2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)

Supporting Delay-Sensitive IoT Applications: A Machine Learning Approach

Ali Alnoman
Faculty of Applied Science and Technology, Sheridan College, Canada

***Abstract*—In this paper, a supervised machine learning approach, namely, the decision tree is used to classify IoT applications based on their delay requirements. The decision-tree is trained and tested using simulated datasets to classify tasks into delay-sensitive and delay-insensitive based on the application features such as type and location. Delay-sensitive tasks are generally related to applications such as medical, manufacturing, and connected vehicles that require high service quality and short response time. Once delay-sensitive tasks are recognized, a prioritized scheduling mechanism is implemented to reduce the queueing delay at edge devices. Here, a two-class priority queueing system is used to model the scheduling mechanism at the edge device. Results show the effectiveness of machine learning in identifying delay-sensitive tasks that will experience shorter queueing delay at the edge device to enable high quality edge computing services.**

***Index Terms*—Machine learning, decision tree, edge computing, IoT, delay-sensitive.**

I. Introduction

The growing popularity of Internet of Things (IoT) applications imposed unprecedented amounts of connected devices on both communication and computing nodes in mobile networks. Therefore, it is necessary to devise efficient network management schemes to accommodate the diverse requirements of IoT devices. To this end, hierarchical network architectures such as heterogeneous cloud radio access networks (H-CRANs) and cooperative cloud-edge computing emerged to provide efficient network-wide management benefitting the advances in network function virtualization and software-defined networking technologies [1].

In terms of task processing, cloud computing helps to process tasks at the network core using the powerful cloud processors; nevertheless, with the ever-increasing computing demands, it is necessary to relieve cloud servers using efficient load management strategies such as task offloading. In this regard, edge computing emerged as a supporting architecture that assists the central cloud by processing tasks at the network edge near mobile users [2] [3]. As a result, several delay-sensitive applications especially in the areas of e-health, manufacturing, and connected vehicles will experience low latency and high service quality.

Therefore, in order to support delay-sensitive applications, it is essential to consider reducing delay in both computing and communication nodes since the overall delay experienced by users is a combination of both computing and communication. In this context, coordinating the performance of H-CRANs, cloud, and edge devices can bring the advantages of efficient network management, reduced latency, and context-awareness in mobile networks [4]. Moreover, incoming tasks can be classified into delay-sensitive such as biomedical sensors, and delay-insensitive (delay-tolerant) such as climate monitoring; accordingly, these tasks will be associated with either the near edge device or the distant cloud in order to maintain satisfactory service quality and load balancing between cloud and edge resources [5].

Minimizing delay in mobile computing networks has been considered in the literature using different approaches such as task offloading [6], content caching [7], and reducing the amounts of data exchange among IoT systems [8]. Moreover, minimizing the queueing delay in cloud servers to achieve low latency has been considered in several works such as [9]. Machine learning can efficiently improve the computing performance by adapting service provisioning according to real-world needs. For instance, content caching, resource allocation, task classification and aggregation can be more efficient with machine learning that helps to predict future behavior of users based on past events [10].

In this paper, a supervised machine learning technique, namely, the decision tree is proposed to classify tasks into delay-sensitive and delay-insensitive. Afterwards, a priority queueing system is introduced to represent the edge device in which delay-sensitive tasks are granted higher priority to avoid the long queueing delays. The decision tree model is trained and tested using simulated datasets taking into account the type, category, and delay-sensitivity nature of the proposed IoT applications. Furthermore, the accuracy of the proposed machine learning model is investigated and presented in the results section.

The rest of this paper is organized as follows. Section II introduces the system model. Section III explains the proposed machine learning and IoT task classification scheme. In Section IV, the priority queueing model is illustrated. Simulation setup and results are presented in Section V. Finally, Section VI provides some concluding remarks.

II. System Model

The mobile computing network under consideration consists of edge devices that are responsible for processing the computing tasks requested by users via the small base stations (SBSs). An edge device is modeled as an M/M/k queueing system where k denotes the number of servers or virtual machines (VMs) in the edge device. The existence of the macro base station (MBS) is aimed to assist in coverage and network

TABLE I: Simulating delay requirements of the proposed IoT applications

IoT Type \ Category	**Home**	**Enterprise**	**Manufacturing facility**	**Connected vehicles**
Lighting actuator	0	1	1	1
Temperature sensor	0	0	0	0
Motion sensor	0	0 or 1	0 or 1	1
Biomedical Sensors	1	1	1	1

control but is not considered in this work. Fig. 1 depicts the system layout.

