Hindawi
Security and Communication Networks
Volume 2022, Article ID 9419676, 14 pages
https://doi.org/10.1155/2022/9419676

Research Article

A Decision Tree-Based Online Traffic Classification Method for QoS Routing in Data Center Networks

Shengxu Xie, Guyu Hu, Xiulei Wang, Changyou Xing, and Yaqun Liu

Command & Control Engineering College, Army Engineering University of PLA, Nanjing, China

Correspondence should be addressed to Xiulei Wang; xiuleiwang1988@126.com

Received 9 April 2022; Revised 20 September 2022; Accepted 10 November 2022; Published 22 November 2022

Academic Editor: Naghmeh Moradpoor

In data center networks, how to ensure the quality of service (QoS) of traffic effectively has become an important optimization goal. Traffic classification has been widely studied to guarantee network flow QoS since it provides a basis for the network performance level requirement of different flow types. However, traditional network traffic classification methods, which can only identify traffic type offline, have no ability for online classification and the in-time QoS guarantee. To solve this problem, we propose an online traffic classification for QoS routing (OCQR) framework based on programmable data plane in this paper. Firstly, in order to improve the accuracy of flow classification and reduce the impact on line-rate packet forwarding, the way of machine learning model offline training and online deployment is used to classify the network flow in real time. Based on the hardware foundation of programmable switch and the complexity of machine learning algorithm, a CART decision tree model is used to classify network flows. Secondly, in order to actually deploy and run the decision tree model in the programmable switch with limited computing and storage resources, we prune the decision tree model and optimize the leaf nodes to reduce the size of the decision tree and improve the classification recall rate. Finally, by building a P4 simulation environment, the recall rate of OCQR's classified scheduling of some application flows reaches more than 98%, which is consistent with the evaluation results of the offline dataset. At the same time, compared with the classification method based on SDN, OCQR does not require additional bandwidth occupation, so it has less impact on network performance.

1. Introduction

Network traffic type identification is a key issue in network performance improvement, and Internet traffic classification (ITC) has become an important research direction in the network field [1]. High-accuracy traffic classification plays an important role in intrusion detection [2], quality of service (QoS) guarantee [3], and traffic engineering [4]. In the data center, the data center network (DCN), which connects the computing resources and services, is growing in size and network throughput [5]. With the increase of network application types and service capabilities of data center network, how to ensure the QoS requirements of network flow has been a hot topic.

For unknown flows, network devices need to get their QoS requirements to guarantee the QoS of the flow. Because most of flows in the network do not use the type of service (ToS) field in the IP header, network forwarding devices cannot quickly classify the type of flows to provide corresponding QoS guarantees. Real-time traffic classification enables network operators to respond to flows of different network applications quickly and improve the QoS guarantees effectively. Traditional QoS guarantee mechanisms in which custom switches or routers are used through predefined mechanisms such as IntServ [6] or DiffServ [7], as well as routing policies, are not flexible enough. With the development of software-defined networking (SDN), the analysis of network flow's QoS requirements and routing strategies are moved to controller [8]. Although this improves the ability of QoS guarantee to a certain level, SDN switches need to upload the flow information to the controller for analysis and then install the corresponding flow entries to the switch to realize the reasonable routing of the flow. This not only increases the bandwidth consumption between the data plane and the control plane but also increases the processing delay of the packets.

As network programmability extends to the data plane, it becomes possible to perform logical operations within switches, such as in-network computing, which is well used with the support of programmable switches [9]. At the same time, because programmable switches (such as P4 switch) can maintain the state of network flow [10], the extraction of network flow features used for machine learning is possible. Based on these premises, this paper achieves the online classification of network traffic and reroutes the flow in time according to the QoS requirements of different types of flow. The main contributions of this paper are as follows:

(i) An online classification and identification algorithm of network flow is proposed based on decision tree. The dataset is formed by extracting the related features of the first *n* packets of each network flow from the original offline network traffic traces. Using the dataset, the decision tree model is trained offline, and then the model is used to realize the online real-time classification of network flow in programmable switch.

(ii) We have optimized the feature values that can be extracted by the programmable switches. At the same time, in order to reduce the complexity of decision tree and improve the classification recall, pruning strategy is used and a decision tree optimization reconstruction method is proposed.

(iii) We apply different application types of network flows with different QoS requirements as classification objectives to realize the reasonable routing of the flow by programmable switches. The network administrator can set the QoS requirement for flows of different network application types and then guarantee the QoS of the identified flow based on the network status information effectively.

The rest of this paper is organized as follows. Section 2 introduces the related work of traffic classification and the motivation of the research on online traffic classification for QoS routing based on programmable switch. The overall framework of OCQR is described in Section 3. Section 4 analyses the construction and optimization method of offline decision tree model. Section 5 describes the online classification and QoS routing mechanism implemented in programmable switch. Section 6 evaluates the performance of OCQR by setting up an environment based on P4 software switches. Finally, in Section 7, we summarized this paper.

2. Related Work and Motivation

2.1. Traffic Classification. The ITC method enables network operators to classify the network traffic and manage it according to the network state. In recent years, a variety of ITC technologies have been proposed, such as port-based classification, payload-based classification, and machine learning-based classification method [11].

Port-based traffic classification uses TCP/UDP port numbers to distinguish traffic types. For example, network traffic using well-known port number 80 is often considered WEB traffic. However, with the explosive development of network applications, a large number of network flows use random port numbers. At the same time, the application of new network technology (such as NAT protocol) makes the classification of port numbers become more and more inaccurate and eliminated [12, 13].

The payload-based classification method analyses the network flow load and constructs the state of session and application information from the content of each packet for application identification. However, due to the complexity of implementation, this kind of in-depth detection using packets has a high overhead to run on traffic identification devices, and it is difficult to analyse encrypted traffic.

With the rapid development of machine learning (ML) technology, ML has been widely studied and applied in the field of traffic classification [14]. By extracting the features such as average packet length and packet time interval of network flows, a network traffic classifier is constructed by using classical algorithms such as support vector machine (SVM) [15], hidden Markov model [16, 17], Naive Bayesian [18], decision tree [19], K-NN algorithm [20], and random forest [21]. Although machine learning has shown good performance in the classification of offline network flow data, in the traditional distributed computing network environment, network data collection and algorithms are difficult to carry out in traditional switches.

As the SDN controller has global view and powerful computing ability, SDN-based ITC for QoS guarantee has attracted the attention of researchers. Sun et al. [22] proposed a network flow classification method using a variety of machine learning to obtain its QoS requirements by accurately classifying the network flow and then planned the path of the flow to ensure QoS. Zheng et al. [8] used deep neural network to classify traffic data according to QoS requirements and made routing decisions according to transmission delay and bandwidth demand. In addition, because different video flows have different requirements in real time, bandwidth, and packet loss rate, Tang et al. [23] proposed a fine-grained flow classification method according to the fractal characteristics of flows. This method shows superior performance in fine-grained classification of video traffic. The ITC mechanism based on SDN can adopt complex machine learning models to improve the classification accuracy, so as to provide effective guarantee for the QoS of corresponding flows. However, in the process of flow feature extraction, the switch needs to upload packets to the controller, which increases the bandwidth occupation between the data plane and the control plane. At the same time, after flows are classified by the model, the corresponding flow entries need to be installed in switch to make decisions of QoS routing for network flows, which will lead to an increase in packet processing delay.

2.2. Motivation and Challenges. In order to reduce the occupation of network bandwidth by flow feature extraction and decrease packet processing delay, this paper considers using programmable data plane to carry out online classification of network traffic. Through deploying the classification model directly in the programmable switch, the network flows will be classified and identified without affecting the line-rate packet forwarding, and the network flows can be rerouted quickly to meet the QoS requirements of flows.

Although the programmable switches have arithmetic logic unit (ALU) which makes them computationally capable, in order to ensure line-rate forwarding of packets, floating point, exponential, and division operations that require multiple clock cycles are not supported. Therefore, it is not efficient to directly use the existing complex machine learning model to the programmable switch.

Decision tree, a rule-based supervised machine learning method, is one of the most popular classification techniques in machine learning and data mining. A decision tree model is represented by a sequence of branching statements, which continuously judges from the root node of the tree to reach the leaf node of the expected classification label. It is simple, interpretable, and efficient, so it is widely used in the classification of network packets [24]. Because programmable switches have state maintenance capability and support data size comparison in the pipeline of packet processing, the decision tree algorithm is naturally considered for real-time classification of network flows in programmable switches.

