ICAQ: Adaptive QoS System for 5G and Beyond Applications

Liang Zhang*, Guoqing Ma*, Amer Al-Ghadhban†, Shuping Dang*, and Basem Shihada*
*Computer, Electrical and Mathematical Science and Engineering Division,
King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Saudi Arabia
†Electrical Engineering Department in College of Engineering at University of Hail, Hail 55476, Saudi Arabia
{liang.zhang, guoqing.ma, shuping.dang, basem.shihada}@kaust.edu.sa, a.alghadhban@uoh.edu.sa

***Abstract*—Management of network becomes increasingly difficult due to the rapid proliferation of the Internet of Things (IoT) and heterogeneous demands of applications. To mitigate congestion and maintain a high quality of service (QoS) for application users, we propose an importance-oriented clustering-based QoS system named ICAQ. The system uses unsupervised machine learning to determine and differentiate different application flow priority. The system is proposed to be a generic to both network core features and the application features. An environment monitoring IoT application is implemented as an example to demonstrate the expressive power of ICAQ. Experimental results have verified the effectiveness and efficiency of ICAQ and show a promising potential for using ICAQ for the QoS management of IoT applications in 5G and beyond networks.**

***Index Terms*—Feature clustering, software-defined network (SDN), quality of service (QoS) management, Internet of Things (IoT).**

I. Introduction

The concepts of smart city, automatic vehicle and beyond fifth generation (5G) edge computing have gained popularity over the last decade, which pours myriads of data into network [1]. It has been found difficult to guarantee the QoS of Internet of Things (IoT) applications by the uniform traffic transferring strategy, since the spatial distribution of IoT sensors is not uniform and the numbers of the packets generated by IoT sensors at different time are uneven [2]. Traffic engineering makes it feasible to enhance QoS by detecting and controlling traffics [3]. Analogously, research effort in recent years has focused on transferring the methods in traffic engineering to manage data traffic through networks. This can be achieved by extracting and recognizing the features of the network core, e.g., the size and the typology of traffic [4].

Such network core features cannot fully exhibit the importance of IoT packets. The IoT sensors have the tendency to be coupled together as clusters because of the similar IoT sensor features [5]. In general, two kinds of flows appear in communication network, elephant and mice. In this paper, we enhance the definition of elephant and mice flows with additional context corresponding to the IoT application layer. Specifically, the flows extracted from high-density clusters sharing similar characters are defined as elephant flows, while the mice flows are generated by the sensors in low-density clusters. The so-called elephant flows have huge traffic volumes and thereby are always the determinants of network congestion and resource waste [6]. Obviously, most of the samples with similar features provide similar information for users. As a direct consequence of this similarity, increasing transmission rate of elephant flows with similar features cannot significantly increase the QoS of IoT applications [7]. Moreover, analyzing and transmitting these elephant flows consume both the computing resource and energy. On the other hand, the flows with distinguish features normally deserve relatively higher importance in this regards. In an extreme case, if the mice flow is the single traffic that contains the information of a certain cluster, the importance of this mice flow can be infinitely high by definition. Therefore, once the mice flow is dropped, the information of the certain cluster is totally lost. For this reason, the mice flows are generally more important than elephant flows. Thus, an explicit classification process for IoT flows and a QoS guarantee mechanism based on the flow importance are imperative.

In practice, however, providing such a QoS guarantee mechanism is a complex process. There exist enormous obstacles in traditional networks hindering efficient QoS guarantees. First of all, traffic shaping, bandwidth guarantee and priority protocol regulations need to be manually configured for thousands of distributed switches [8]. Furthermore, the dynamic management of huge-volume IoT traffic in real time becomes a necessity to meet the stringent communication QoS [9]. Both requirements are beyond the capability of traditional data networks. By adopting the principle of decoupling of the control and data planes, the software-defined network (SDN) a promising solution to tackle the aforementioned difficulties [10]. SDNs can dynamically manage the network behaviors via the open south interfaces, e.g., OpenFlow protocol [11]. In addition, the flexible and scalable SDN technique can be applied to implement OpenFlow rules assigned by various machine learning algorithms [12]. Primary studies have shown that a well-designed SDN can monitor and guarantee the QoS in core networks by the extracted flow features [13].

