. Long sequences cause a major problem known as vanish gradient, where the relevant information is located far away in the past experiences almost zero gradient [27]. To solve the problem of long-term dependencies, Hochreiter and Schmidhuber introduced their LSTM architecture in 1997 [23]. It uses the same concept of basic RNN, but with complex four-gate functions connecting past and current instants to extract relevant information from long and short memories. As shown in Fig. 3, the LSTM has two inputs at each instant the short-term and the long-term memories given by h_{t-1} and C_{t-1}, respectively. While the former yields the previous hidden layer just as in RNN, the latter allows the LSTM to learn what to add and what to ignore from the very long past to keep only relevant information for prediction. The LSTM can be classified
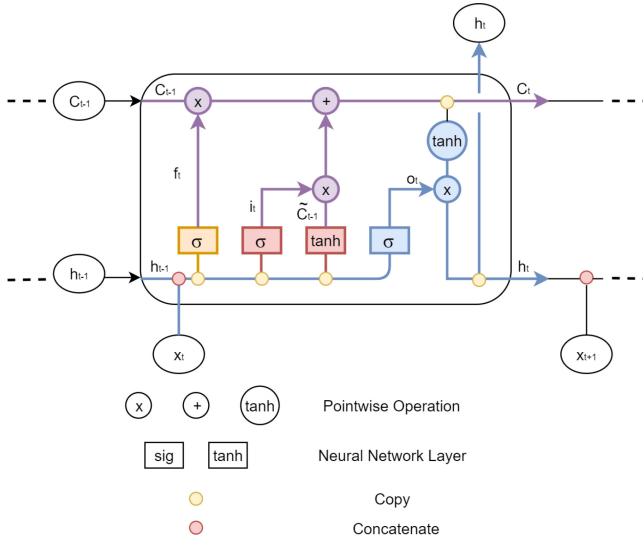
Fig. 3.   Long short-term memory architecture.

Fig. 4.   Visualizing prediction of MTD status in real time. Time axis is divided into windows, which are composed of training and prediction phases.

into four main gates: 1) *forget gate*, which describes accurately what information to forget from both input features and previous hidden state; 2) *learn gate*, which is responsible for learning new features related to the model; 3) *remember gate*, where the long-term memory is updated, C_t, using the results of the forget gate and the learn gate; and 4) *use gate*, where the current short-term memory is updated. The LSTM key equations are presented as follows:

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ \tilde{C}_t^l \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \tag{17}$$

$$C_t^l = f_t \odot C_{t-1}^l + i_t \odot \tilde{C}_t^l \tag{18}$$

$$h_t^l = o_t \odot \tanh\left(C_t^l\right) \tag{19}$$

where i_t is the output of the learn gate, f_t is the output of the forget gate, o_t is the output of the use gate, $\tilde{C}_t^l$ is the initial long term memory vector, σ is the sigmoid function, W^l is the weighs vector, h_t^{l-1} is the current initial hidden layer vector, h_{t-1}^l is the previous hidden layer vector, C_t^l is the current updated long-term memory, $\odot$ is a pointwise multiplication, and h_t^l is the current updated hidden layer vector. To address the computation efficiency of the LSTM architecture, we formulate the spatial complexity of each LSTM layer in terms of the number of parameters (K) as follows [28]:

$$K = 4\,(I + 1)\,O + O^2 \tag{20}$$

where I is the size of the input vector to an LSTM layer, and O is the size of the output vector of an LSTM layer.

The serving BS needs to collect a relevant data set, which contains the status of each MTD for a period of time. This data set is then trained with an LSTM model to predict the status of the MTDs in the future for a certain period of time. The training and prediction phases should be continuously alternated. We divide the time axis into windows, each window consists of two phases: 1) training phase and 2) prediction and correction phase. 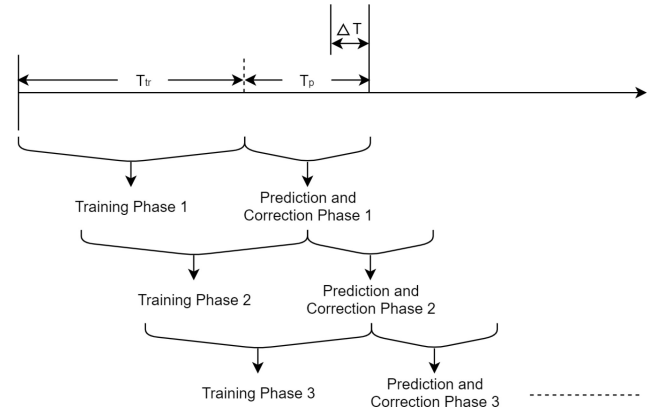In the training phase, the BS uses the collected data set about each MTD activity for a certain period of time T_{tr} to use it with the LSTM model to predict its activity in the following T_p. The prediction is subject to a degree of accuracy and therefore, prone to errors. Particularly, in this context, there are two types of errors: 1) the device is silent and it is predicted as active and 2) the device is active and it is predicted as silent.