To reduce complex operations that need to be used in the training process of the decision tree model, the way of offline training and online deployment is used to classify the network flow in real time. After identifying the flow type, the programmable switch timely carries out the corresponding QoS routing for the flow according to the real-time link state of the network. However, at present, the computing and storage resources of programmable switches are limited [10]. The following challenges need to be solved in the actual deployment.

2.2.1. Extraction of Flow Features. Because complex operations cannot be performed in the packet processing pipeline in the programmable switch, it is necessary to consider whether the programmable switch can extract relevant features when generating the offline training dataset. For example, the mean, standard deviation, and other feature values commonly used for classification cannot be used.

2.2.2. Efficiency of Real-Time Classification. The online classification of network traffic flow undoubtedly increases the processing logic of programmable switch for the network flow and thus increases the forwarding delay of packets. Therefore, under the condition of ensuring certain classification accuracy, the complexity of the decision tree model needs to be optimized.

3. System Overview

The construction of the traffic QoS routing system based on online decision tree classification (OCQR system) is mainly divided into three steps: decision tree model offline training, programmable switch (P4 switch) configuration, and flow online classification and QoS routing. The overall system framework is shown in Figure 1.

3.1. Decision Tree Model Offline Training. The flow features extracted from the first *n* packets of the flow in raw traffic trace files are used as datasets to train the initial decision tree model. Since the initial decision tree model has high complexity and cannot run effectively in programmable switches with limited computing and memory resources, it is necessary to optimize it to obtain a suitable decision tree model for online classification. Therefore, in model training, we use the pruning strategy to reduce the size of decision tree and design the leaf node modification strategy to improve the recall rate of application type classification.

3.2. P4 Switch Configuration. Based on the original forwarding rules of the network, network administrator embeds the optimized decision tree model into P4 code logic and configures the switch that supports the online flow classification for QoS routing with the compiled P4 code.

3.3. Flow Online Classification and QoS Routing. In a programmable switch that supports flow classification, information such as the size and arrival time of the first *n* packets of a flow is recorded, and related features are extracted for classification. The forwarding rules for the first *n* packets use the default path, whereas for subsequent classified packets, the switch reroutes them to a path with QoS guarantee for forwarding. The mapping relationship of “QoS requirement-port” can be determined by obtaining the link parameters of the data center network through a method such as in-band network telemetry [25].

4. Offline Model Training

4.1. Flow Definition and Metrics

4.1.1. Flow Definition. Network flow is usually defined by five tuples, including IP source address, IP destination address, protocol type, source port, and destination port. At the same time, network flow has the difference between unidirectional flow and bidirectional flow. The unidirectional flow is strictly distinguished according to the source and destination address defined by five tuples. The bidirectional flow refers to the bidirectional packet sequence of the same connection with the same five tuples without distinguishing the strict source destination IP address or source destination port [26]. To better classify network flows, this paper regards bidirectional flows with the same five tuples as the same flow to make full use of the features of bidirectional flows.

4.1.2. Evaluation Metrics. For the evaluation of classification results, we use the confusion matrix commonly used in ML for analysis [27]. In the confusion matrix, four values as shown in Table 1 are used. For the flow of application type *i*, true positive (TP) represents the flow of the actual

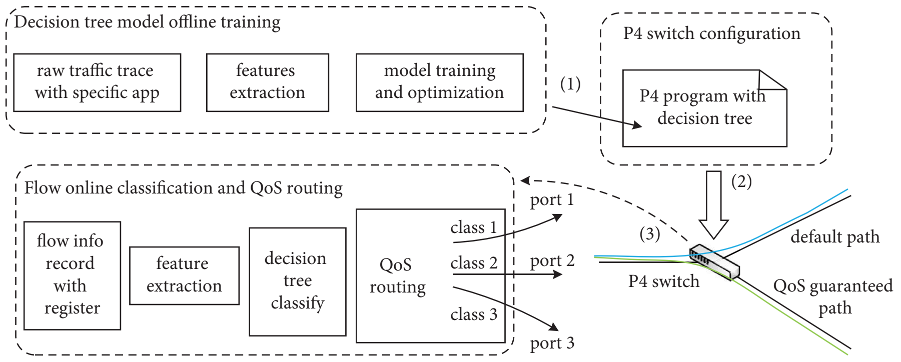

Decision tree model offline training
raw traffic trace with specific app
features extraction
model training and optimization
(1)
P4 switch configuration
P4 program with decision tree
(2)
Flow online classification and QoS routing
flow info record with register
feature extraction
decision tree classify
QoS routing
class 1
class 2
class 3
port 1
port 2
port 3
(3)
default path
P4 switch
QoS guaranteed path

Figure 1: OCQR system overall framework.

Table 1: Metrics in confusion matrix.

	Class i	Class $j \neq i$
Class i	True positive (TP)	False negative (FN)
Class $j \neq i$	False positive (FP)	True negative (TN)

application type i, and the prediction result is also i. False negative (FN) represents the flow of the actual application type i, and the prediction result is j. False positive (FP) represents the flow of the actual application type j, but the predicted result is i. True negative (TN) represents the flow of actual application to j, and the prediction result is also the flow of j.

To better evaluate the classifier, four metrics are used: *accuracy* (see formula (1)), *precision* (see formula (2)), recall (see formula (3)), and *F*-score (the weighted average of accuracy and recall,see formulas (4) and (5)).

$$Acc = \frac{TP + TN}{TP + FP + TN + FN}, \tag{1}$$

$$pre = \frac{TP}{TP + FP}, \tag{2}$$

$$Rec = \frac{TP}{TP + FN}, \tag{3}$$

$$F - \text{score} = \frac{\left(1 + \beta^2\right) pre \cdot Rec}{\beta^2 \cdot pre + Rec}, \tag{4}$$

where when $\beta = 1$, called F_1,

$$F_1 = \frac{2\, pre \cdot Rec}{pre + Rec}. \tag{5}$$

4.2. Determination of Features and Classification Objectives

4.2.1. Flow Feature Selection. To quickly classify the type of network flow for timely QoS routing, we extract the relevant features of the first n packets of the network flow. Common features used in traffic classification include time series, header, statistical features, and so on. In order to achieve the classification of flow types with different network applications, combined with the status maintenance capabilities of programmable switches, we selected some time series, statistical features, and bidirectional flow features as shown in Table 2. These features have achieved good results in the previous traffic classification based on machine learning [28]. In addition, many studies have shown that the related features of the first 20 packets of the flow have been proved to have good classification accuracy [29]. It is mainly noted that the larger n is, the later the switch completes flow classification and the worse the guaranteed effect of flow QoS is.

The standard formula for calculating standard deviation of packet length is shown in formula (6), where len_i represents the length of the i-th packet, and *mean_len* represents the average length of the first n packets.

$$std\_len = \sqrt{\frac{\sum_{i=1}^{n} \left(len_i - mean\_len\right)^2}{n-1}}. \tag{6}$$

Because the programmable switch does not support the operations such as floating point and division, we use square, multiplication, and other operations on both sides of formula (6) and then convert it to the following equation:

$$n(n-1) \cdot std\_len^2 = n \cdot \sum_{i=1}^{n} len_i^2 - \left(\sum_{i=1}^{n} len_i\right)^2. \tag{7}$$

At this point, the right side of formula (7) contains only the addition, subtraction, and multiplication operations that can be performed in a programmable switch. The standard deviation of the packet time interval is calculated in the same way.

Proposition 1. *The classification result of the decision tree will not change when applying square, multiplication (multiply by a positive number), addition, or subtraction operation on a continuous positive feature.*

Proof. Assume that the feature A has the following n values $\{a_1, a_2, \ldots, a_n\}$, and $0 < a_1 < a_2 < \cdots < a_n$. Now, $f(a)$ operation is performed on the values of feature A. When $f(a) = k \times a^2 + m$, $(k > 0)$, there is $0 < f(a_1) < f(a_2) < \cdots < f(a_n)$ because $0 < a_1 < a_2 < \cdots < a_n$ and $f(a)$ monotonically increases in the $[0, +\infty]$ interval. Suppose that the decision tree T takes the feature A as a splitting node, that is, a value a'

Table 2: Extracting features.