Meanwhile, plenty of research activities in the realm of data science and network science have focused on the intelligent management for the network traffic so as to mitigate the network congestion [14], [15]. Wang et. al. in [16] used semi-supervised machine learning to classify the traffic according to different QoS requirements. They monitored the network traffic depending on the statistical features extracted from

network flows. However, the per-flow statistics cannot be well processed in practice because of limited storage space and computational complexity. Also, the features of edge IoT sensors are not concerned. Cohen and Moroshko provided a demand based sampling method by applying an efficient utility function [17]. The method is difficult to extend to a general application scenario, albeit with a high efficiency. Ahmad et. al. proposed an SDN-based system, called SADIQ, which implements the QoS policies in a high-level and SQL-like programming language [18]. Prioritization of IoT flows was expressed with a list-based policy that contains an ordered list of mice and elephant regions. On the other hand, the flow classification and prioritization have not been well designed yet. Alaslani et. al. proposed an intelligent edge for detecting traffic load and perform proper rate limiting[19]. Despite the above milestones in flow control and QoS management, the management of the QoS of IoT applications is still an open issue and even a challenge due to heterogeneous service requirements and the diversity of IoT flows.

To realize efficient QoS management for IoT applications, we propose an importance-oriented clustering-based QoS system named ICAQ. Our proposed system aims at classifying IoT packets into clusters with different priority levels, on which the traffic management in SDNs depends. To construct ICAQ, a clustering algorithm is given and embedded to determine QoS optimization priority. The clustering process considers both traditional traffic features extracted from the core network and novel features extracted from IoT sensors as well as the traffic contexts. This adaptive system is capable of dynamically updating priority rules at switches via an SDN controller.

The rest of this paper is organized as follows. In Section II, we formulate the problem of QoS management for IoT applications. Then, the implementation details of ICAQ is given in Section III. We carry out performance evaluation and discussion based on experimental results in Section IV. Finally, we conclude the paper in Section V.

II. Problem Formulation

The IoT packets generated by IoT sensors are sent to destination servers via switches. The SDN controller enables switches to duplicate traffic flow to the controller for analysis. The analytical process is executed on analytical servers in an offline manner and thus does not cause additional network jitters and delay. The k-means algorithm is adopted for the module training due to its advantage of low complexity and high compatibility of various data distributions. The hyper-parameter k is determined based on the QoS levels and known *a priori*. Samples related to the observations extracted from IoT sensors are classified into differentiated clusters. IoT flows assigned to the same cluster share similar features and present a similar context enclosed in the packets. Each cluster is lined to a set of potential labels, which are unknown to the clustering process in advance. The label in our module is defined as a function of importance. The importance function is grasped by the user of various applications. The central controller makes the forwarding decision and implements the traffic management rules against traffic congestion. The IoT flows with certain features matching the flow entries are assigned to the corresponding QoS queues. Fig. 1 exhibits the overall SDN architecture considered in this paper.

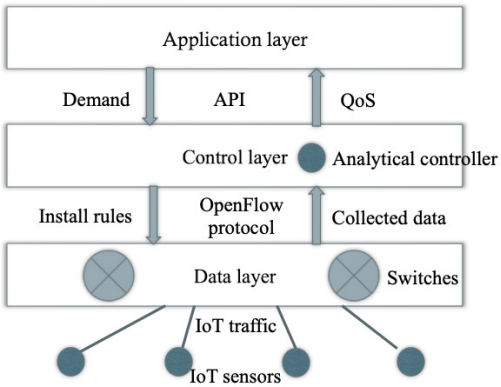

Fig. 1. SDN architecture with key components and layers.

A. ICAQ Parameters

Most of the flows contain multiple features, which can be organized as a vector. Accordingly, an n-tuple vector $\mathbf{x}$ is presented as $\mathbf{x}(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$, where $\mathbf{x}_i$ is derived to present the features of traffic $\forall\ i = 1, 2, \ldots, n$. There are three ICAQ parameters of paramount importance required to be explained, i.e., the QoS level, the dimension of IoT features, and the number of samples. We expatiate on each of them in the following paragraphs.

1) QoS level: The number of the clusters k is the assessment of the resolution of the QoS level, which needs to be decided in advance. To achieve fine-grained classification, k is supposed to be large. On the contrary, a small k corresponds to coarse-grain classification. The value of k yields impacts on the algorithmic complexity. It is thereby essential to maintain a proper trade-off between the complexity of the algorithm and the resolution of the QoS level, since the computational capability of the controller and the updating time interval impose constraints on the dynamic management.

