Randomization of Data Generation Times Improves Performance of Predictive IoT Networks

1st Mert Nakıip
Institute of Theoretical and Applied Informatics
Polish Academy of Sciences, IITIS-PAN
44-100 Gliwice, Poland
mnakip@iitis.pl

2nd Erol Gelenbe
Institute of Theoretical and Applied Informatics
Polish Academy of Sciences, IITIS-PAN
44-100 Gliwice, Poland, &
Lab. I3S, Université Côte d'Azur,
Grand Château, 06103 Nice Cecex 2, France
ORCID:0000-0001-9688-2201

***Abstract*—Input traffic from Internet of Things (IoT) devices is often both periodic and requires to be received by a given deadline. This can create congestion at instants of time when traffic flowing from multiple devices arrives at a shared input port or gateway, resulting in missed deadlines at the receiver. As a consequence, scheduling techniques such as the "Earliest Deadline First" (EDF) and "Priority based on Average Load" (PAL) are used to schedule the flow from different devices so as to try to satisfy the needs of the largest number of traffic flows in a timely fashion. In this paper, we propose the Randomization of flow Genaration Times (RGT) in order to smooth the total incoming traffic to the input port or gateway, on top of the use of EDF and PAL. We then evaluate the performance of RGT together with PAL and EDP, for traffic load with a varying number of up to 6400 IoT devices. Our simulation results show that RGT provides significantly better performance when added to EDF and PAL. Also, the additional computation required by RGT at each device can be quite small, suggesting that RGT is a very useful addition for improving the performance of IoT networks.**

***Index Terms*—Internet of Things (IoT), Scheduling, Massive Access Problem, Predictive networks**

I. Introduction

The number of devices on the Internet is constantly growing, and expected to attain 30 billion by the year 2023 [1]. In addition, while the IoT is the key enabler for "smart cities" and Machine-to-Machine (M2M) communications [2], it is expected that the majority of devices on the Internet will be machine-type devices [3] and the usage of the IoT in smart cities has expanded to a wide range [4][5].

Moreover, it is estimated that 52% of all the IoT will be comprised of devices that fall in the Massive IoT segment by the end of 2025 [6] where a large number of low-cost devices will exist within the coverage area of a single base station or gateway, causing access problems due to Physical Random Access Channel (PRACH) overload [3][7] also known as the "Massive Access Problem".

The Massive Access Problem has been addressed with reactive techniques [8], [9], [10], [11], [12], [13], [14], [15], [16], [3], [17], [18], [19], [20], [21], [22], [7], [23] for IoT traffic that is generated in a random fashion. On the other hand, some work [24] has shown that IoT traffic at the MAC-layer can be predicted with acceptable accuracy via certain machine-learning models for distinct IoT traffic classes.

Another trend of research [25], [26], [27], [28], [29], [30] has suggested proactive network solutions, and other research has used the predictability of traffic generation patterns of IoT devices to address the Massive Access Problem under the "predictive network" framework. To this end, a recent paper [31] proposed the Joint Forecasting-Scheduling (JFS) system and the Priority based on Average Load (PAL) scheduling algorithm which allocates available resources to IoT devices based on forecasting the traffic generation pattern of each device for the case of single frequency channels.

In addition, in order to improve the performance of JFS and make it more applicable to longer time windows, a Multi-Scale Algorithm was proposed [32], and in [33] optimal scheduling and a Multi-Channel version of PAL was suggested to extend JFS for multi-frequency channels. The results in [31], [32], [33] showed that predictive networks with JFS are a promising for the solution to the Massive Access Problem, and that scheduling heuristics are crucial when optimal scheduling policies cannot be implemented due to their high computational requirements in practical circumstances.

In this paper, we propose an additional heuristic that we call "Randomization of Generation Times" (RGT) in order to improve the performance and decrease the optimality gap of scheduling heuristics for predictive networks. To the best of our knowledge, RGT could be implemented at each device and it is a novel pre-processing method that can be applied to any scheduling heuristic under the JFS system. In particular, we apply RGT to PAL and to the Earliest Deadline First (EDF) algorithm.