Type	Features
Time series	Packet arrival time interval (average, maximum, minimum, and standard deviation)
Statistical features	Packet length (average, maximum, minimum, and standard deviation)
Bidirectional flow features	The difference between the sum of forward and reverse packet lengths

of the feature A is used as the splitting point, and the dataset D is divided into two parts: D_1 $(a \leq a')$ and D_2 $(a > a')$. At this time, after the $f(a)$ operation of all the values of the feature A, taking the $f(a')$ as the splitting point, the dataset D can still be divided into D_1 $(f(a) \leq f(a'))$ and D_2 $(f(a) > f(a'))$. Therefore, the classification effect of decision tree T is not affected.

4.2.2. Classification Objectives. Traditional network traffic classification mostly takes specific applications as the classification objectives, such as works [30, 31]. However, too many classification objectives will require more complex decision tree model to classify effectively, which makes the effectiveness of online traffic classification worse. In computer networks, QoS requirement can be determined by bandwidth, end-to-end delay, packet loss, and jitter [32]. At the same time, the network flow of the same application type has similar QoS requirements. According to the existing QoS framework protocol (IETF, NGN and ITU-T, etc.), the QoS requirements of different flow types are obtained through analysis [33], as shown in Table 3. We use these five application types as classification objectives to analyse network traffic.

4.2.3. Dataset Generation. Since the relevant statistical information of the first n packets of the flow is used as the classification feature to achieve the real-time classification, some existing traffic classification datasets such as Moore [34] use the relevant information of the whole flow as the feature. At the same time, some datasets, such as Mirage [28], do not have do not have packet traces of original traffic, so it is impossible to conduct a valid online classification effect evaluation. To this end, considering comprehensively, we use the original traffic traces (pcap files) of some applications of ISCX's VPN-nonVPN dataset [35, 36] and Tor-nonTor dataset [37, 38] for experimental analysis. The applications corresponding to each application type are shown in Table 4. By extracting the relevant information of the first n packets of each type of original network flow, calculate the features corresponding to Section 4.2.1 to form the experimental dataset.

4.3. Decision Tree Construction

4.3.1. Algorithm Selection. During the decision tree model training, the use of different splitting feature selection algorithms will result in different decision trees, such as ID3 algorithms, C4.5 algorithms, and CART algorithms, as shown in Table 5.

(1) ID3 Algorithm. The selection of splitting features is determined by information gain. The feature with the largest information gain is selected as the splitting node and greedily traverses the selection from top to bottom to form the decision tree space. Information gain indicates the difference of data entropy before and after using the feature A to classify data, that is,

$$\text{Gain}(D, A) = H(D) - H(D|A). \tag{8}$$

In formula (8), $H(D)$ denotes the information entropy of dataset D before feature A is selected for classification, and $H(D|A)$ denotes the information entropy of dataset D after feature A is selected for classification. When the information gain of feature A is greater than that of all other features, feature A is selected as the splitting node.

(2) C4.5 Algorithm. The selection of splitting features is determined by information gain rate. The information gain ratio $\text{Gain}_R(D, A)$ represents the ratio of the information gain $\text{Gain}(D, A)$ of the dataset D on feature A to the entropy $H_A(D)$ of the dataset D about the value of feature A, that is,

$$\text{Gain}_R(D, A) = \frac{\text{Gain}(D, A)}{H_A(D)}, \tag{9}$$

where

$$H_A(D) = -\sum_{i}^{n} \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|}. \tag{10}$$

In formula (10), n is the number of values of feature A. C4.5 first selects the features with higher information gain than the average value from all candidate splitting features and then selects the feature with the highest gain rate as the node of the classification.

(3) CART Algorithm. The selection of splitting features is determined by Gini coefficient, and the optimal binary splitting point of the feature is determined at the same time. Gini coefficient is calculated as follows:

$$\text{Gini}(p) = \sum_{k=1}^{K} p_k (1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2, \tag{11}$$

where K represents the category number of a feature column and p_k represents the probability of the k-th category. For dataset D, when a value of feature A is used to divide the dataset into D_1 and D_2, under the condition of feature A, the Gini coefficient is

$$\text{Gini}(D, A) = \frac{|D_1|}{|D|} \text{Gini}(D_1) + \frac{|D_2|}{D} \text{Gini}(D_2). \tag{12}$$

Table 3: Examples of QoS requirements for network flows of various application types.

Application types	Delay (ms)	Bandwidth (Kbps)	Packet loss rate	Jitter (ms)
Real-time video	≤50	≥128	$\leq 10^{-3}$	≤50
Multimedia streaming	≤100	≥128	$\leq 10^{-1}$	≤100
Data transfer	≤250	≥64	$\leq 10^{-1}$	≤100
Voice	≤50	≥16	$\leq 10^{-2}$	≤10
Text	≤500	≥32	$\leq 10^{-3}$	≤100

Table 4: Applications corresponding to each application type in the dataset.

Application types	Applications	Sample size ($n = 5$)	Sample size ($n = 10$)	Sample size ($n = 15$)
Real-time video (rtvideo)	Skype video	323	322	317
Multimedia streaming (mstreaming)	YouTube, Vimeo	854	844	796
Data transfer (dtransfer)	FTP, P2P	2963	2961	2953
Voice	Skype voice, Facebook voice, Hangout voice	2254	2252	2247
Text	Browsing	25103	15683	12345

Table 5: Decision tree algorithm comparison.

Algorithm	Tree structure	Split feature selection
ID3	Multi-branch tree	Information gain
C4.5	Multi-branch tree	Information gain ratio
CART	Binary tree	Gini

The feature with the largest gain of Gini coefficient will be selected as the node attribute of the decision tree.

ID3 algorithm cannot deal with continuous features and is not applicable to the flow classification having continuous features. The decision tree model generated by C4.5 algorithm is a multi-tree, that is, a parent node can have multiple child nodes, and its operation efficiency is much lower than that of the binary tree model. In addition, although ID3 algorithm and C4.5 algorithm can mine as much information as possible in the learning of the training sample set, the branches and scale of the decision tree model they generated are relatively large, while the dichotomy adopted by CART algorithm can simplify the scale of the decision tree. Therefore, considering the online flow classification in programmable switches with limited computing and storage resources, the CART algorithm was selected as the decision tree model algorithm of OCQR.

To verify the classification effect of the first n packet-related features in the decision tree model based on CART algorithm, 75% of the datasets are used as the training set to train the decision tree model, and the remaining 25% of the datasets are used as the validation set to evaluate the trained decision tree model. The results are shown in Table 6. In Table 6, the tree scale is the number of all nodes in the tree, the maximum depth is the maximum length from root to leaf of the tree, and the average depth is the average length from root node to all leaf nodes of the tree.

As can be seen from Table 6, except for the poor classification effect of multimedia stream, the classification precision, recall, and F_1 score of the other four application types have exceeded 88%, and the classification accuracy reached 96%. On the scale of the tree, with the same

Table 6: Classification effect using the relevant features of the first n packets.

	$n = 5$			$n = 10$			$n = 15$		
	Pre	*Rec*	F_1	*Pre*	*Rec*	F_1	*Pre*	*Rec*	F_1
Rtvideo	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Mstreaming	0.76	0.69	0.72	0.70	0.70	0.64	0.72	0.65	0.68
Dtransfer	0.89	0.90	0.90	0.88	0.92	0.90	0.91	0.95	0.93
Voice	0.91	0.92	0.91	0.94	0.92	0.93	0.97	0.97	0.96
Text	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98
Acc		0.96			0.96			0.96	
Tree scale		1499			1285			1043	
Maximum depth		26			26			21	
Average depth		14			13			12	

accuracy, the tree size is the smallest when $n = 15$, indicating that the difference between the application flow types is most significant when $n = 15$. At the same time, from the classification effect of Rtvideo and Text, nearly 100% of the classification index values indicate that there is a certain possibility that the decision tree model is overfitted in classification.

4.3.2. Model Pruning. As a method of reducing overfitting for decision trees, pruning strategy not only improves the classification effect but also reduces the complexity of decision trees by actively removing some branches [39]. The pruning operation consists of prepruning and postpruning [40]. Prepruning restricts the size of a decision tree by giving the conditions under which it grows when building a decision tree model. Postpruning improves the classification accuracy of decision trees by pruning some subtrees when the decision tree is completely built.

Due to the limited computing and storage resources of programmable switch and to meet the requirements of line-rate forwarding, the processing logic of network flow should be as simple as possible. To reduce the impact of decision tree algorithm running in programmable switches on packet processing delay, it is necessary to reduce the size of decision tree as much as possible.