Assume the BS has feedback, which we will discuss later, that senses these errors and corrects them. Define ΔT as the correction time. The serving BS should correct errors that exist in the first $T_p - \Delta T$ period of the predicted samples. Afterward, a new training phase starts by shifting the training window by $T_p - \Delta T$, where the BS uses the corrected prediction samples concatenated with the last $T_{tr} - (T_p - \Delta T)$ period of the previous training period to train new samples using the same model. Then, a prediction will be performed for the next T_p with correction of errors for first the $T_p - \Delta T$ interval of predicted samples. The process repeats so that we have error-free data at every training phase. The proposed algorithm is illustrated in Fig. 4.

C. Resource Allocation

As the serving BS classifies the type of each MTD and predicts their traffic, the next step is to schedule the resources for these devices. The resource allocation algorithm is illustrated in Fig. 5. The BS predicts the active devices at each instant, and then classifies their priorities and grants them the needed resources with given order according to their priorities. Devices from the same class are scheduled as first-come–first-served (FCFS). In addition, the BS performs some error correction techniques to perform an accurate prediction without accumulation of errors at each phase. The serving BS implements different procedures to carry out feedback and error correction as follows.

1) The serving BS avoids eventual prediction errors by adding safety margins. The length of T_m is chosen according to an exploration rate[1] α.

---

[1]The exploration rate α controls the margin time T_m, available resources for RA, and available resources for random allocation. Adjusting different exploration rates is out of the scope of this work; therefore, we arbitrarily set these parameters based on experimentation.
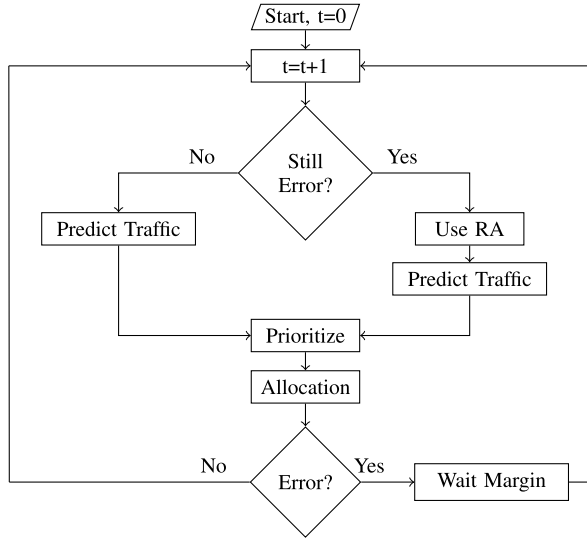
Fig. 5.   Fast uplink grant resource allocation algorithm.

Algorithm 1: Fast Uplink Grant Resource Allocation Algorithm

1 Define T_{tr}.
2 Define T_p.
3 Define ΔT.
4 Define exploration rate α.
5 while *True* **do**
6   Run device classification.
7   Run traffic prediction.
8   Prioritize active devices.
9   Allocate resources.
10   **if** *error* **then**
11   Wait T_m.
12   **if** *still error* **then**
13   Use RA.
14   **end**
15   **end**
16   Explore random devices.
17   Correct errors in T_p - ΔT.
18   Create new error-free training sequence.
19 end

---

2) When the MTD is predicted to be active, while it is silent, the BS senses that the allocated resource has not been used. Afterward, the status of this device is changed from active to silent in the data set to avoid wrong future predictions.

3) When the MTD is predicted to be silent, while it is active, the device should wait for a period of T_m to get a resource. If it does not get a resource, it should communicate with the BS using the RA procedure. Hence, the BS should dedicate some resources for the RA procedure to be used in the case of prediction errors. The percentage a of dedicated RA resources is also adjusted according to the exploration rate α.