Our simulation results show that RGT improves the performance of each scheduling algorithm significantly, even when forecasting is conducted with high error, as well as in the case of zero-error perfect forecasting. Furthermore, we provide and estimate of the computation time of RGT to be under 2.5 μs per IoT device, and iindicate that it increases linearly with the number of devices. Thus we can conclude that RGT is an important addition that can facilitate the practical usage of JFS.

The rest of this paper is organized as follows. Section II

provides background on the scheduling of IoT traffic as well as regarding the PAL and EDF algorithms. Section III proposes our RGT process. Section IV presents our results. Section V summarizes our main conclusions and sugget further research avenues.

II. Scheduling of IoT Traffic

We now explain the scheduling problem of the IoT device traffic in predictive networks, and start with the JFS system proposed in [31] which is comprised of the Forecasting and Scheduling modules. We shall note that we evaluate the performance of the system for perfect forecasting and for the increasing forecasting error in Section IV. Between the Forecasting and the Scheduling modules, we insert a module that operates our RGT process that we will now define.

First, we define each burst j of any device i by the triple (r_j, d_j, a_j) where r_j is the instant when it is generated, its deadline is d_j and the number of bits is forwards in a particular burtst is a_j. In addition, if c_j denotes the maximum number of bits that can be transmitted by burst j in a single MAC-layer slot, and p_j is the total number of slots p_j needed to transmit burst j, we have:

$$c_j = R_i.\tau_{\text{MAC}},\ p_j = \lceil \frac{a_j}{c_j} \rceil, \tag{1}$$

where R_i is the data rate of device i which generated burst j, and τ_{MAC} denotes the duration of a MAC-layer slot, while each burst j has a strict delay constraint $\Delta_j = d_j - r_j$. Furthermore, we let N denote the total number of devices in the coverage area of IoT gateway and $\mathcal{J}$ denote the set of bursts that are generated by those devices.

Next, in the JFS system, based on the output of the Forecasting module, the Scheduling module allocates the available resources to IoT devices for a given upcoming time interval, namely the scheduling window. That is, the inputs of the Scheduling module is the future traffic generation pattern and the output of that is the binary schedule matrix $\mathbf{S}$, where $\mathbf{S}(j,m) = 1$ if MAC-layer slot m is allocated for burst j and $\mathbf{S}(j,m) = 0$ otherwise. We also let u_j be a binary variable which equals 1 if burst j is successfully transmitted, and 0 otherwise.

Although a variety of scheduling algorithms can be used in the Scheduling module of JFS, in this paper, we only consider PAL and the EDF. Note that EDF is optimal [**?**], [34] in the sense that it will minimize the number of bursts whose deadline cannot be met. PAL aims to maximize the total number of bits in transmitted bursts, while EDF aims to maximize the total number of successfully transmitted bursts without considering the number of bits in each burst.

A. Priority based on Average Load (PAL)

The greedy scheduling algorithm PAL was proposed in [31] for resource allocation in predictive networks, and especially for the JSF system, and it schedules the bursts by prioritizing each burst j based on its average load over Δ_j.For each time slot m, we define the set of "active bursts":

$$\mathcal{J}_{\text{active}}[m] = \{j :\ r_j \le m \le d_j\}, \tag{2}$$

and the average load per burst:

$$\alpha_j = \frac{a_j}{\Delta_j}. \tag{3}$$

PAL schedules the bursts by prioritizing each burst j based on its average load over the duration Δ_j, and works as follows starting for $m = 1$:

- It computes the burst j^* with the largest average load:

$$j^* = \arg \max_{j \in \mathcal{J}_{\text{active}}[m]} \alpha_j\ . \tag{4}$$

- Then for j^*, PAL allocates the upcoming p_{j^*} slots starting with current slot m.
- Finally, it updates $m \leftarrow m + p_{j^*}$ and updates $\mathcal{J}_{\text{active}}[m]$ accordingly.

As the definition of PAL suggests it is a non-preemptive algorithm, which means that it completes the transmission of each burst j which has been started, i.e., when the transmission of a burst has started, PAL waits for the completion of the transmission before it schedules another burst.