2) Dimensions of IoT features: As the input of the clustering strategy, $\mathbf{x}_i$ is a d-dimension vector $\mathbf{x}_i(x_{i,1}, x_{i,2}, ...x_{i,u}, ..., x_{i,d})$, where $x_{i,u}$ presents the value of the uth feature for the ith packet; the parameter d is the number of the features that contribute to the performance of the IoT application. The features extracted from the packets can be categorized into two classes:

- Features identified from the header of the flows by using a monitor probe, e.g., the IP address and the protocol type.
- Additional features extracted from the IoT sensors, e.g., the sensor location, monitoring values, and the sensor type.

In addition, some statistical features in a certain time interval, including the packet number and the packet size, are treated

TABLE I
CATEGORIES OF FEATURES EXTRACTED FROM TRAFFIC

Network flow features	IoT sensor features
addr: src_ip, dst_ip	sen_inf: longitude (x)
src_mac, dst_mac	atitude (y)
src_port, dst_port	monitoring sensor value
protocol_type:	sensor_type:
RTSP, RTP, RTCP	CCD camera
FTP, HTTP	air temperature sensor (ATS)
TCP, UDP	wearable devices
packets_num, packets_size	

as conjunct features for network flows and IoT sensors. The categories of the features are summarized and shown in Table I. The features summarized in Table I are some examples of each category. The feature selection is all up to the demands of applications. Investigation of feature extracting has been done by using machine learning, such as decision tree and neural network [20], [21]. The chosen features jointly contribute to the importance estimation. After clustering, IoT sensor features can be mapped to network flow features. This mapping process can reduce complexity of matching the flow entries and save the storage space at switches.

3) Number of samples: The huge number of packets is a challenge for sampling for practical IoT networks. Fortunately, two of the main advantages of this k-means approach are the fast convergence and the easy-to-implement nature in large-scale networks. Some preliminaries and data cleaning methods, such as data normalization, can be used by imposing scikit-learn [22]. To maintain an adequate equilibrium between computational burden and accuracy, the number of samples needs to be carefully designed. A large number of samples can inevitably result in a high accuracy when extracting features from samples, albeit with a loss of computational efficiency and long processing time.

B. Coupling Relation among IoT Flows

The IoT flows with similar features are coupled, and the flows corresponding to the same cluster tend to have the same importance status for QoS management. A metric function is required to describe this coupling relation among the IoT flows. Regarding the IoT flows $\mathbf{x}_i(x_{i,1}, x_{i,2}, ..., x_{i,d})$ and $\mathbf{x}_j(x_{j,1}, x_{j,2}, ..., x_{j,d})$, the coupling relation can be characterized by as the distance measure $\mathsf{dist}(\mathbf{x}_i, \mathbf{x}_j)$. The smaller the distance measure is, the tighter the coupling relation will be. For good practicability, this distance measure must satisfy some conditions. First, it should have no preference on the starting point of measurement. Second, the distance measure equals zero only when two IoT flows are identical. Mathematically, we can express these two conditions as

$$\mathsf{dist}(\mathbf{x}_i, \mathbf{x}_j) = \mathsf{dist}(\mathbf{x}_j, \mathbf{x}_i) \tag{1}$$

and

$$\mathsf{dist}(\mathbf{x}_i, \mathbf{x}_j) = 0 \text{ iff } \mathbf{x}_i = \mathbf{x}_j. \tag{2}$$

The Minkowski distance, as the most commonly used distance measure, satisfies both of the conditions specified above, which is given by [23]

$$\mathsf{dist}_{\mathrm{MK}}(\mathbf{x}_i,\ \mathbf{x}_j) = \left(\sum_{u=1}^{d} |x_{i,u} - x_{j,u}|^P\right)^{1/P}, \tag{3}$$

where P is an integer constant parameter. When $P = 2$, the distance function reduces to be the Euclidean distance

$$\mathsf{dist}_{\mathrm{ED}}(\mathbf{x}_i,\ \mathbf{x}_j) = \left(\sum_{u=1}^{d} |x_{i,u} - x_{j,u}|^P\right)^{1/2}. \tag{4}$$