(1) Prepruning. Since the model trained by the decision tree using CART algorithm is a binary tree, the tree depth is the maximum comparison number for classifying a sample. When a programmable switch implements classification, it can directly use the if-else statements to judge at each splitting node and then get the classification type at leaf node. Therefore, by using the prepruning strategy to limit the depth of the tree, it can effectively limit the maximum time consumption for running the decision tree in the programmable switch.

When training the decision tree model for OCQR, the parameters of prepruning are as follows: the maximum tree depth is 10, the minimum number of samples of each leaf node is 3, and the minimum number of samples of each split node is 4.

(2) Postpruning. Using postpruning to remove some subtrees can not only alleviate the overfitting phenomenon but also reduce the complexity and the average depth of the tree. For the programmable switch that executes the decision tree model, reducing the average depth of the tree can effectively reduce the average time consumption and computing overhead of flow classification.

Cost complexity pruning (CCP) [41] is used as postpruning algorithm for OCQR to further prune the decision tree model after using the prepruning strategy. By calculating the cost complexity pruning path of the decision tree, the decision tree model after each pruning in the path can be obtained. Through analysis, it is found that when the decision tree size is greater than 70, the accuracy of the classification model can be more than 90%. Therefore, when using the postpruning strategy, we choose a decision tree model with size as small as possible but not less than 70.

We still use 75% of the dataset as the training set for the decision tree model training by using the prepruning strategy and then optimize the model by using the postpruning strategy. The remaining 25% of the dataset is used as validation set to validate the model after pruning, and the results are shown in Table 7.

Comparing the classification results in Table 6, it can be seen that the decision tree size is more than one thousand nodes before pruning and less than 80 nodes after pruning. Although the decision tree model has some underfitting after a lot of pruning, the decline of each evaluation metrics of its classification is relatively small.

Table 7: Classification effect after pruning of training model for correlation features of first n packets.

	$n = 5$			$n = 10$			$n = 15$		
	Pre	*Rec*	F_1	*Pre*	*Rec*	F_1	*Pre*	*Rec*	F_1
Rtvideo	0.89	0.68	0.77	0.82	0.96	0.88	1.0	0.90	0.95
Mstreaming	0.68	0.10	0.18	0.42	0.28	0.34	0.70	0.46	0.56
Dtransfer	0.66	0.70	0.68	0.79	0.76	0.78	0.84	0.78	0.80
Voice	0.71	0.79	0.75	0.85	0.85	0.85	0.95	0.85	0.90
Text	0.97	0.98	0.94	0.97	0.98	0.98	0.93	0.98	0.95
Acc		0.93			0.93			0.91	
Tree scale		77			71			79	
Maximum depth		11			11			11	
Average depth		7			7			7	

4.3.3. Leaf Node Modification. In the QoS routing process of network flows, if the flow of a network application type is misclassified into other network application types, it may lead to a serious impact. For example, if Mstreaming-type flows, with high network bandwidth demand, are misclassified as Voice-type flows, it will be rerouted to the link with low latency but insufficient bandwidth guarantees. This will not only make the flow fail to obtain QoS guarantee but also affect the performance of the link and even the whole network after rerouting. It is especially important to improve the recall of classification so that the flow of a certain application type can be classified as accurately as possible. However, on the one hand, there are packet loss, delay jitter, and packet fragmentation in the network, resulting in the deviation between the extracted features and the actual features of the flow; on the other hand, as shown in Table 7, the decision tree model obtained by offline training has certain underfitting due to the influence of the training dataset and pruning.

To this end, we optimized and reconstructed the trained decision tree model. As shown in Figure 2, by evaluating the Gini coefficient for leaf node classification (see formula (11)), when the Gini coefficient value exceeds 0.5, its type is considered as a general flow. Such flows are processed by default paths to ensure basic QoS requirements and to stabilize network performance. Assume that the number of the flow of application type i that can be classified as general flow is Nor (Normal) in the modified decision tree model. At this time, the formula for classification recall rate of application type i is changed to

$$Rec = \frac{TP}{TP + FN - \text{Nor}}. \tag{13}$$

We improved three decision tree models in Table 7 and evaluated the modified decision tree model with the same proportion of validation sets. Figure 3 shows the traffic classification recall of five different network application types under different n conditions with the model before pruning, after pruning, and after modification, respectively. As can be seen from Figure 3, after only pruning the decision tree model, the traffic classification recall of different network application types has decreased to a certain extent because the scale of the decision tree has been reduced to about 1/20 of the original, making the classification effect worse. However, after modification of the pruned decision tree model, the traffic classification recall of different network application types has been greatly improved. This ensures that even if some flows are not classified accurately, the network administrator can still use the default forwarding path and guarantee the basic QoS of this part. Thus, the decision tree models are all modified in subsequent experiments.

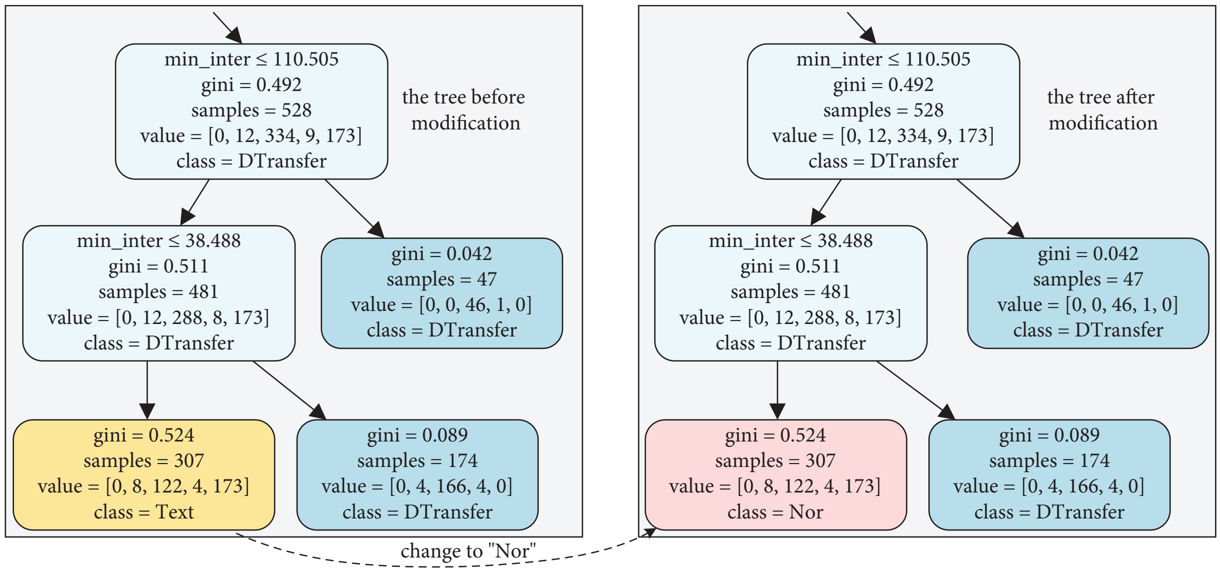
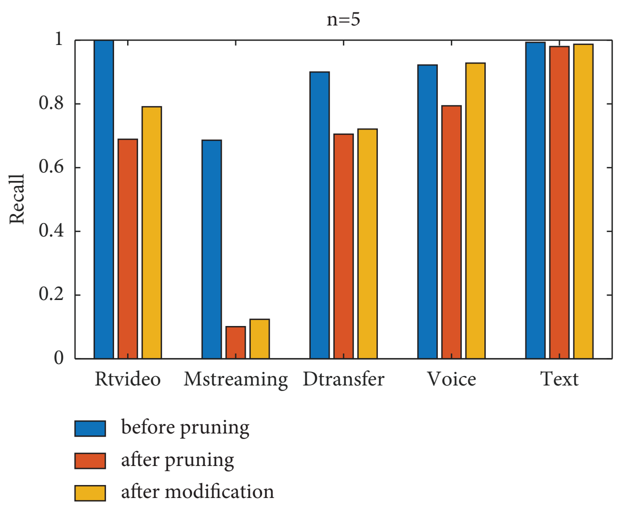
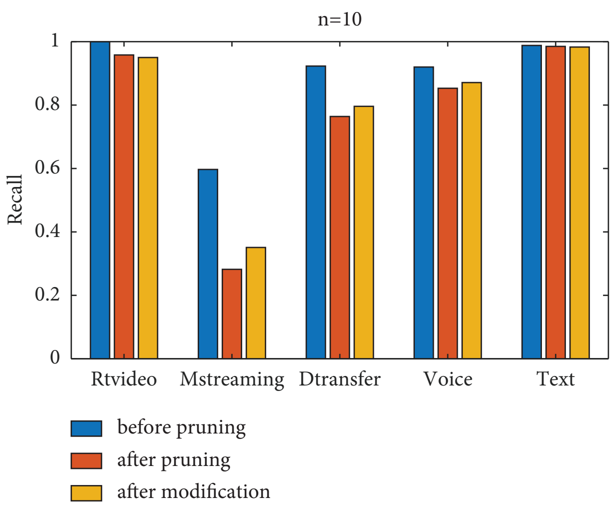