B. Earliest Deadline First (EDF)

We use the Earliest Deadline First (EDF) algorithm for the resource allocation for IoT. In this paper, we use a non-preemptive EDF algorithm which is optimal to maximize the total number of successfully delivered bursts for our scheduling problem [35]. In short, the EDF algorithm works as follows: We first sort the bursts in $\mathcal{J}$ with respect to d_j's into vector J_{sorted}. Then, for each j in J_{sorted} starting with the first burst, if there are enough available slots to transmit j, EDF reserves the first p_j available slots between r_j and d_j.

C. Performance Metrics

Throughout this paper, in order to measure the network performance, we define the following metrics:

First, η denotes the cross-layer network throughput and is defined as

$$\eta \equiv \frac{\sum_{j \in \mathcal{J}} u_j a_j}{\sum_{j \in \mathcal{J}} a_j} \tag{5}$$

Second, ζ denotes the fraction of the bursts that are successfully delivered, which is defined as

$$\zeta \equiv \frac{\sum_{j \in \mathcal{J}} u_j}{|\mathcal{J}|} \tag{6}$$

Third, $\mathcal{E}$ denotes the transmit energy consumption per successfully delivered bit. In this paper, we assume that one unit of energy is consumed by an IoT device for each time slot at which the IoT device continues its transmission. Thus, $\mathcal{E}$ is defined as

$$\mathcal{E} \equiv \frac{\sum_{j \in \mathcal{J}} u_j p_j}{\sum_{j \in \mathcal{J}} u_j a_j} \tag{7}$$

III. Randomization of the Generation Times

Since the heuristic algorithms are fast, inexpensive with respect to the computational hardware requirements and able to achieve relatively high QoS, those are promising for the scheduling of the IoT traffic in the predictive networks. Thus, we now aim to improve the performance of the heuristic scheduling algorithms. To this end, in this section, we propose Randomization of the Generation Times (RGT) which is a preprocessing on the predicted IoT traffic for the heuristic scheduling algorithms. The RGT process relieves the system by distributing the traffic generations over the scheduling window with duration of T_{sch}. In this process, we update the generation time, r_j, of each burst j via uniformly distributed random offset value as

$$r_j^{new} \leftarrow r_j + U[\Delta_j - S_j] \tag{8}$$

where $U[\Delta_j - S_j]$ is the realization of the uniformly distributed random variable in the range $[0, \Delta_j - S_j]$. In addition, S_j is a *safety time* via which the RGT limits the upper bound of r_j^{new}. S_j may take value in the range $[0, \Delta_j]$, where $S_j = 0$ indicates the maximum randomization and $S_j = \Delta_j$ indicates that there is no randomization. In this range, in order to determine the value of S_j, we can use one of the following methods:

- (M1) We search for the best value of S_j; however, this method is expensive in computation time and not suitable for real-time implementation. On the other, we may set a loose upper bound for the performances of EDF and PAL algorithms under RGT.
- (M2) We estimate the value of S_j while aiming to achieve the close to the performance of the upper bound. Even though the performance of the scheduling under the estimation of S_j might be inferior to the upper bound, it is fast and cheap in computation, and is therefore suitable for real-time applications.

A. Search for S_j to Achieve the Performance Upper Bound

In this method, we aim to determine the best value of S_j so the upper bound for the scheduling performance under RGT. To this end, we search for the best value of the fraction γ of the range $[0, \Delta_j]$. That is, for each $j \in \mathcal{J}$, we first define $S_j = \gamma\Delta_j$, then search for γ in range $[0, 1]$ as follows:

1) Set $\gamma = 0$.
2) Set $S_j = \gamma\Delta_j \qquad \forall j \in \mathcal{J}$
3) Update r_j for each j according to (6)
4) Schedule IoT traffic for the updated r_j's
5) Compute the performance metric (e.g. η, ζ or $\mathcal{E}$) and save those in a vector
6) Update $\gamma \leftarrow \gamma + 0.05$
7) If $\gamma \leq 1$, return to step 2); else, continue.
8) Find the best value of γ which leads to the best values of S_j's to maximize performance metric

Note that in this search, we use each of η, ζ and $\mathcal{E}$ as the performance metric.

B. Estimation of S_j

We now aim to calculate the estimation of S_j based on the theory of the single server waiting line, M/M/1 queue, model. First, since our scheduling basically works on the required processing times of the bursts, we replace a single customer of M/M/1 queue model with a single required processing slot of a burst. Due to this replacement, in our system, the average number of serving at a slot, μ, equals 1.