III. IoT Task Classification

In this section, a machine learning model, namely, the decision tree is introduced to classify IoT tasks into delay-sensitive and delay-insensitive.

A. Decision Tree

A decision tree is a supervised machine learning technique that helps to make decisions based on historical data. The tree has its root at the top node and splits into one or more layers of subnodes. Each node tests a particular feature and the answer would be either 'True' or 'False'. The hierarchy of nodes (features) is arranged according to the best information gain obtained; that is, features that score the lowest uncertainty will make the best splits for the tree.

B. Model Training

The decision tree under consideration is trained to classify IoT tasks into delay-sensitive and delay-insensitive. Herein, a simulated dataset is simulated where each IoT task is assumed to have a particular type (function) and belongs to a particular category (place). For instance, a home temperature sensor has a type 'temperature' and belongs to the 'home' category. Four different IoT types are proposed, namely, lighting actuator, temperature sensor, motion sensor, and biomedical sensor. Each of the aforementioned IoT types is randomly assigned to one of four different categories, namely, home, enterprise, manufacturing facility, and vehicle. Table I illustrates the formation of the simulated training dataset. In Table I, delay-sensitive tasks are indicated by 1, whereas delay-insensitive tasks are indicated by 0. It is worth to mention that some cells in the table have 0 or 1 indicating that tasks can be either delay-sensitive or delay-insensitive.

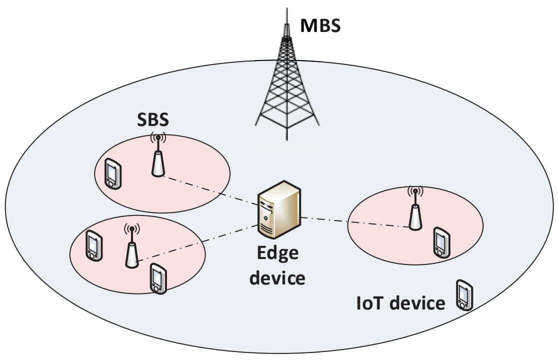

Fig. 1: General network and edge computing model.

C. Model Evaluation

In order to evaluate the proposed machine learning model, the classification accuracy is calculated along with other metrics such as the precision and recall based on the confusion matrix given below:

$$\begin{bmatrix} \text{True Positive (TP)} & \text{False Negative (FN)} \\ \text{False Positive (FP)} & \text{True Negative (TN)} \end{bmatrix}$$

where True Positive indicates the number of IoT tasks that are predicted as delay-sensitive and they are actually delay-sensitive in the dataset, whereas the False Positive is the number of IoT tasks that are predicted as delay-sensitive but they are actually delay-insensitive. Same meanings apply to the True Negative and False Negative scores. From the confusion matrix the accuracy, precision, and recall metrics can be calculated as follows: accuracy $= \frac{\text{TP+TN}}{\text{Total number of samples (population)}}$, precision $= \frac{\text{TP}}{\text{TP+FP}}$, recall $= \frac{\text{TP}}{\text{TP+FN}}$.

IV. Priority-based Edge Computing Model

The surge of IoT tasks arriving at edge devices can overload and degrade the performance of edge computing. Thus, to support delay-sensitive IoT applications, a priority-based edge computing model is introduced in this section as depicted in Fig. 2. In general, the utilization of an M/M/k queuing system is expressed as follows

$$\rho = \frac{\lambda}{k\mu} \tag{1}$$

where λ, k, and μ denote the arrival rate of tasks, number of servers, and the service rate of servers, respectively. It should be noted that ρ must be less than 1 in order to maintain system stability. Tasks arriving at the edge device will enter two separate queues (classes): delay-sensitive tasks (high priority class), and delay-insensitive tasks (low priority class). These two classes will be considered to calculate the mean queueing delay experienced by IoT devices [11] [12]. It is worth to mention that the delay experienced by users in a priority queueing system comes from three waiting periods, namely, the mean remaining service time of users being served, users already found in the queue with equal or higher priority, and users coming after and have higher priority.