Note that the features of IoT samples are classified into the ordinal feature and the non-ordinal feature. The location of the IoT sensor and the IP address, for instance, are ordinal features, of which the distance can be easily calculated. The distance between (25N, 121E) and (25N, 151E) is obviously larger that between (25N, 121E) and (25N, 131E). As typically a non-ordinal attribute, the protocol value (VoIP or FTP), however, is not advisable to be assessed by directly applying the Minkowski distance. In this case, the value difference metric (VDM) is used to describe the coupling relation between IoT flows [24]:

$$\mathsf{VDM}_{\mathrm{p}}(\mathrm{VoIP}, \mathrm{FTP}) = \sum_{i=1}^{k} \left| \frac{m_{p,\mathrm{VoIP},i}}{m_{p,\mathrm{VoIP}}} - \frac{m_{p,\mathrm{FTP},i}}{m_{p,\mathrm{FTP}}} \right|^P, \tag{5}$$

where $m_{p,\mathrm{VoIP}}$ denotes the number of IoT flows adopting the VoIP protocol; $m_{p,\mathrm{FTP}}$ denotes the number of flows adopting the FTP protocol; $m_{p,\mathrm{VoIP},k}$ and $m_{p,\mathrm{FTP},k}$ refer to the numbers of VoIP and FTP flows in the kth cluster, respectively.

By (3) and (5), we can associate the Minkowski distance with the VDM method to process both ordinal features and non-ordinal features for the estimation of the importance. Assume that the number of ordinal features is d_c. Then, the number of the non-ordinary features is $d - d_c$. The distance measure can be expressed as

$$\begin{aligned}&\mathsf{MinkovDM}(\mathbf{x}_i,\ \mathbf{x}_j)\\ &= \left(\sum_{u=1}^{d_c} |x_{i,u} - x_{j,u}|^P + \sum_{u=d_c+1}^{d} \mathsf{VDM}_u(x_{i,u}, x_{j,u})\right)^{1/P}.\end{aligned} \tag{6}$$

Though this association is useful, as a direct negative consequence, the comparability principle among multiple features is violated. This is because the features of IoT flows are with different physical significance and are distinctive in unit. Moreover, the features are not equally sensitive to the estimation of importance status. One possible way to solve the dilemma is to set a weighting factor w_u, $\forall\ u = 1,\ 2,\ ...,\ d$, corresponding to all features with divergent importance levels. In this way, we can write the weighted distance as

$$\mathsf{dist}_{\mathrm{wmk}}(\mathbf{x}_i,\ \mathbf{x}_j) = \left(\sum_{u=1}^{n} w_u |x_{i,\ u} - x_{j,\ u}|^P\right)^{1/P}. \tag{7}$$

If some of the features have less contribution to the QoS of IoT applications, the weights of the features are set to be lower. In this regard, the system is a general strategy for both network core and network edge QoS management. When the importance estimation is determined by the features only related to edge IoT sensors, the weight parameters of attribute associated with network core can be even set to zero.

C. Clustering Process of IoT Flows

Among the traffic collected from IoT sensors, an initial set of k IoT flows $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k$ are selected randomly or depending on the demands from a certain IoT application. The selected flows are treated as the initial central flows denoted by $\mu(\mu_1, \mu_2, ..., \mu_k)$. Taking the smart city environment monitoring as an example. The initial central flow can be extracted from the IoT sensors located in metropolises. The clustering algorithm proceeds by two simple steps [25]. First, for each IoT flow, find the most tightly coupled central flow μ_i. The flow is apportioned to the same cluster with μ_i. To achieve this, we sum up the distance metrics between IoT flow and each central flows. For an n-tuple vector $\mathbf{x}$, we donate the minimum distance partition as $\mathbf{S}(\mathbf{x}) = (\mathbf{S}_1(\mathbf{x}), \mathbf{S}_2(\mathbf{x}), ..., \mathbf{S}_k(\mathbf{x}))$, where $\mathbf{S}_i(\mathbf{x})$ is a set containing the nearest samples to $\mathbf{x}_i$ [26]. Now, the objective of the clustering process is to find a proper set $\mathbf{S}(\mathbf{x})$ so as to minimize the sum distance of IoT flows:

$$\mathbf{S}^*(\mathbf{x}) = \underset{\mathbf{S}(\mathbf{x})}{\arg\min} \left\{ \sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathbf{S}_i(\mathbf{x})} |\mathbf{x} - \mu_i|^2 \right\}. \tag{8}$$