Then, we let λ denote the average arrival rate of required processing slots and calculate that as

$$\lambda = \frac{\sum_{j \in \mathcal{J}} p_j}{T_{sch}/\tau_{MAC}} \tag{9}$$

We also know that the average waiting time spent in the system by a single processing slot, denoted by W, equals $1/(\mu - \lambda)$. When $\mu = 1$,

$$W = \frac{1}{1 - \lambda} \tag{10}$$

Furthermore, for each burst j, we can estimate the waiting time that might be spent by that burst. To this end, we scale the average waiting time per required processing slot, W, for the total required processing slot by burst j, p_j. That is, we calculate the estimation of the waiting time of burst j, denoted by W_j^{est}, as

$$\begin{aligned} W_j^{est} &= p_j W \\ &= \frac{p_j}{1 - \lambda} \end{aligned} \tag{11}$$

However, according to (7), if the network is highly loaded and the system resources are insufficient to transmit all bursts, λ will be greater than 1. Since $\lambda > 1$, W_j^{est} takes negative values, which is unrealistic and impossible in real-life. On the other hand, for the highly loaded networks, this issue is expected since (7) actually does not satisfy the assumption $\mu > \lambda$ of M/M/1 models. In order to prevent the case where $W_j^{est} < 0$, we are not able to increase the system resources but we may revise (9) as

$$W_j^{est} = \max(0, \frac{p_j}{1 - \lambda}) \tag{12}$$

Considering it is estimated that j might wait for W_j^{est} slots in the system (including p_j), there should be at least W_j^{est} slots within Δ_j to transmit j successfully as long as $W_j^{est} < \Delta_j$. Thus, S_j is defined as

$$S_j = \min(W_j^{est}, \Delta_j) \tag{13}$$

Note that, under the case where the offered network traffic is higher than the resources so $\lambda > 1$ and $S_j = 0$, RGT may randomly drop some of the bursts when $d_j - r_j^{new} < p_j$. This property of RGT will relieve the system.

IV. Results

In this section, we evaluate the performance of each of the R-PAL and R-EDF schedulers under the predictive network. To this end, we use the IoT dataset [**?**] which is described in [33], comprised of the bootstrapped traffic generation patterns of real IoT devices whose traffic output belongs to one of the following traffic classes: Fixed Bit Periodic, Fixed Bit Aperiodic, Variable Bit Periodic, and Variable Bit Aperiodic.

In addition, for each burst j of each bootstrapped device i, the deadline takes one of the following six values $\Delta_j \in \{0.5,\ 1,\ 2,\ 180,\ 600,\ 3600\ (in\ seconds)\}$.

On this dataset, we measure the performance of the scheduling techniques for $N = 12$ devices as well as for integer values in the range$[400 \leq N \leq 6400]$ with increments of 400 devices. For each N, we randomly select $N/4$ devices from each device traffic class. Thus in our simulations, the set of devices from each device class is composed of 25% of all devices that are simulated.

In addition, in order to increase the load of the system and evaluate the algorithms on a highly loaded network, for each device i in this dataset, we decreased the data rate R_i by 30% (i.e. $R_i \leftarrow 0.7R_i$). Furthermore, we set $\tau_{\text{MAC}} = 100$ ms, and performed the simulations for scheduling windows with a duration of 900 seconds.

A. Performance Comparison of Scheduling Algorithms under Perfect Forecasting

We now present the performance of each of the R-PAL and R-EDF algorithms and their comparison with PAL and EDF, and the upper bound of those with respect to each of η, ζ and $\mathcal{E}$.