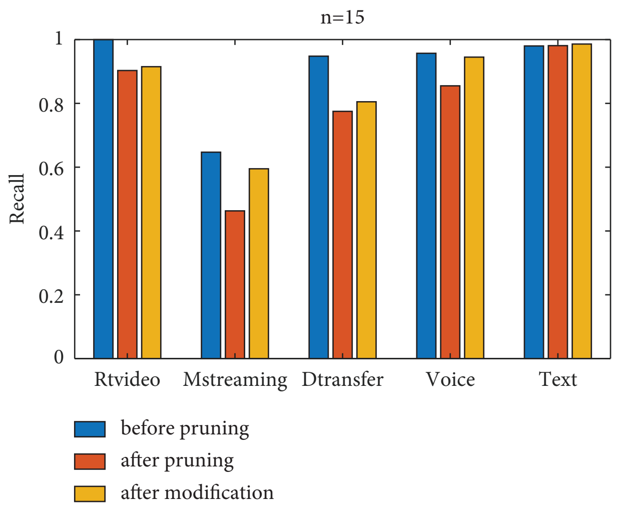
FIGURE 2: Leaf node modification.



(a)



(b)



(c)

FIGURE 3: Comparison of classification recall rate of decision tree model. (a) $n = 5$. (b) $n = 10$. (c) $n = 15$.

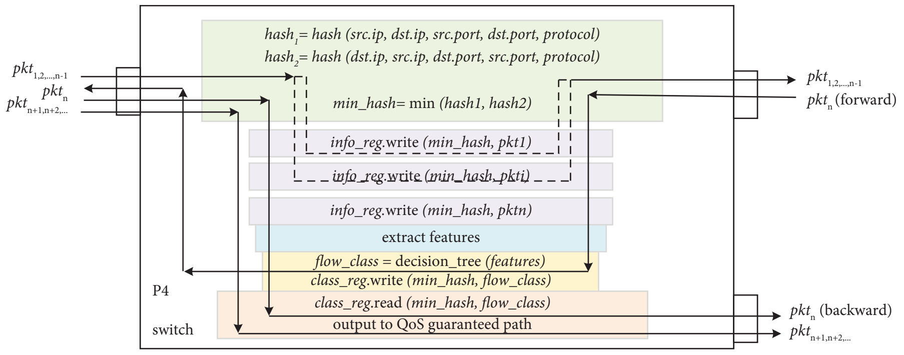

hash1= hash (src.ip, dst.ip, src.port, dst.port, protocol)
hash2= hash (dst.ip, src.ip, dst.port, src.port, protocol)
min_hash= min (hash1, hash2)
info_reg.write (min_hash, pkt1)
info_reg.write (min_hash, pkti)
info_reg.write (min_hash, pktn)
extract features
flow_class = decision_tree (features)
class_reg.write (min_hash, flow_class)
class_reg.read (min_hash, flow_class)
output to QoS guaranteed path
P4
switch
pkt1,2,...,n-1
pktn
pktn+1,n+2,...
pktn (forward)
pktn (backward)

FIGURE 4: Online traffic classification and QoS routing logic in programmable switch.

5. Online Flow Classification Implementation

The trained and modified decision tree model is embedded into the code logic of programmable switch to realize the goal of online classification and reasonable QoS routing of network flow. The packet processing logic of flow is shown in Figure 4. The first n packets of the network flow are indexed by the smallest of the forward and reverse five-tuple hash value (*min_hash*), and the packet size, arrival time, forward and reverse packet size, and other information are recorded, respectively. After extracting the relevant information of n packets, the feature values used in decision tree classification are calculated, and then the flow is classified according to the decision tree model. After identifying the application type of the flow, *min_hash* is used as an index to record the type in the switch, so that the subsequent packets of the flow can be directly recognized according to the *min_hash* value and then rerouted to the path that can meet the QoS requirements (see Algorithm 1).

6. Performance Evaluation

6.1. Experimental Environment Settings. In order to evaluate the effect of OCQR, we use Mininet network simulation tool [42] and P4 software programmable switch based on bmv2 [43] to build a simulation environment on Dell P580 workstation (Intel Core i9-10900x processor, 32 GB memory, and Ubuntu 18.04 OS). Mininet and bmv2 are supported by P4 language consortium, which can reproduce the real functions of P4 environment, so they are widely used in P4 verification and evaluation [44].

Considering the performance limitations of software switch and the QoS requirements of different application flows shown in Table 3, we construct our experimental topology environment by extracting all shortest paths of a pair of edge switches from 6-ary fat-tree topology [45] which is commonly used in the data center network, as shown in Figure 5.

In Figure 5, the parameter configuration of all six shortest paths from S_{11} to S_{12} is shown in Table 8. The default forwarding rule of the switch is to select the shortest path for forwarding according to the destination address. For example, the default forwarding path from host h_1 to host h_4 is S_{11}-S_7-S_1-S_9-S_{12}.

The edge switches S_{11} and S_{12} realize classification, marking, and rerouting of flows, and the intermediate

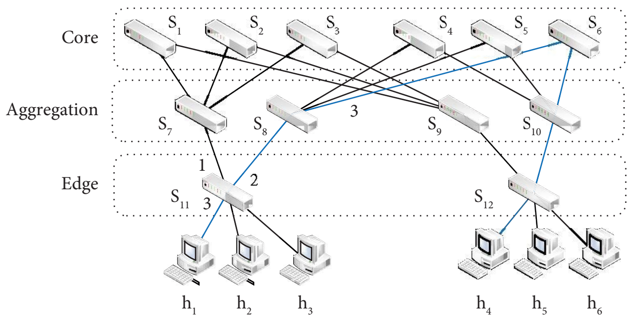


FIGURE 5: Experimental topology.

switch reroutes the flow according to the differentiated services code point (DSCP) marked in the TOS field of the packet IP header by edge switches. Take the packet of Text-type flow as an example. For the flow sent from h_1 to h_4, if the switch S_{11} recognizes the packet that belongs to Text-type flow, first modify the TOS field (DSCP = 4) and then forward it to port 2. After receiving the packet and recognizing it belongs to Text-type flow based on the DSCP value, switch S_8 directly forwards it from port 3, as shown in the blue path in Figure 5.

6.2. Flow Generation. To effectively evaluate the online classification effect of the decision tree model, we consider using tcpreplay [46] network tool to replay the original packets of different types of network traffic flows from the raw traffic traces used to extract features during the model offline training.

Different flows in the original traffic trace have different IP address pairs of source and destination. If the IP address-based routing rules are used, the forwarding of all flows cannot be satisfied. At the same time, the extraction of classification features contains the bidirectional flow information. If the original traffic trace is used for bidirectional replay directly, the link delay and jitter will result in a large difference between the bidirectional flow features collected at the edge switch (S_{11}, S_{12}) and the features extracted directly from the network flow packet. To this end, we process the network and original traffic traces as follows:

```
Input: n: number of packets used for flow classification
(1)  while packet p arriving do
(2)    hash1 = hash (p.src.ip, p.dst.ip, p.src.prot, p.dst.port, protocol);
(3)    hash2 = hash (p.dst.ip, p.src.ip, p.dst.port, p.src.prot, protocol);
(4)    min_hash = min (hash1, hash2);
(5)    flow_class = class_reg.read (min_hash);
(6)    if flow_class = = 0 then
(7)      packet_count = count_reg.read (min_hash);
(8)      if packet_count < n then
(9)        len_info_reg.write (min_hash, p.length);
(10)       time_info_reg.write (min_hash, current_time);
(11)     elif packet_count = = n then
(12)       packet_lengths = len_info_reg.read(min_hash).append(p.length);
(13)       packet_times = time_info_reg.read (min_hash).append (current_time);
(14)       features = extract (packet_lengths, packet_times);
(15)       flow_class = decision_tree (features);
(16)       class_reg.write (min_hash, flow_class);
(17)  flow_class = class_reg.read (min_hash);
(18)  if flow_class > 0 and flow_class < 6 then
(19)    output the packet to the port with the QoS guaranteed link;
(20)  else
(21)    output the packet to the default port with the link in the shortest path;
```

Algorithm 1: Online classification and QoS routing.