However, finding the solution to (8) is demanding, since it is an NP-hard problem [27]. The greedy algorithm is a regular method to solve the problem [28]. Secondly, the optimized set $\mathbf{S}^*(\mathbf{x})$, μ_i obtained by (8) is updated according to newly coming central IoT flows μ'_i by

$$\mu'_i = \frac{1}{|\mathbf{S}_i|} \sum_{\mathbf{x} \in \mathbf{S}_i} \mathbf{x}. \tag{9}$$

The iteration ends up with the indication that the discrepancy between the current central flow and the updated central flow becomes insignificant.

The flows classified in the same cluster with a similar importance status are marked by the same clustering label λ_j. Thus, the QoS guarantee of these samples should also be set to the same level. According to the assumptions that elephant traffic dominates the network congestion and samples with similar features enclose similar information, setting a low QoS priority for elephant flows can reduce the network resource consumption and improve the operating efficiency. The numbers of IoT flows in each cluster ρ_i demarcate by the elephant IoT flows and mice IoT flows. The importance function is designed to positively correlate with $1/\rho_i$. The number of clusters ρ_i, the cluster label of each IoT flow λ_j, and the importance function P_i are set as the outputs of the clustering process.

Once the importance function is attained, OpenFlow entries on queue allocation for IoT traffic are installed at switches by the controller. All traffic flows with features matching the flow entries are partitioned into relevant queues with different QoS guarantee levels. Data collection is accomplished by the controller or server hosts in the same manner. For the missing information due to the drop of packets, the interpolation method is employed to estimate the information and restore the lost information offline [29]. Simultaneously, the IoT traffic is also collected by the controller for analytical purposes. The number of samples in the cluster with higher priority grows faster, since the possibility of transmitting these samples is higher. In every time slot, the QoS priority rules need to be updated to avoid the over-guarantee of mice flows. By adopting this adaptive strategy, the priority level decreases along with an increasing number of the flows in the relevant cluster. This adaptive system can avoid an undesirable situation, in which certain IoT flows belonging to the low-priority clusters are always dropped.

III. System Implementation

In this section, we evaluate effectiveness and efficiency of ICAQ using a typical 5G and beyond smart city environment monitoring IoT application. We consider an open sensor dataset that contains the measured data of temperatures and locations, including latitude and longitude, sensor IDs, and measuring time [30]. Without loss of generality, latitude, longitude, and temperature are bundled as a three-tuple vector input in our example. The data set with UDP packets was collected from 256 distributed IoT sensors. We utilized traffic flows sampled in an one-hour time frame as the training data. The clustering algorithm was trained offline in the controller. Once the training process is completed, the forwarding rules are implemented at switches by the controller. In this clustering algorithm, data normalization is necessary since the units of temperature and location features are different. The hyper-parameter of the k-means clustering algorithm, i.e., k, is set to be 5 corresponding to five QoS guarantee policies: the default forwarding (DF) policy, three expedited-forwarding (EF1, EF2, EF3) policies, and the assured-forwarding (AF) policy. We present the pseudocode of the k-means clustering algorithm in Algorithm 1.

For simplicity, the values of importance functions for different clusters are given by $P_1 = 1,\ P_2 = 0,\ P_3 = 2,\ P_4 = 3,$ and $P_5 = 4$. The higher the value of the importance function is, the better the QoS guarantee should be assigned to the queue corresponding to the cluster. The IoT flows marked as green hexagrams constitute Cluster 5 with the highest priority, and thus are assigned to the queue guaranteed with the AF policy. On the other hand, IoT flows pertaining to Cluster 2 are discerned as elephant flows that are allocated to the queue with the QoS policy of best effort forwarding, i.e., the DF policy. The other three clusters are allocated to the queues with the EF1, EF2, and EF3 policies, respectively.