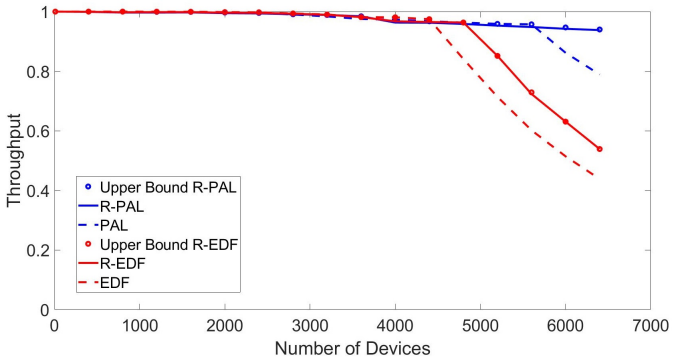

Fig. 1. Comparison of the Upper Bound R-PAL, R-PAL, PAL and the Upper Bound R-EDF, R-EDF, EDF algorithms with respect to throughput η for 12 to 6400 devices

In Figure 1, we see that the PAL-based scheduling algorithms significantly outperform the EDF-based algorithms for $N > 4800$ devices. The reason is that PAL aims to maximize the total number of bits in successfully delivered bursts while EDF aims to maximize the total number of successfully delivered bursts as described in Section II. We also see that both R-PAL and R-EDF aare able to achieve their upper bounds.

Furthermore, within the PAL-base algorithms in Figure 1, we see that the R-PAL outperforms the PAL for $N > 5400$ devices and the throughput difference between R-PAL and PAL increases with N. As explained in Section II-A, the PAL algorithm is a greedy algorithm that schedules the job with respect to the generation times. This is why the RGT process improves the performance of the PAL significantly (about 0.15 for $N = 6400$ devices). Within the EDF-base algorithms (Upper Bound R-EDF, R-EDF and EDF) in Figure 1, we see that the R-EDT outperforms the EDT for $N > 4400$ devices; that is, the RGT process significantly improves the performance of EDT for $N > 4400$.

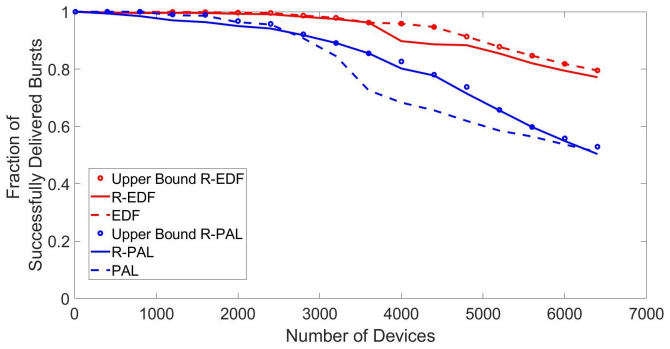

Fig. 2. Comparison of the Upper Bound R-PAL, R-PAL, PAL and the Upper Bound R-EDF, R-EDF, EDF algorithms with respect to the fraction of the successfully delivered bursts (ζ) for 12 to 6400 devices

In Figure 2, we see that the EDF-base algorithms are able to schedule more bursts for successful transmission than PAL-based algorithms. The reason is that the EDF aims to schedule maximum number of bursts without considering the number of bits that is carried by that burst. On the other hand, PAL aims to maximize the total delivered bits. This shows that the EDF is a more fair algorithm than PAL throughout the devices in the network since its values all burst equally. In this figure, we also see that the ζ performance of the R-EDF and R-PAL are comparable with EDF and PAL respectively.

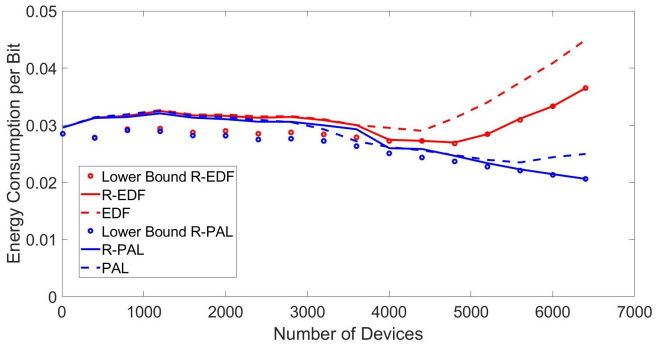

Fig. 3. Comparison of the Lower Bound R-PAL, R-PAL, PAL and the Lower Bound R-EDF, R-EDF, EDF algorithms with respect to the energy consumption per bit in delivered bursts ($\mathcal{E}$) for 12 to 6400 devices

In Figure 3, we see that each device consumed more energy to transmit a single bit under EDF-based algorithms than that

under PAL-based algorithms. The reason is that PAL aims to schedule bursts with a higher number of bits; in other words, it maximizes the denominator of (5). Moreover, our results in Figure 3 shows that the RGT process significantly decreases the energy consumption of both PAL and EDF algorithms.