(1) The shortest path algorithm is used to calculate the port p_j in the shortest path of the switch S_i to the destination host, and the default port forwarding matching rule is the MAC address (dst_mac$_k$) of the destination host h$_k$, which is called "dst_mac$_k$: p_j."
(2) The path performance parameter setting depends on the link between switches S_1–S_6 and switches S_9-S_{10}.
(3) Using the auxiliary tool tcprewrite of tcpreplay, modify the source MAC address of all packets to the MAC address of h_1 and modify the destination MAC address of all packets to the MAC address of h_4 in the experimental environment.
(4) The original packet is always sent from one end (host h_1) to the other end (host h_4).

At this time, after the original traffic traces are replayed by the host h_1, the packet received at port 3 of switch S_{11} can obtain the same time series and statistical features as those in the source pcap files.

6.3. Experimental Results and Analysis

6.3.1. Effect of Pruning. To evaluate the effect of pruning on the network, the processing time of packets in the switch is analysed. Using the decision tree model constructed at $n = 10$, we calculate the average processing time of the first 20 packets of 1000 network flows of different application types (200 flows for each type) at switch S_{11}, as shown in Figure 6. As can be seen from Figure 6, in the case of OCQR with pruning, the processing delay of the 10th packet of the flow is about 300 μm less than OCQR without pruning. This is because when the 10th packet arrives at the switch, the switch needs to execute different number of if-else statements with different decision tree depths to get the type of flow. The average depth of the decision tree model after pruning is smaller than that without pruning, which reduces the amount of computation that the switch needs to perform and thus reduces the processing delay. In addition, when the 10th packet arrives, the switch needs not only to extract the packet information but also to calculate the relevant feature values and identify the application type through the decision tree model. Because the computational resource occupation of the decision tree model only contains a part of it, the delay difference between packet processing with or without pruning is relatively small. At the same time, due to the poor performance of P4 software switch, the packet processing delay is still about 1.4 ms even without the OCQR method.

Furthermore, in Figure 6, when using OCQR, the processing delay of the first 10 packets is higher than that of the subsequent arrival packets, and the processing delay of the 10th packet is nearly three times higher. This is because when the first 10 packets arrive at the switch, the switch needs to maintain the relevant information, and additional operations such as feature extraction and decision tree classification are required in the processing of the 10th packet. After classifying the flow, subsequent packets of the flow only need to confirm which application type they belong to based on the hash value of the five-tuple, so the processing delay is significantly shorter.

Secondly, we counted the size of P4 source and compiled files, as shown in Figure 7. As can be seen from Figure 7, whether P4 source files or compiled files, when using OCQR with pruning, the file size is reduced by more than one order of magnitude than when using OCQR without pruning. This greatly reduces the consumption of OCQR on the storage resources of programmable switches.

Table 8: Path parameters and corresponding flow types.

Path	Path parameters	Flow type corresponding to QoS
S_{11}-S_7-S_1-S_9-S_{12}	Delay = 0, bandwidth = default, packet loss rate = 0	Normal
S_{11}-S_7-S_2-S_9-S_{12}	Delay = 10 ms, bandwidth = 12.8 Mbps, packet loss rate = 10^{-4}	Rtvideo
S_{11}-S_7-S_3-S_9-S_{12}	Delay = 20 ms, bandwidth = 12.8 Mbps, packet loss rate = 10^{-2}	Mstreaming
S_{11}-S_8-S_4-S_{10}-S_{12}	Delay = 50 ms, bandwidth = 6.4 Mbps, packet loss rate = 10^{-2}	Dtransfer
S_{11}-S_8-S_5-S_{10}-S_{12}	Delay = 10 ms, bandwidth = 1.6 Mbps, packet loss rate = 10^{-3}	Voice
S_{11}-S_8-S_6-S_{10}-S_{12}	Delay = 100 ms, bandwidth = 3.2 Mbps, packet loss rate = 10^{-4}	Text

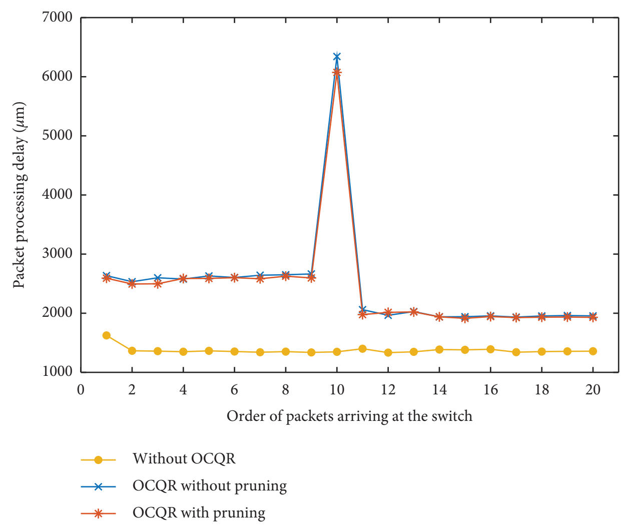


Figure 6: Effect of pruning on packet processing delay.

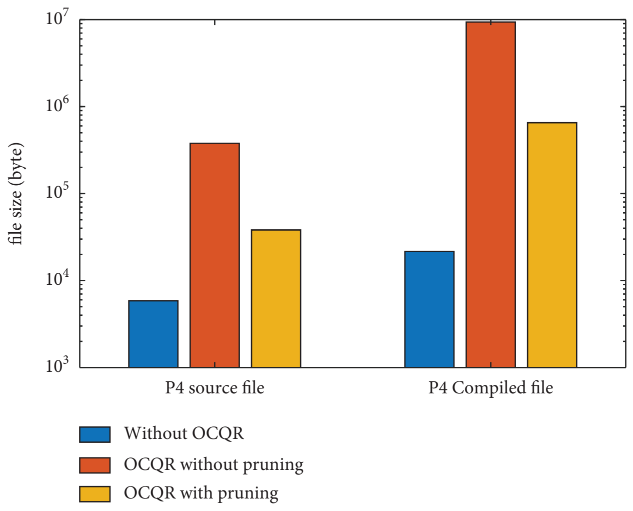
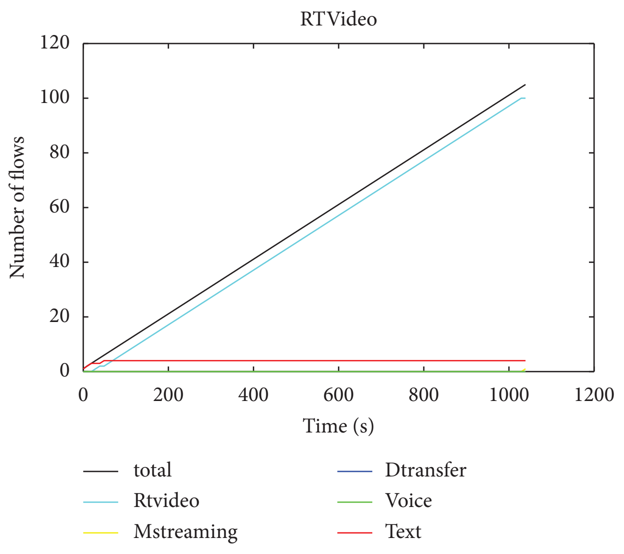


Figure 7: Effect of pruning on P4 code file size.