4) The BS explores random devices, other than those that it has predicted, using available unused resources randomly according to α.

Algorithm 1 summarizes the proposed FUG resource allocation learning algorithm including the discussed error correction techniques.

V. RESULTS AND DISCUSSION

In this section, we present the simulation results of the proposed FUG. First, we introduce the performance evaluation metrics. Afterward, the SVM classification results are presented and compared to other classifiers. Then, we present the LSTM prediction of sensors activity. Finally, we apply those results along with the error correction techniques to schedule resources to MTDs. The proposed FUG model is compared with grant-based RA (GB-RA), random FUG, where the serving BS randomly allocates resources to devices, and genie-aided FUG, where BS knows perfectly the traffic of each device and its priority. This comparison is done while adjusting the number of available frequency resources dedicated for 1000 devices in a 1000 m × 1000 m deployment area within 60 s. Moreover, the exploration rate α is set to 0.1, a = α, and $T_m = 20$ ms. The simulations are carried out on python using Keras library. In addition, we use a single NVIDIA Tesla

V100 GPU and 10 GB of RAM on a Linux operating system server dedicated for the researchers in the Center of Wireless Communications (CWC) at the University of Oulu to train the model.

A. Performance Evaluation Metrics

Both the device classification and traffic prediction algorithms are considered binary classification problems. Thus, there are many appropriate evaluation metrics, which are suitable for skewed data applications (rare alarms), such as [29] and [30].

1) *The confusion matrix* is the most important evaluation method, which illustrates the number of correct and wrong classifications in each class.

2) *The classification accuracy* describes the overall performance of the classifier.

3) In *the precision (P) and recall (R)*, the former describes the ratio of true predicted samples for each class to the total predicted samples of that class, while the latter describes the ratio of true predicted samples for each class to the total actual samples of that class.

4) *The F1-Score (f1s)* combines the P and R measurements via harmonic mean, resulting in a percentage near to the minimum between them

$$f1s = \frac{2RP}{R+P}. \tag{21}$$

We evaluate the performance of the network with respect to throughput. The transmission rate is calculated using $C = \log(1 + \mathsf{SNR}\,|h|^2)$, where SNR is the signal-to-noise ratio, and h is the given channel coefficient between an MTD and the BS. Each MTD will have a certain rate depending on the SNR and the channel condition between that device and the serving BS. In addition, for each transmission, different rates

TABLE I
CONFUSION MATRIX, ACCURACY, PRECISION, RECALL, AND f1-SCORE FOR CMMPP DEVICE CLASSIFICATION (SUPPORT: 544 DATA AND 113 ALARM) AND M-CMMPP DEVICE CLASSIFICATION (SUPPORT: 539 DATA AND 153 ALARM)

	Algorithm	Conf. Mat.		Acc.	$P\&R$		$f1 - Sc.$
CMMPP	Poly. SVM	354	190	0.90	0.98	0.65	0.78
		7	106		0.36	0.94	0.52
	RBF SVM	474	70	0.87	0.97	**0.87**	0.92
		14	99		0.59	**0.88**	0.70
	ANN	462	82	0.86	0.97	0.85	0.91
		12	101		0.55	0.89	0.68
	DT	478	66	0.87	0.97	0.88	0.92
		17	96		0.59	0.85	0.70
	RF	473	71	0.87	0.97	0.87	0.92
		15	98		0.58	0.87	0.70
M-CMMPP	Poly. SVM	539	0	0.85	0.78	1.00	0.88
		153	0		0.00	0.00	0.00
	RBF SVM	518	21	0.97	0.99	**0.96**	0.97
		5	148		0.88	**0.97**	0.92
	ANN	486	53	0.92	1.00	0.90	0.95
		2	151		0.74	1.00	0.85
	DT	518	21	0.96	0.98	0.96	0.97
		8	145		0.87	0.95	0.91
	RF	517	22	0.96	0.99	0.96	0.97
		6	147		0.87	0.96	0.91

exist depending on the transmission power of an MTD and channel condition at the time of transmission. As our main scope is to compare FUG to other allocation techniques, we assume the radio channel to be degraded by flat fading.