B. Sensitivity of Scheduling Algorithms to Forecasting Error

We now aim to evaluate the performance of the scheduling algorithms for the increasing forecasting errors in the predictive network. To this end, we model the forecasting error as the realization of a random variable from the Normal Distribution[1]. That is, for each burst j, the forecast number of bits, $\hat{a}_j = a_j + n$, where n is the realization of the random variable from the Normal Distribution with zero mean and σ variance. In this subsection, we will analyze the network performance against the increasing value of σ which leads to the increasing forecasting error.

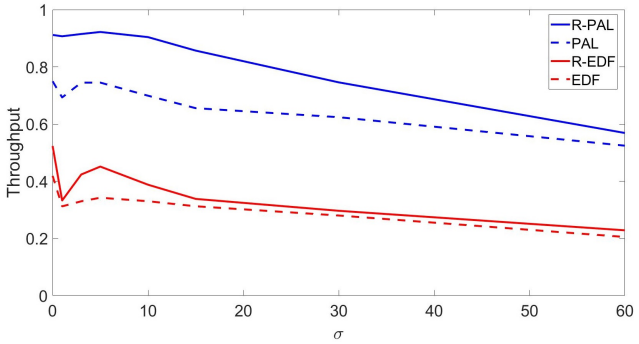

Fig. 4. Comparison of the R-PAL, PAL and R-EDF, EDF algorithms with respect to throughput (η) for $\sigma \in \{0, \ldots, 60\}$ for $N = 6400$ devices

In Figure 4, we present the network throughput η for the increasing value of σ so for the increasing forecasting error for $N = 6400$ devices. In this figure, we see that the RGT process significantly improves the network throughput performance of each of the PAL and EDF algorithm. That is, even while the forecasting error is very high compared to the maximum number of bits ($\sigma = 60$ and $\max_{j \in \mathcal{J}} a_j = 128$), RGT has impact on the scheduling performance. On the other hand, we see that the performance difference between R-PAL and PAL as well as the difference between R-EDF and EDF decreases as the forecasting error increases. In practice, since it is shown that the IoT traffic is predictable with acceptable forecasting error [24], the throughput performance of the algorithms at $\sigma = 20$ can be considered as their performance under an average performing forecaster (where the σ is higher than the 10% of the maximum generated bit over all devices).

C. Computation Time

We present the computation time of the RGT process for increasing number of devices. To this end, we measure the computation time of RGT on MATLAB on a Laptop with Intel Core i7-10750H cpu and 16 GB ram.

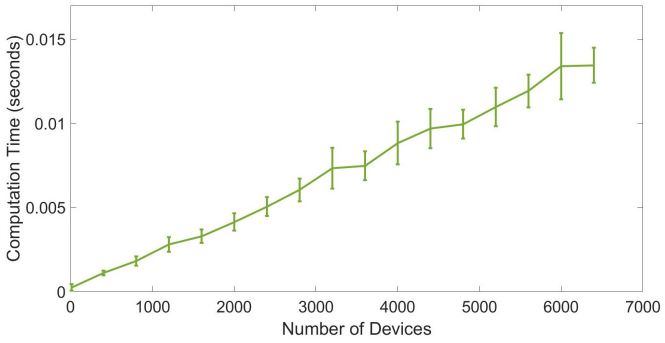

Fig. 5. The mean and the standard deviation of the computation time of the RGT process over 100 runs for 12 to 6400 devices

In Figure 5, we now present the mean of the computation time with a standard deviation bar for each N over 100 simulation runs. In this figure, we see that the computation time of RGT is under 0.015 seconds for all values of N and increases linearly with N. Thus, the computation time cost of RGT to scheduling algorithm is very small but the performance improvement is relatively high.