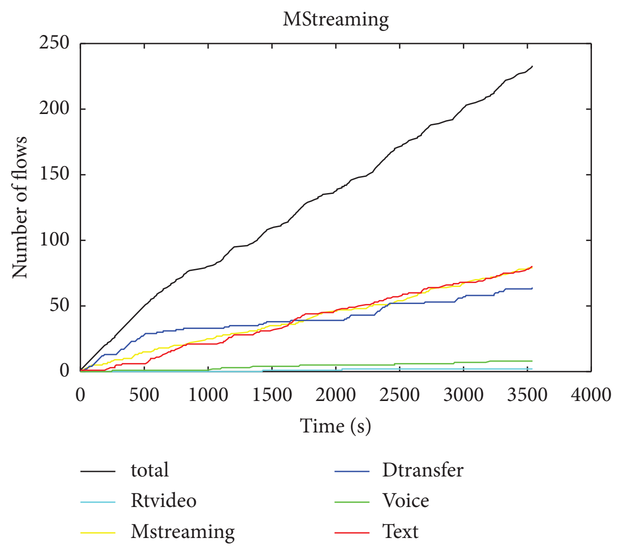
6.3.2. Online Classification QoS Routing. In order to evaluate the QoS routing effect of OCQR, we choose the final decision tree model obtained by pruning the features of the first 10 packets for experiments and realize the classification and rerouting of the network traffic in the programmable switch. The experiment replays some of the original traffic flows of different application types and monitors the actual classification and QoS routing result of each application type of network traffic by countering the flow numbers arrived at switches S_2–S_6. The results are shown in Figure 8. As can be seen from Figure 8, the classification and QoS routing effect of Rtvideo and Text flows are better, and the classification and QoS routing effect of Mstreaming are the worst. At the same time, by comparing the results of QoS scheduling with each application type in Figure 8 and the recall rate of the decision tree model in Figure 3, it can be found that when $n = 10$, the classification effect of the model in the offline dataset is close to the classification effect of QoS routing for different application types after the OCQR is actually deployed in the programmable switch. It can be said that OCQR can be normally deployed in the data center network based on the programmable switch and run effectively, and the effect of QoS routing scheduling of different application flows is consistent with expectations.

6.3.3. Comparison with SDN-Based Methods. When the SDN architecture is used to implement the online classification of network flow, complex classification models can be run in the SDN controller. However, the SDN-based method needs to upload the packets of the network flow to the controller, which not only increases the bandwidth occupation between the data plane and the control plane but also increases the processing delay of the packets. For this reason, we experimentally evaluated the bandwidth utilization between data plane and control plane of SDN-based classification methods and OCQR methods, as shown in Figure 9. From Figure 9, the SDN-based method takes up more bandwidth utilization as the n value increases, while the OCQR takes up 0. This is because when SDN-based methods are used, the larger the n value is, the more the packets need to be uploaded to the controller, while OCQR methods do not take up bandwidth because they extract features and classify flow directly within the P4 switch.

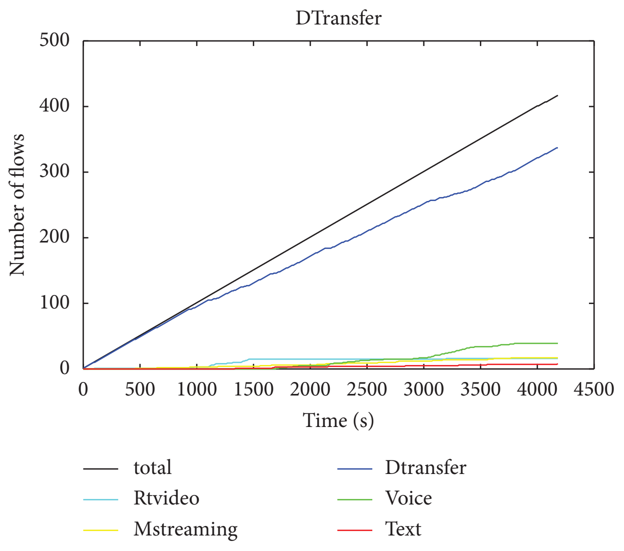
Due to the poor performance of the P4 software switch used in the experiment, it is impossible to forward packets with line-rate, and the delay between the data plane and the control plane cannot be quantified clearly. Therefore, the packet processing delay with the SDN-based method and OCQR method is not evaluated in our experiment.



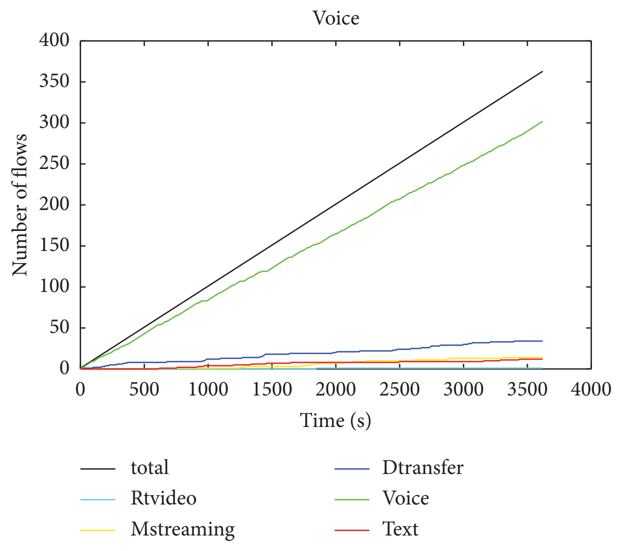
(a)



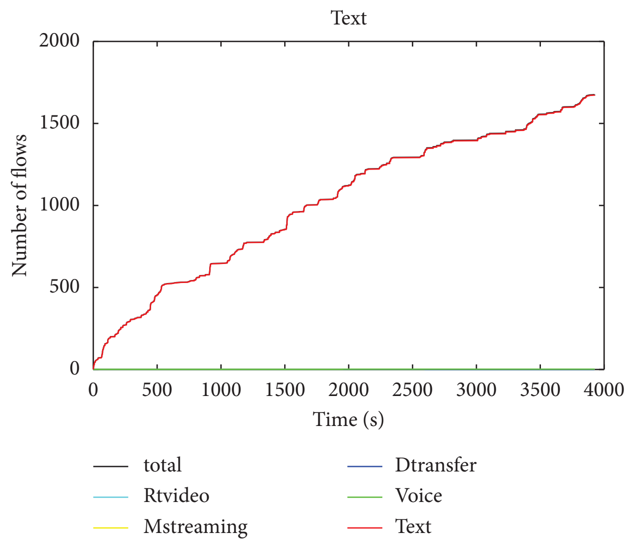
(b)



(c)



(d)



(e)

FIGURE 8: Traffic online identification for QoS routing results. (a) Real-time video. (b) Multimedia streaming. (c) Data transfer. (d) Voice. (e) Text.

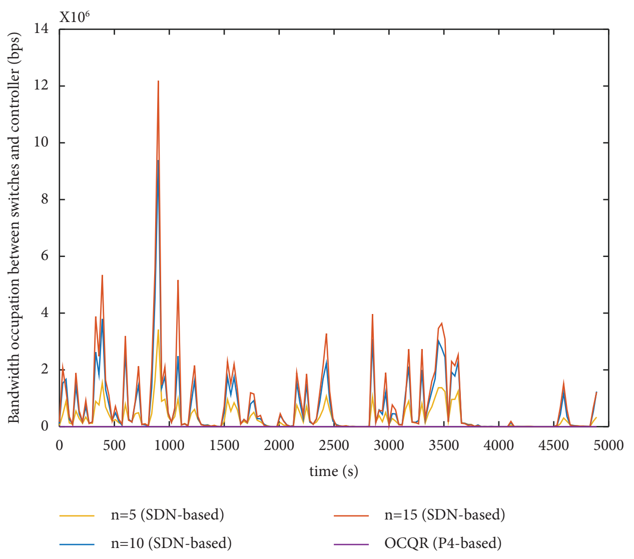


Figure 9: Comparison of bandwidth utilization between data plane and control plane in real-time classification based on SDN and P4 switches.

7. Conclusion

In this paper, we propose an online traffic classification for QoS routing method, OCQR, which runs in data center networks based on programmable switch. With the offline-trained decision tree model, OCQR classifies flows using the features extracted from the first n packets of different application flows and reroutes the flows according to the corresponding QoS requirements. Firstly, we transform some classification features to ensure that the programmable switch can effectively extract the relevant features required by the decision tree model and reduce the impact on packet processing delay and optimize the decision tree model using the pruning strategy and leaf node modification strategy. Secondly, to achieve the QoS guarantee of flows, we reroute flows according to the corresponding QoS requirements and the performance of each link in data center network. Finally, OCQR is validated experimentally based on the simulation environment built with P4 software switch.

The decision tree can only classify the existing traffic types in the training dataset, which means that the flow of general application types that are not adequately trained cannot be effectively processed. Therefore, to improve the recall of traffic classification in our future work, it is necessary to find a more appropriate classification method to identify flow type with a certain confidence (such as Gaussian Bayesian classification).

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported in part by the Natural Science Foundation of Jiangsu Province (grant no. SBK2020043435) and the National Natural Science Foundation of China (no. 62172432).