The testbed used in this paper consists of a Mininet simulator, a Ryu controller, the OpenFlow south protocol, and the Open vSwitch (OVS). Scapy and Wireshark are also used for message packing and traffic capture. We first construct

a typical linear structure using the Mininet simulator. 256 distributed IoT sensors send data to the server via OVSs. As an interaction-friendly SDN controller, Ryu provides plenty of application programming interfaces (APIs) for customer programming. We modify the `simple_switch` API to activate the QoS ensuring functionality. Then, we execute `rest_qos`, `qos_simple_switch_13`, and `rest_conf_switch` APIs by the Ryu controller. The QoS settings are configured by installing the flow entries into OVSs. The traffic apportion is realized by keywords matching, and the traffic shaping is performed by limiting the transmission rate. We map the IoT sensor features, i.e., locations of the sensors, to the network flow features, i.e., IP addresses of the hosts, for simplicity. In this way, a small set of features are used for flow entries matching, which reduces the computational complexity and save the storage space. The hierarchical token bucket (HTB) protocol contributes to the utilization of bandwidth, since the IoT flows with lower priority utilize the resource only if the bandwidth is not occupied by transmission tasks of high-priority IoT flows.

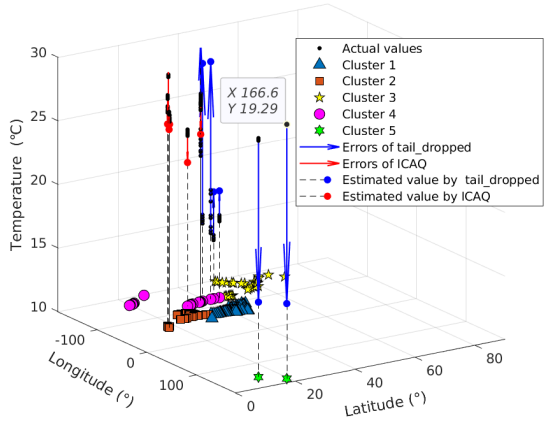

Fig. 2. Clustering results for the environment monitoring IoT application.

Algorithm 1 k-means clustering algorithm for IoT flows.

BEGIN
Load $\mathbf{D} = (\mathbf{x}_1,\ \mathbf{x}_2,\ ...,\ \mathbf{x}_n)$ for three dimensions: altitude, longitude, and temperature;
Set $k = 5$, corresponding to DF, EF1, EF2, EF3, and AF;
Normalize equal weights of location attribute and monitoring context attribute, excluding the impact caused by the disparateness of feature unit;
Select five IoT flows randomly to form the initial cluster center $\mu_1,\ \mu_2,\ \mu_3,\ \mu_4,\ \mu_5$;
while the center IoT flow is varying **do**
  for $i = 1 : 5$ **do**
    $\mathbf{C}_i = \varnothing$;
  end for
  for $j = 1 : n$ **do**
    for $i = 1 : 5$ **do**
      Calculate the Minkowski distance between $\mathbf{x}_j$ and μ_i by $d_{j,\ i} = |\mathbf{x}_j - \mu_i|$;
    end for
    Derive the cluster label of $\mathbf{x}_j$ by $\lambda_j = \arg\min_i\{d_{j,\ i}\}$;
    Place $\mathbf{x}_j$ into the cluster by $\mathbf{C}_{\lambda_j} \bigcup \mathbf{x}_j \mapsto \mathbf{C}_{\lambda_j}$;
  end for
  for $i = 1 : 5$ **do**
    Calculate the new center IoT flow by $\mu'_i = \frac{1}{|\mathbf{C}_i|}\sum_{\mathbf{x}\in\mathbf{C}_i}\mathbf{x}$;
    if $\mu'_i \neq \mu_i$ **then**
      Update μ_i by μ'_i;
    else
      Keep the center IoT flow μ_i unchanged;
    end if
  end for
end while
Count the number of the samples for each cluster ρ_i;
Obtain the importance function P_i by mapping $1/\rho_i$ to interval (0, 1, 2, 3, 4);
return the number of each cluster $\{\rho_i\}$, the cluster label of each IoT flow $\{\lambda_j\}$, and the importance function $\{P_i\}$;
END

IV. Performance Evaluation

To evaluate the effectiveness of the proposed system, we carry out the experiments and compare the performance of ICAQ with classic tail dropping strategy. In the experiments, open datasets generated in the time frames from 1:00 am, Sept. 7th, 2019 until 2:00 am, Sept. 7th, 2019 are selected as test data. 2825 packets are poured into the network topology simulated by the Mininet simulator. We impose restrictions on the bandwidth with 200 kbit/s, 190 kbit/s, 180 kbit/s, 170 kbit/s, and 149 kbit/s, resulting in the traffic loss of 0.57%, 3.76%, 10.23%, 13.77% and 25.63%, respectively. The Python module Scapy is used to sniff the traffic at the server host. The priority rules are regularly updated by the controller. The updating time interval is flexible and depends on the end-user requirements of the IoT application.