V. Conclusion

In this paper, we propose the Randomization of Release Times (RGT) process for the scheduling heuristics to allocate the resources in a predictive IoT network. The RGT can be implemented as a preprocessing algorithm on any scheduling heuristics to distribute the load of the network over a time window. In this paper, we evaluate the performance of the RGT process under each of the Priority based on Average Load (PAL) and Earliest Deadline First (EDF) algorithms for the IoT network with the increasing number of devices up to 6400 devices. Our results showed that the RGT process significantly improves the performance of both PAL and EDF heuristics in terms of the QoS metrics (throughput and energy consumption) while the fairness of each heuristic remains almost the same. In addition, we showed that the computation time of RGT is under 15 ms for 6400 devices and increases linearly with the number of devices. That is, the RGT is a fast and effective preprocessing algorithm that significantly improves the performance of the scheduling algorithms for IoT. Thus, since RGT enables to achieve much higher performances via fast heuristics, it will pave the way to include the devices with much lower delay constraint into predictive IoT networks.

[1]During the sensitivity analysis, in order to be able to control the forecasting error, we assume that an imaginary linear forecasting model minimizes the mean squared error and the distribution of the traffic generation pattern is Normal Distribution. Thus, we may model the forecasting error via Normal Distribution as well.

References

[1] Cisco, *Cisco Annual Internet Report (2018–2023)*, Mar. 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] M. Hasan, E. Hossain, and D. Niyato, "Random access for machine-to-machine communication in LTE-advanced networks: issues and approaches," *IEEE communications Magazine*, vol. 51, no. 6, pp. 86–93, 2013.

[3] F. Ghavimi and H.-H. Chen, "M2M communications in 3GPP LTE/LTE-A networks: Architectures, service requirements, challenges, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 525–549, 2015.

[4] O. Bello and S. Zeadally, "Toward efficient smartification of the Internet of Things (IoT) services," *Future Generation Computer Systems*, vol. 92, pp. 663–673, 2019.

[5] G. Fortino, W. Russo, C. Savaglio, M. Viroli, and M. Zhou, "Modeling opportunistic IoT services in open IoT ecosystems." in *WOA*, 2017, pp. 90–95.

[6] Ericsson, *Ericsson Mobility Report*, Nov. 2019. [Online]. Available: https://www.ericsson.com/en/mobility-report

[7] A. Zanella, M. Zorzi, A. F. dos Santos, P. Popovski, N. Pratas, C. Stefanovic, A. Dekorsy, C. Bockelmann, B. Busropan, and T. A. Norp, "M2M massive wireless access: Challenges, research issues, and ways forward," in *2013 IEEE Globecom Workshops*. IEEE, 2013, pp. 151–156.

[8] L. Liang, L. Xu, B. Cao, and Y. Jia, "A cluster-based congestion-mitigating access scheme for massive M2M communications in Internet of Things," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2200–2211, 2018.

[9] Z. Alavikia and A. Ghasemi, "Collision-aware resource access scheme for LTE-based machine-to-machine communications," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4683–4688, 2018.

[10] L. Tello-Oquendo, D. Pacheco-Paramo, V. Pla, and J. Martinez-Bauset, "Reinforcement learning-based ACB in LTE-A networks for handling massive M2M and H2H communications," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.

[11] N. Shahin, R. Ali, and Y.-T. Kim, "Hybrid slotted-CSMA/CA-TDMA for efficient massive registration of IoT devices," *IEEE Access*, vol. 6, pp. 18 366–18 382, 2018.

[12] L. Tello-Oquendo, I. Leyva-Mayorga, V. Pla, J. Martinez-Bauset, J.-R. Vidal, V. Casares-Giner, and L. Guijarro, "Performance analysis and optimal access class barring parameter configuration in LTE-A networks with massive M2M traffic," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 4, pp. 3505–3520, 2018.

[13] H. Jin, W. T. Toor, B. C. Jung, and J.-B. Seo, "Recursive pseudo-Bayesian access class barring for M2M communications in LTE systems," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 9, pp. 8595–8599, 2017.

[14] J. Liu, L. Song *et al.*, "A novel congestion reduction scheme for massive machine-to-machine communication," *IEEE Access*, vol. 5, pp. 18 765–18 777, 2017.

[15] M. Shirvanimoghaddam, M. Dohler, and S. J. Johnson, "Massive non-orthogonal multiple access for cellular IoT: Potentials and limitations," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 55–61, 2017.

[16] I. Park, D. Kim, and D. Har, "MAC achieving low latency and energy efficiency in hierarchical M2M networks with clustered nodes," *IEEE Sensors Journal*, vol. 15, no. 3, pp. 1657–1661, 2015.