References

[1] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, “Towards the deployment of machine learning solutions in network traffic classification: a systematic survey,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2019.
[2] Y. Zeng, H. Gu, W. Wei, and Y. Guo, “$Deep-Full-Range$: a deep learning based network encrypted traffic classification and intrusion detection framework,” *IEEE Access*, vol. 7, pp. 45182–45190, 2019.
[3] J. A. Caicedo-Muñoz, A. Ledezma Espino, J. C. Corrales, and A. Rendón, “QoS-Classifier for VPN and Non-VPN traffic based on time-related features,” *Computer Networks*, vol. 144, pp. 271–279, 2018.
[4] V. Punitha and C. Mala, “Traffic classification for efficient load balancing in server cluster using deep learning technique,” *The Journal of Supercomputing*, vol. 77, no. 8, pp. 8038–8062, 2021.
[5] T. Chen, X. Gao, and G. Chen, “The features, hardware, and architectures of data center networks: a survey,” *Journal of Parallel and Distributed Computing*, vol. 96, pp. 45–74, 2016.
[6] R. Braden, D. Clark, and S. Shenker, “Integrated services in the Internet architecture: an overview,” IETF, 1994, https://datatracker.ietf.org/doc/html/rfc1633.
[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, “An architecture for differentiated services,” IETF, 1998, https://www.rfc-editor.org/rfc/rfc2475.
[8] W. Zheng, M. Yang, C. Zhang et al., “Application-aware QoS routing in SDNs using machine learning techniques,” *Peer-to-Peer Networking and Applications*, vol. 15, no. 1, pp. 529–548, 2022.
[9] A. Sapio, I. Abdelaziz, A. Aldilaijan, M. Canini, and P. Kalnis, “In-network computation is a dumb idea whose time has come,” in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks (HotNets ’17)*, pp. 150–156, New York, NY, December 2017.
[10] X. Zhang, L. Cui, K. Wei, F. P. Tso, Y. Ji, and W. Jia, “A survey on stateful data plane in software defined networks,” *Computer Networks*, vol. 184, Article ID 107597, 2021.
[11] M. B. Umair, Z. Iqbal, M. Bilal, T. A. Almohamad, J. Nebhen, and R. M. Mehmood, “An efficient internet traffic classification system using deep learning for IoT,” *CoRR*, vol. abs/2107, Article ID 12193, 2021.
[12] A. Madhukar and C. L. Williamson, “A longitudinal study of P2P traffic classification,” in *Proceedings of the 14th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2006)*, pp. 179–188, Monterey, CA, USA, September 2006.
[13] A. W. Moore and K. Papagiannaki, “Toward the accurate identification of network applications,” in *Proceedings of the Passive and Active Network Measurement, 6th International Workshop (PAM 2005)*, pp. 41–54, Boston, MA, USA, March 2005.
[14] T. T. T. Nguyen and G. J. Armitage, “A survey of techniques for internet traffic classification using machine learning,” *IEEE Commun. Surv. Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[15] R. Yuan, Z. Li, X. Guan, and L. Xu, "An SVM-based machine learning method for accurate internet traffic classification," *Information Systems Frontiers*, vol. 12, no. 2, pp. 149–156, 2010.

[16] C. Yin, S. Li, and Q. Li, "Network traffic classification via HMM under the guidance of syntactic structure," *Computer Networks*, vol. 56, no. 6, pp. 1814–1825, 2012.

[17] F. Ertam and E. Avcı, "A new approach for internet traffic classification: ga-wk-elm," *Measurement*, vol. 95, pp. 135–142, 2017.

[18] Y. Okada, S. Ata, N. Nakamura, Y. Nakahira, and I. Oka, "Application identification from encrypted traffic based on characteristic changes by encryption," in *Proceedings of the 2011 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, pp. 1–6, Naples, FL, USA, May 2011.

[19] P. Branch and J. But, "Rapid and generalized identification of packetized voice traffic flows," in *Proceedings of the 37th Annual IEEE Conference on Local Computer Networks*, pp. 85–92, Clearwater Beach, FL, USA, October 2012.

[20] Y. n. Dong, J. j. Zhao, and J. Jin, "Novel feature selection and classification of Internet video traffic based on a hierarchical scheme," *Computer Networks*, vol. 119, pp. 102–111, 2017.

[21] A. Raghuramu, P. H. Pathak, H. Zang, J. Han, C. Liu, and C.-N. Chuah, "Uncovering the footprints of malicious traffic in wireless/mobile networks," *Computer Communications*, vol. 95, pp. 95–107, 2016.

[22] W. Sun, Z. Wang, and G. Zhang, "A QoS-guaranteed intelligent routing mechanism in software-defined networks," *Computer Networks*, vol. 185, p. 107709, 2021.

[23] P. Tang, Y. Dong, J. Jin, and S. Mao, "Fine-grained classification of internet video traffic from QoS perspective using fractal spectrum," *IEEE Transactions on Multimedia*, vol. 22, no. 10, pp. 2579–2596, 2020.

[24] Y.-C. Cheng and P.-C. Wang, "Packet classification using dynamically generated decision trees," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 582–586, 2015.

[25] L. Tan, W. Su, W. Zhang et al., "In-band network telemetry: a survey," *Computer Networks*, vol. 186, p. 107763, Article ID 107763, 2021.

[26] X. Peng, L. Qiong, and L. Sen, "Internet traffic classification using support vector machine," *Journal of Computer Research and Development*, vol. 46, no. 3, pp. 407–414, 2009.

[27] C. Sammut and G. I. Webb, *Encyclopedia of Machine Learning*Springer, Berlin, Heidelberg, 2010.

[28] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapè, "MIRAGE: mobile-app traffic capture and ground-truth creation," in *Proceedings of the 2019 4th International Conference on Computing, Communications and Security (ICCCS)*, pp. 1–8, Rome, Italy, October 2019.

[29] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[30] B. Yamansavascilar, M. A. Güvensan, A. G. Yavuz, and M. E. Karsligil, "Application identification via network traffic classification," in *Proceedings of the 2017 International Conference on Computing, Networking and Communications (ICNC 2017)*, pp. 843–848, Silicon Valley, USA, January 2017.

[31] D. Bhat, J. Anderson, P. Ruth, M. Zink, and K. Keahey, "Application-based qoe support with p4 and openflow," in *Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 817–823, Paris, France, April 2019.

[32] S. Tomovic, N. Prasad, and I. Radusinovic, "SDN control framework for QoS provisioning," in *Proceedings of the 22nd Telecommunications Forum Telfor (TELFOR)*, pp. 111–114, Belgrade, Serbia, November 2014.

[33] P. Tang, Y. Dong, W. Tian, Z. Wang, and L. Yang, "Dynamic aggregation of flows according to qos class based on preference logic," *Chinese Journal of Computers*, vol. 43, no. 3, p. 21, 2020.

[34] W. Li, M. Canini, A. W. Moore, and R. Bolla, "Efficient application identification and the temporal and spatial stability of classification schema," *Computer Networks*, vol. 53, no. 6, pp. 790–809, 2009.

[35] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016)*, pp. 407–414, Rome, Italy, February 2016.

[36] Unb, "VPN-nonVPN dataset (ISCX VPN2016)," 2016, https://www.unb.ca/cic/datasets/vpn.html.

[37] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017)*, pp. 253–262, Porto, Portugal, February 2017.

[38] Unb, "Tor-nonTor dataset (ISCXTor2016)," 2016, https://www.unb.ca/cic/datasets/tor.html.

[39] D. Kumar and N. Priyanka, "Decision tree classifier: a detailed survey," *International Journal of Information and Decision Sciences*, vol. 12, no. 3, pp. 246–269, 2020.

[40] L. A. Breslow and D. W. Aha, "Simplifying decision trees: a survey," *The Knowledge Engineering Review*, vol. 12, no. 01, pp. 1–40, 1997.

[41] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Taylor & Francis, Wadsworth, 1984.

[42] Mininet, "Rapid prototyping for software defined networks," https://github.com/mininet/mininet.

[43] "P4 software switch - behavioral model version 2 (Bmv2)," https://github.com/%20p4lang/behavioral-model.

[44] S. Kaur, K. Kumar, and N. Aggarwal, "A review on P4-Programmable data planes: architecture, research efforts, and future directions," *Computer Communications*, vol. 170, pp. 109–129, 2021.

[45] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM 2008)*, pp. 63–74, New York, NY, October 2008.

[46] F. Klassen, "Tcpreplay - pcap editing and replaying utilities," http://tcpreplay.appneta.com/.