Communication systems have different sources of delay, such as hardware delay T_h, queuing delay T_q, and transmission delay T_t. The access delay T_a is the time difference between the moment an MTD is ready for transmission and the moment it receives a resource. It is a function of hardware delay, signaling overhead delay, and queuing delay originated from the existence of a lower number of resources compared to the number of ready devices at a time. In our simulation, we neglect the hardware delay and focus only on the access delay as a function of queuing delay and message exchange between MTDs and the BS. The access delay can be calculated as follows:

$$T_{\text{total}} = T_h + T_{\text{overhead}} + T_q + T_t \tag{22}$$

$$T_a = T_h + T_{\text{overhead}} + T_q = T_{\text{total}} - T_t. \tag{23}$$

B. Simulation Results

1) Device Classification: We initially collect the training set by running multiple CMMPP and M-CMMPP simulations with uniformly distributed epicenters, variances, and intervals of background processes. Then, we preprocess the data by balancing, normalizing, and then removing redundancies to ease the training phase and extract the correct features. Afterward, we compare different classification algorithms to a new CMMPP/M-CMMPP model as shown in Table I. We observe that a polynomial kernel (degree = 6) SVM provides good performance in the CMMPP case, but fails in the network congestion case. An ANN architecture with four hidden layers (16, 32, 8, and 4 neurons) works very well in both cases in terms of data and alarm classification, where it introduced the lowest errors (only 12 alarm errors in case of CMMPP and

2 alarm errors in case of M-CMMPP) in classifying alarms compared to all classifiers. However, it has a relatively large number of errors in classifying data devices.

The RBF SVM, decision trees (DTs), and random forests (RFs) produce better results than ANN in classifying data devices and almost similar results as ANN in classifying alarm devices. The RBF SVM is the simplest algorithm among them and works extremely fast. It produces a recall of 0.87 and 0.88 for data and alarm classification, respectively in the case of CMMPP traffic and a recall of 0.96 and 0.97 for data and alarm classification in the case of M-CMMPP traffic. These small number of errors reflect the strength of the RBF SVM classifier in prioritizing the devices. In addition, the RBF SVM needs less than 3 s of execution time for both training and prediction, which reflects its high efficiency. Hence, RBF SVM is the chosen algorithm to be used by the BS to classify devices.

2) Traffic Prediction: The NAB is a time-series data set, which contains 58 time-series data files designed to help researchers in time-series prediction and streaming anomaly detection applications [31]. This data set provides real-time data collected from sensors monitoring different physical quantities in industrial deployment scenarios. We do some preprocessing steps, where the sensors are active and need resources when their measurement exceeds a certain threshold. After preprocessing steps, two months of training data have been prepared to be used for the first training phase. The prediction and correction window is set as $T_p = 10$ min and $\Delta T = 5$ min for each iteration. To illustrate the performance of our scheme, we perform four prediction and correction iterations.[2]

We build up an LSTM architecture with two hidden layers (150 and 100 Neurons), 20% dropout, mean square error (MSE) loss function, 50 unrolling (create an array of 50 samples, then, for the next array, add one element and use the last 49 from the previous array), 50 epochs, and using the Adam optimizer [32]. In our predictions, this architecture produces overall accuracy of 95%, the predictor failed 11 times to correctly infer the sensor activity of a total of 171 activation instants, and it wrongly predicts 41 times that the sensor is active, while it is silent. Furthermore, it achieves f1-Scores of 0.98 for silent prediction and around 0.90 for active predictions. According to (20), the number of parameters in our architecture is 2 551 100, which is computationally efficient in terms of spatial efficiency as the larger the number of parameters, the better the architecture can extract relevant information [28]. In addition, each training phase needs around 200 s of execution time. We should point out that much deeper LSTM architecture would enhance the prediction accuracy as it may extract more relevant information, but it would increase complexity and time of training and prediction as well. In addition, the length of the training phase can be increased, at the expense of a long time to extract the pattern. Selecting the appropriate neural network architecture is a very challenging