[17] P. Si, J. Yang, S. Chen, and H. Xi, "Adaptive massive access management for QoS guarantees in M2M communications," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 7, pp. 3152–3166, 2015.

[18] A. Aijaz and A. H. Aghvami, "Cognitive machine-to-machine communications for Internet-of-Things: A protocol stack perspective," *IEEE Internet of Things Journal*, vol. 2, no. 2, pp. 103–112, 2015.

[19] Y. Liu, C. Yuen, X. Cao, N. U. Hassan, and J. Chen, "Design of a scalable hybrid MAC protocol for heterogeneous M2M networks," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 99–111, 2014.

[20] Y.-C. Pang, S.-L. Chao, G.-Y. Lin, and H.-Y. Wei, "Network access for M2M/H2H hybrid systems: A game theoretic approach," *IEEE Communications Letters*, vol. 18, no. 5, pp. 845–848, 2014.

[21] T.-M. Lin, C.-H. Lee, J.-P. Cheng, and W.-T. Chen, "PRADA: Prioritized random access with dynamic access barring for MTC in 3GPP LTE-A networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 5, pp. 2467–2472, 2014.

[22] A. Aijaz, S. Ping, M. R. Akhavan, and A.-H. Aghvami, "CRB-MAC: A receiver-based MAC protocol for cognitive radio equipped smart grid sensor networks," *IEEE Sensors Journal*, vol. 14, no. 12, pp. 4325–4333, 2014.

[23] S.-Y. Lien, T.-H. Liau, C.-Y. Kao, and K.-C. Chen, "Cooperative access class barring for machine-to-machine communications," *IEEE Transactions on Wireless Communications*, vol. 11, no. 1, pp. 27–32, 2012.

[24] M. Nakip, B. C. Gül, V. Rodoplu, and C. Güzeliş, "Comparative study of forecasting schemes for IoT device traffic in machine-to-machine communication," in *Proceedings of the 2019 4th International Conference on Cloud Computing and Internet of Things*, 2019, pp. 102–109.

[25] Y. Edalat, J.-S. Ahn, and K. Obraczka, "Smart experts for network state estimation," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 622–635, 2016.

[26] V. Petkov and K. Obraczka, "Collision-free medium access based on traffic forecasting," in *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2012, pp. 1–9.

[27] ——, "The case for using traffic forecasting in schedule-based channel access," in *2011 IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE, 2011, pp. 208–212.

[28] D. Raca, A. H. Zahran, C. J. Sreenan, R. K. Sinha, E. Halepovic, R. Jana, and V. Gopalakrishnan, "On leveraging machine and deep learning for throughput prediction in cellular networks: Design, performance, and challenges," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 11–17, 2020.

[29] L. Ruan, M. P. I. Dias, and E. Wong, "Machine learning-based bandwidth prediction for low-latency H2M applications," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3743–3752, 2019.

[30] H.-Y. Kim and J.-M. Kim, "A load balancing scheme based on deep-learning in IoT," *Cluster Computing*, vol. 20, no. 1, pp. 873–878, 2017.

[31] M. Nakip, V. Rodoplu, C. Güzeliş, and D. T. Eliiyi, "Joint forecasting-scheduling for the Internet of Things," in *2019 IEEE Global Conference on Internet of Things (GCIoT)*. IEEE, 2019, pp. 1–7.

[32] V. Rodoplu, M. Nakıp, D. T. Eliiyi, and C. Güzelis, "A multi-scale algorithm for joint forecasting-scheduling to solve the massive access problem of IoT," *IEEE Internet of Things Journal*, 2020.

[33] V. Rodoplu, M. Nakip, R. Qorbanian, and D. T. Eliiyi, "Multi-channel joint forecasting-scheduling for the Internet of Things," *IEEE Access*, vol. 8, pp. 217 324–217 354, 2020.

[34] E. L. Lawler and J. Labetoulle, "On preemptive scheduling of unrelated parallel processors by linear programming," *Journal of the ACM*, October.

[35] L. George, N. Rivierre, and M. Spuri, "Preemptive and non-preemptive real-time uniprocessor scheduling," 1996.