With the aforementioned settings, we first present the clustering results for the environment monitoring IoT application in Fig. 2. The dropped traffic under the network condition with 3.76% loss, corresponding to the bandwidth with 190 kbit/s, is depicted in Fig. 2 by small black points. To assess the performance, we restore the data by interpolation fitting. The results regarding the traditional tail dropped method shows that the IoT flows with higher priority are totally lost. Also, the restored data by the nearest interpolation fitting method have a large number of errors. Taking the sample located at (166.6N, 19.3E) as an example, the estimated value is far from the actual value. When implementing ICAQ, interpolated data concentrates upon elephant flows, by which information can be extracted from the nearest IoT flows.

Based on the clustering results, we are able to compare the root mean square error (RMSE) of the data obtained by ICAQ

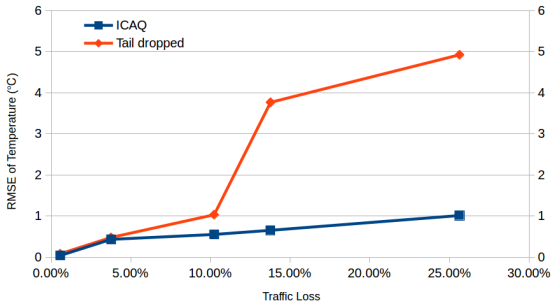

Fig. 3. RMSE comparison between ICAQ and the classic tail dropping strategy.

and the classic tail dropping strategy under the same network configurations. RMSE represents the standard deviation of the residuals (prediction errors). It is an instinctive measure of how far the estimated data are from the actual data. In the contexts of clustering and QoS management, the RMSE value is negatively related to the precision of the information received. In this regard, RMSE for our experiments can be defined as

$$\mathsf{RMSE} = \left(\frac{1}{n} \sum_{i} |T(\mathbf{x}_i) - T'(\mathbf{x}_i)|^2 \right)^{1/2}, \qquad (10)$$

where $T(\mathbf{x}_i)$ and $T'(\mathbf{x}_i)$ represent the measured value and the estimated value of IoT flow $\mathbf{x}_i$. We present the RMSE comparison between ICAQ and the classic tail dropping strategy in Fig. 3. The experimental results in Fig. 3 verify the performance superiority of ICAQ over the benchmark method under various traffic-loss settings. In the worst case of 25.63% traffic loss, ICAQ is still able to reduce RMSE by 80%.

V. Conclusion

In this paper, we proposed a novel QoS management system named ICAQ. This system employs a k-means clustering algorithm to classify IoT flows and identify both elephant and mice flows. By the clustering process, the importance function can be determined for each cluster, and the QoS guarantee is achieved by feature matching and traffic shaping with the support of flow entries. We also verified the effectiveness and efficiency of ICAQ by an IoT application for smart city environment monitoring. Experimental results have shown a significant performance improvement brought by ICAQ is attained, which leads to a reduction of RMSE by 80% compared to the classic tail dropping method.

References

[1] S. Dang, O. Amin, B. Shihada, and M.-S. Alouini, "What should 6G be?" *Nature Electronics*, vol. 3, no. 1, pp. 20–29, 2020.

[2] C. Ren, X. Lyu, W. Ni, H. Tian, and R. P. Liu, "Distributed online learning of fog computing under nonuniform device cardinality," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1147–1159, 2018.

[3] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Computer Networks*, vol. 71, pp. 1–30, 2014.

[4] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[5] R. Morabito, I. Farris, A. Iera, and T. Taleb, "Evaluating performance of containerized IoT services for clustered devices at the network edge," *IEEE Internet of Things Journal*, vol. 4, no. 4, pp. 1019–1030, 2017.

[6] J. Liu, J. Li, G. Shou, Y. Hu, Z. Guo, and W. Dai, "SDN based load balancing mechanism for elephant flow in data center networks," in *Proc. IEEE WPMC*, Sydney, Australia, 2014, pp. 486–490.