---

[2]For ease of implementation, we perform only four iterations to predict 40 min of activation. Note that these prediction iterations could be extended deep into the future for hundreds or even thousands of iterations with almost the same performance, thanks to the feedback and error correction procedures that are presented in Section IV-C.
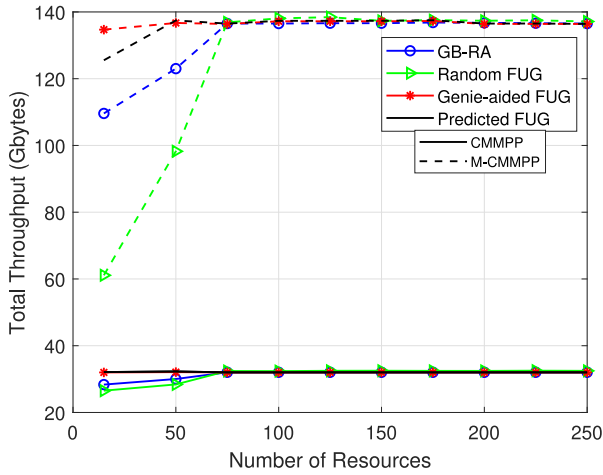
Fig. 6. Total throughput while adjusting the number of available resources: 60 s runtime, 1000 MTDs, $\alpha = 0.1$, $a = 0.1$, and $T_m = 20$ ms. Beyond 75 frequency resources, all schemes converge.
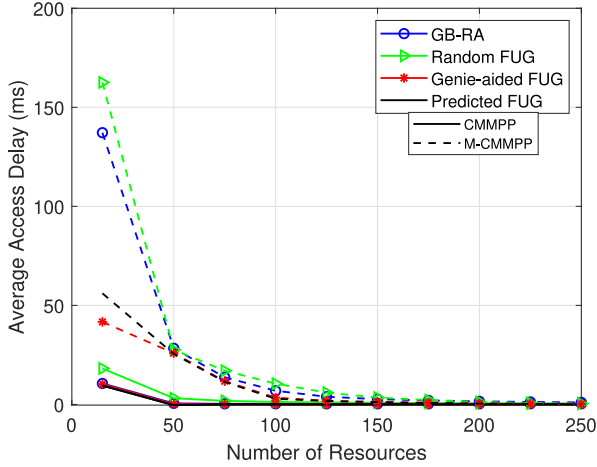

Fig. 7. Average access delay while adjusting the number of available resources: 1000 MTDs, $\alpha = 0.1$, $a = 0.1$, and $T_m = 20$ ms. Beyond 75 frequency resources, all schemes converge.

research problem, where large architectures consume a long time, whereas short ones may introduce lower accuracy [33]. The depth of the model and the length of the training data should be adjusted according to the available hardware at the BS, the criticality of the application, the number of devices, and the number of expected errors. The resulting errors are corrected, as mentioned, to have clean training data again to be used in the next phase.

3) Resource Allocation: Throughput and access delay are monitored for different allocation schemes while adjusting the number of available frequency resources at the BS. In Fig. 6, the throughput, which is the total successfully received packets, is plotted while adjusting the number of available resources during 60 s. Random FUG resource allocation has the worst performance, whereas the predicted FUG outperforms the GB-RA and almost achieves the genie-aided FUG for both CMMPP and M-CMMPP. We notice that as the number of available resources increases beyond 75 frequency resources, all schemes start to converge and perform well.

In Figs. 7 and 8, the average access delay and maximum access delay are plotted, respectively. The proposed predicted
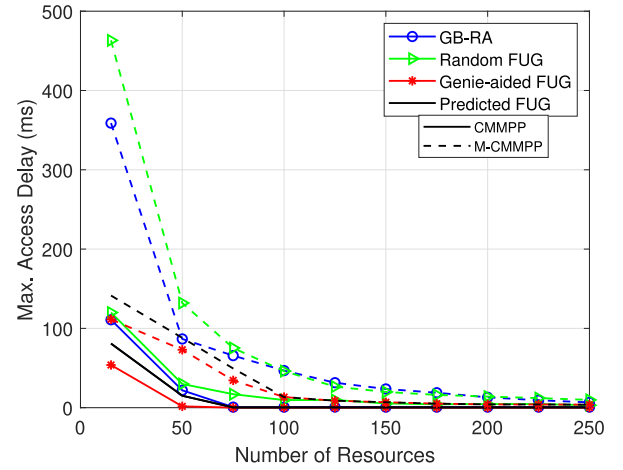

Fig. 8. Maximum access delay, while adjusting the number of available resources: 1000 MTDs, $\alpha = 0.1$, $a = 0.1$, and $T_m = 20$ ms. Beyond 75 frequency resources, all schemes converge.