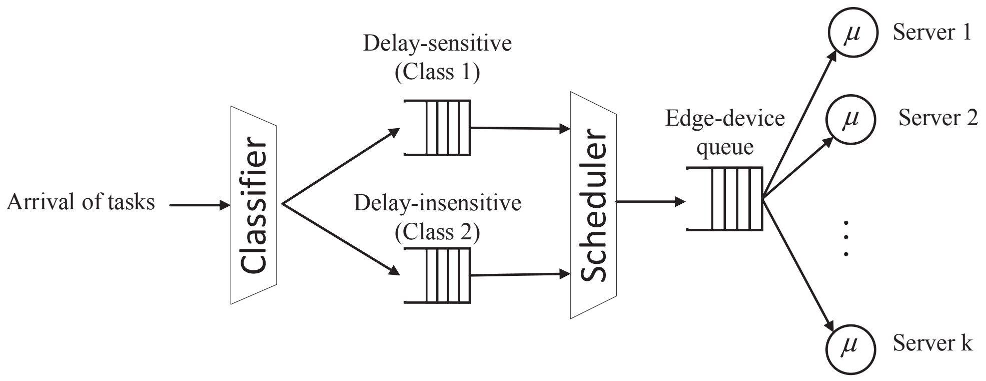

Fig. 2: Proposed priority queueing model.

V. Simulation Setup and Results

The proposed machine learning model is trained and tested using the machine learning libraries of the Python software whereas the dataset is built using the random functions of MATLAB. Portion of the dataset is presented in Table II. The dataset contains 1000 samples from which 700 samples are used for training the model and 300 for testing. The numbers from 1 to 4 in the 'IoT Type' column indicate lighting actuator, temperature sensor, motion sensor, and biomedical sensor, respectively. Similarly, the numbers from 1 to 4 in the 'Category' column indicate home, enterprise, manufacturing facility, and connected vehicles, respectively. As mentioned earlier, the task class is represented by either 1 or 0 indicating delay-sensitive or delay-insensitive, respectively. Moreover, we consider an edge device with 50 VMs ($k = 50$), an arrival rate λ in the range of 1-50 task/sec, a VM service rate $\mu = 1$ task/sec, and the permissible ratio of prioritized to all tasks in the edge device is 20%.

TABLE II: Portion of the simulated dataset

Sample no.	**IoT type**	**category**	**Class**
1	4	3	1
2	4	2	1
3	1	4	1
4	2	2	0
5	3	1	0

The machine learning model classifies IoT tasks as shown in Fig. 3 which shows the proposed decision tree with four levels. At each level, the tree splits into subnodes whereby IoT tasks are classified by testing a particular condition and returning a result of either true or false. In each node, the numbers of delay-insensitive and delay-sensitive tasks are respectively represented by a set of numbers between brackets. For instance, the condition 'IoT type <=3.5' in the first node checks if the IoT task belongs to any of the types 1-3 but not 4 which is 'biomedical sensor', and when the answer is 'False', the IoT task (biomedical sensor) is classified as delay-sensitive as seen in the dark blue node in which 159 samples are classified as delay-sensitive. The maximum number of levels can be chosen according to the desired accuracy; however, based on the accuracy shown in Table. III, a four-level decision tree was found to be sufficient to achieve an accuracy of 0.96, a precision of 0.93, and a recall of 1.

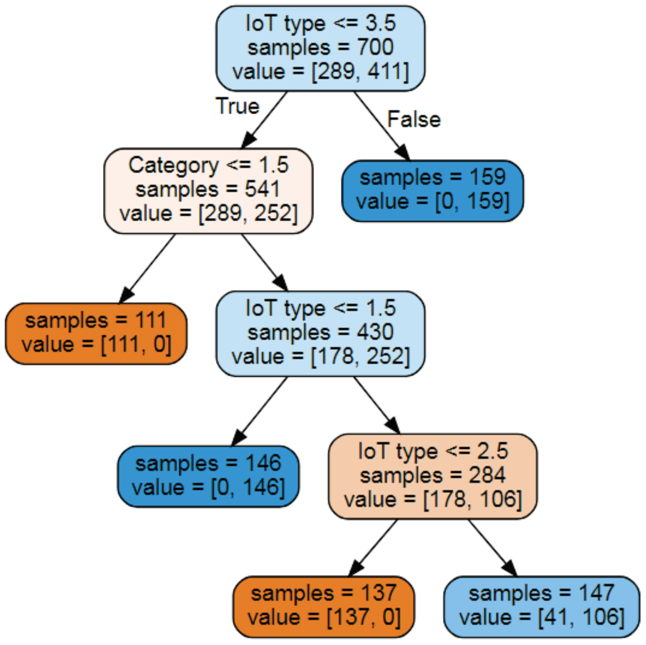

Fig. 3: Proposed four-level decision tree.