[7] R. B. Basat, G. Einziger, R. Friedman, and Y. Kassner, "Optimal elephant flow detection," in *IEEE INFOCOM*, Atlanta, GA, USA, 2017.

[8] L. Chen, M. Qiu, and J. Xiong, "An SDN-based fabric for flexible data-center networks," in *Proc. IEEE CSCloud*, New York, NY, US, 2015.

[9] F. Bannour, S. Souihi, and A. Mellouk, "Distributed sdn control: Survey, taxonomy, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 333–354, 2018.

[10] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," in *Proc. ACM SIGCOMM*, Kyoto, Japan, Aug. 2007, pp. 1–12.

[11] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-defined networking (SDN): Layers and architecture terminology," Tech. Rep., 2015.

[12] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir, "Application-awareness in SDN," in *Proc. ACM SIGCOMM*, New York, NY, US, 2013, pp. 487–488.

[13] S. V. Morzhov and M. A. Nikitinskiy, "Development and research of the PreFirewall network application for floodlight SDN controller," in *Proc. IEEE MWENT*, Moscow, Russia, Mar. 2018, pp. 1–4.

[14] S. Yoon, T. Ha, S. Kim, and H. Lim, "Scalable traffic sampling using centrality measure on software-defined networks," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 43–49, Jul. 2017.

[15] Z. Su, T. Wang, Y. Xia, and M. Hamdi, "Flowcover: Low-cost flow monitoring scheme in software defined networks," in *Proc. IEEE GLOBECOM*, Austin, TX, USA, Dec. 2014, pp. 1956–1961.

[16] P. Wang, S. Lin, and M. Luo, "A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs," in *Proc. IEEE SCC*, Jun. 2016, pp. 760–765.

[17] R. Cohen and E. Moroshko, "Sampling-on-demand in SDN," *IEEE/ACM Trans. Networking*, vol. 26, no. 6, pp. 2612–2622, Dec. 2018.

[18] E. Ahmad, M. Alaslani, F. R. Dogar, and B. Shihada, "Location-aware, context-driven QoS for IoT applications," *IEEE Syst. J.*, 2019.

[19] M. Alaslani and B. Shihada, "Intelligent edge: An instantaneous detection of iot traffic load," in *Proc. IEEE ICC*, 2018, pp. 1–6.

[20] Y.-C. Wu, H.-R. Tseng, W. Yang, and R.-H. Jan, "DDoS detection and traceback with decision tree and grey relational analysis," in *Proc. IEEE MUE*, Qingdao, China, Jun. 2009, pp. 306–314.

[21] S. R. Mounce, A. J. Day, A. S. Wood, A. Khan, P. D. Widdop, and J. Machell, "A neural network approach to burst detection," *IWA Water Sci. Technol.*, vol. 45, no. 4-5, pp. 237–246, Feb. 2002.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *MIT Press J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[23] R. C. De Amorim and B. Mirkin, "Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering," *Pattern Recognition*, vol. 45, no. 3, pp. 1061–1075, 2012.

[24] C. Stanfill and D. Waltz, "Toward memory-based reasoning," *ACM Commun.*, vol. 29, no. 12, pp. 1213–1228, Dec. 1986.

[25] Y. Yang, "Information theory, inference, and learning algorithms," *J Am. Stat. Assoc.*, vol. 100, no. 472, pp. 1461–1462, 2005.

[26] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Berkeley Symposium on Mathematical Statistics and Probability*, Jun. 1967, pp. 281–297.

[27] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "NP-hardness of Euclidean sum-of-squares clustering," *Mach. Learn.*, vol. 75, no. 2, pp. 245–248, May 2009.

[28] C. E. Leiserson, R. L. Rivest, T. H. Cormen, and C. Stein, *Introduction to algorithms*. Cambridge, MA, USA: MIT press, 2001.

[29] H. Akima, "A new method of interpolation and smooth curve fitting based on local procedures," *Journal of the ACM*, vol. 17, no. 4, pp. 589–602, 1970.

[30] "NOAA NOS SOS, experimental, 1853-present, air temperature." [Online]. Available: https://coastwatch.pfeg.noaa.gov/erddap/index.html