FUG almost approaches the genie-aided FUG. Furthermore, the proposed FUG presents a free-collision resource allocation scheme, whereas GB-RA suffers from several preamble collisions. The random FUG is presented to mention the importance of having a traffic predictor with high accuracy. Having low traffic prediction accuracy can cause even worse performance than the RA schemes. For less than 50 frequency resources, we notice that the predicted FUG algorithm can achieve extremely low latency in the order of 1 to few milliseconds. For this low latency, at least a total of 120 Gbytes of packets are successfully delivered to the BS in the M-CMMPP scenario compared to 108 and 60 Gbytes for the GB-RA and random FUG, respectively as shown in Fig. 6. This evidences the outstanding performance of the proposed predictive FUG in the presence of a limited number of resources serving a massive IoT deployment, which is the case of interest.

VI. CONCLUSION

In this article, we have proposed a novel FUG resource allocation scheme based on SVM and LSTM. First, we have set up the CMMPP and M-CMMPP models to be our system models. Next, we have implemented an SVM algorithm to classify devices into different priorities. Then, we have developed an LSTM architecture to predict the traffic of a real-time MTD data set. Afterward, the device classification and traffic prediction were used to schedule the resources. Simulation results have shown that the proposed FUG outperforms RA schemes and almost approaches genie-aided FUG in terms of throughput and latency. We have presented different kinds of error correction techniques that are used to avoid error accumulation. In addition, we have shown the importance of having an accurate traffic predictor to avoid random allocation behavior. Applying learning-based solutions to choose the appropriate exploration rate α is an open research problem for future work.

REFERENCES

[1] N. H. Mahmood, H. Alves, O. A. López, M. Shehab, D. P. M. Osorio, and M. Latva-Aho, "Six key enablers for machine type communication in 6G," in *Proc. 6G Summit*, 2019, pp. 1–5.

[2] D. Minoli, K. Sohraby, and B. Occhiogrosso, "IoT considerations, requirements, and architectures for smart buildings-energy optimization and next-generation building management systems," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 269–283, Feb. 2017.

[3] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[4] M. Laner, P. Svoboda, N. Nikaein, and M. Rupp, "Traffic models for machine type communications," in *Proc. 10th Int. Symp. Wireless Commun. Syst. (ISWCS)*, 2013, pp. 1–5.

[5] N. Nikaein *et al.*, "Simple traffic modeling framework for machine type communication," in *Proc. 10th Int. Symp. Wireless Commun. Syst. (ISWCS)*, 2013, pp. 1–5.

[6] M. Centenaro and L. Vangelista, "A study on M2M traffic and its impact on cellular networks," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, 2015, pp. 154–159.

[7] E. Grigoreva, M. Laurer, M. Vilgelm, T. Gehrsitz, and W. Kellerer, "Coupled Markovian arrival process for automotive machine type communication traffic modeling," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2017, pp. 1–6.

[8] P. Popovski *et al.*, "Wireless access in ultra-reliable low-latency communication (URLLC)," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5783–5801, Aug. 2019.

[9] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[10] S. Ali, N. Rajatheva, and W. Saad, "Fast uplink grant for machine type communications: Challenges and opportunities," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 97–103, Mar. 2019.

[11] N. H. Mahmood, R. Abreu, R. Böhnke, M. Schubert, G. Berardinelli, and T. H. Jacobsen, "Uplink grant-free access solutions for URLLC services in 5G new radio," in *Proc. 16th Int. Symp. Wireless Commun. Syst. (ISWCS)*, 2019, pp. 607–612.

[12] *Study on Latency Reduction Techniques for LTE*, 3GPP Standard TS 36.881, 2015.

[13] L. Lyu, C. Chen, S. Zhu, and X. Guan, "5G enabled codesign of energy-efficient transmission and estimation for industrial IoT systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 6, pp. 2690–2704, Jun. 2018.

[14] M. S. Ali, E. Hossain, and D. I. Kim, "LTE/LTE-A random access for massive machine-type communications in smart cities," *IEEE Commun. Mag.*, vol. 55, no. 1, pp. 76–83, Jan. 2017.

[15] R. Abbas, M. Shirvanimoghaddam, Y. Li, and B. Vucetic, "A novel analytical framework for massive grant-free NOMA," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 2436–2449, Mar. 2019.

[16] S. K. Sharma and X. Wang, "Toward massive machine type communications in ultra-dense cellular iot networks: Current issues and machine learning-assisted solutions," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 426–471, 1st Quart., 2020.