Prioritizing IoT tasks at the edge device can remarkably reduce the queueing delay as shown in Fig. 4. On the other hand, the delay-insensitive queue has experienced longer delay as a cost of the advantage granted to the delay-sensitive tasks. In the 'No priority' queue, all IoT tasks are processed equally regardless of the task class.

TABLE III: Accuracy of the decision tree with respect to the number of levels

Number of levels	Decision accuracy
1	0.68
2	0.77
3	0.89
4	0.96
5	0.96
6	0.96

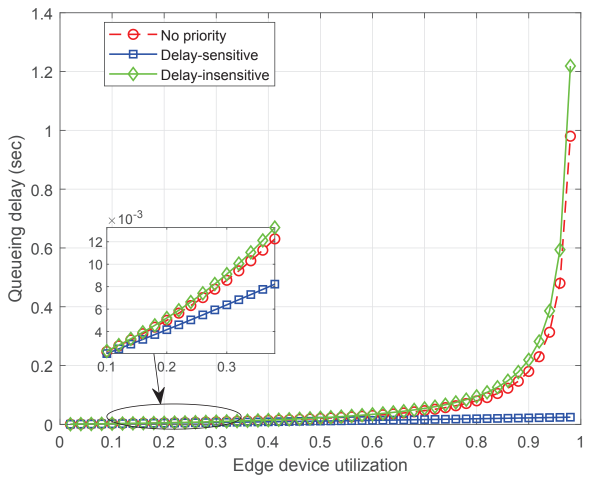

Fig. 4: Queueing delay experienced by IoT tasks.

VI. Conclusion

A supervised machine learning approach, namely, the decision tree was proposed in this paper to classify IoT tasks into delay-sensitive and delay-insensitive. Moreover, a two-class prioritized computing model was implemented to reduce the undesired queueing delays experienced by delay-sensitive applications. Results showed the success of the proposed model in classifying IoT tasks, and the effectiveness of the prioritized system to reduce the delay experienced by delay-sensitive IoT applications.

References

[1] A. Alnoman, G. H. Carvalho, A. Anpalagan, and I. Woungang, "Energy Efficiency on Fully Cloudified Mobile Networks: Survey, Challenges, and Open Issues," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1271 – 1291, 2nd Quart. 2018.

[2] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the Edge: A Scalable IoT Architecture Based on Transparent Computing," *IEEE Network*, vol. 31, no. 5, pp. 96 – 105, Oct. 2017.

[3] E. Ahmed *et al.*, "Bringing Computation Closer toward the User Network: Is Edge Computing the Solution?" *IEEE Communications Magazine*, vol. 55, no. 11, pp. 138 – 144, Nov. 2017.

[4] S. L. S. Hung, H. Hsu and K. Chen, "Architecture Harmonization Between Cloud Radio Access Networks and Fog Networks," *IEEE Access*, vol. 3.

[5] L. Zhang, K. Wang, D. Xuan, and K. Yang, "Optimal Task Allocation in Near-Far Computing Enhanced C-RAN for Wireless Big Data Processing," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 50 – 55, Feb. 2018.

[6] C. Liu, M. Bennis, and H. Poor, "Latency and Reliability-Aware Task Offloading and Resource Allocation for Mobile Edge Computing," in *IEEE Globecom Workshops*, Dec. 2017, pp. 1 – 7.

[7] M. S. Elbamby, M. Bennis, and W. Saad, "Proactive Edge Computing in Latency-Constrained Fog Networks," in *European Conference on Networks and Communications (EuCNC)*, June 2017, pp. 1–6.

[8] T. Yu, X. Wang, and A. Shami, "A Novel Fog Computing Enabled Temporal Data Reduction Scheme in IoT Systems," in *IEEE Global Communications Conference*, Dec. 2017, pp. 1–5.

[9] T. H. Szymanski, "An Ultra-Low-Latency Guaranteed-Rate Internet for Cloud Services," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 123 – 136, Feb. 2016.

[10] A. Alnoman, S. Sharma, W. Ejaz, and A. Anpalagan, "Emerging Edge Computing Technologies for Distributed IoT Systems," *IEEE Network*, vol. 33, no. 6, pp. 140 – 147, Dec. 2019.

[11] A. Alnoman and A. Anpalagan, "A Dynamic Priority Service Provision Scheme for Delay-Sensitive Applications in Fog Computing," in *29th Biennial Symposium on Communications (BSC)*, June 2018.

[12] G. Bolch, S. Greiner, H. de Meer, and K. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley, 2006.