[17] M. Shehab, A. K. Hagelskjær, A. E. Kalør, P. Popovski, and H. Alves, "Traffic prediction based fast uplink grant for massive IoT," in *Proc. IEEE 31st Annu. Int. Symp. Pers. Indoor Mobile Radio Commun.*, 2020, pp. 1–6.

[18] M. V. da Silva, R. D. Souza, H. Alves, and T. Abrão, "A NOMA-based *Q*-learning random access method for machine type communications," *IEEE Wireless Commun. Lett.*, vol. 9, no. 10, pp. 1720–1724, Oct. 2020.

[19] S. Ali, A. Ferdowsi, W. Saad, N. Rajatheva, and J. Haapola, "Sleeping multi-armed bandit learning for fast uplink grant allocation in machine type communications," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 5072–5086, Aug. 2020.

[20] A. Kwasinski, W. Wang, and F. S. Mohammadi, *Reinforcement Learning for Resource Allocation in Cognitive Radio Networks*. Hoboken, NJ, USA: Wiley, 2020, ch. 2, pp. 27–44. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119562306.ch2

[21] F. Shah-Mohammadi and A. Kwasinski, "Deep reinforcement learning approach to QoE-driven resource allocation for spectrum underlay in cognitive radio networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, 2018, pp. 1–6.

[22] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Dec. 1997.

[24] H. Heffes and D. Lucantoni, "A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance," *IEEE J. Sel. Areas Commun.*, vol. 4, no. 6, pp. 856–868, Sep. 1986.

[25] S. L. Miller and D. Childers, "Chapter 9—Markov processes," in *Probability and Random Processes*, 2nd ed., S. L. Miller and D. Childers, Eds. Boston, MA, USA: Academic, 2012, pp. 383–428.

[26] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Jul. 2011.

[27] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[28] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," 2015. [Online]. Available: arxiv.abs/1506.02078.

[29] A. Ng, *Machine Learning*. 2017.

[30] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, 2009.

[31] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms—The numenta anomaly benchmark," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA)*, 2015, pp. 38–44.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: arXiv:1412.6980

[33] D. Hunter, H. Yu, M. S. Pukish, III, J. Kolbusz, and B. M. Wilamowski, "Selection of proper neural network sizes and architectures—A comparative study," *IEEE Trans. Ind. Informat.*, vol. 8, no. 2, pp. 228–240, May 2012.

Eslam Eldeeb received the B.Sc. degree in electrical engineering from Alexandria University, Alexandria, Egypt, in 2019, and the M.Sc. degree from the University of Oulu, Oulu, Finland, in 2021, where he is currently pursuing the Ph.D. degree with the Centre for Wireless Communication.

He is actively working on massive connectivity and ultrareliable low-latency communication. His research interests are machine-type communication, machine learning, and deep learning for wireless communication networks.

Mohammad Shehab received the B.Sc. degree from Alexandria University, Alexandria, Egypt, in 2011, the first M.Sc. degree from Arab Academy, Cairo, Egypt, in 2014, and the second M.Sc. degree from the University of Oulu, Oulu, Finland, in 2017.

He worked as a TA with Alexandria University and the Arab Academy from 2012 to 2015. He has been resuming his doctoral studies with Oulu's Centre for Wireless Communications since October 2017 and working on future wireless networks with focus on energy efficiency in machine type communication. His work so far resulted in ten conference papers, five journal papers, and one patent.

Dr. Shehab won the Best Student Paper Award in ISWCS 2017, and the Nokia Foundation Award consecutively for 2018 and 2019.

Hirley Alves (Member, IEEE) received the Doctoral degree in Communication Engineering from University of Oulu, Oulu, Finland

He is an Assistant Professor and the Head of the Machine-Type Wireless Communications Group at the 6G Flagship, Centre for Wireless Communications, University of Oulu, Oulu, Finland. He is actively working on massive connectivity and ultrareliable low latency communications for future wireless networks, 5GB and 6G, full-duplex communications, and physical-layer security. He leads the URLLC activities for the 6G Flagship Program.

Dr. Alves has received several awards and has been the organizer, the chair, the TPC, and a tutorial lecturer for several renowned international conferences. He is the General Chair of the ISWCS'2019 and the General Co-Chair of the 1st 6G Summit, Levi 2019, and ISWCS 2021.