IEEE Open Journal of the Communications Society

Received 9 November 2023; revised 12 December 2023; accepted 1 January 2024. Date of publication 9 January 2024; date of current version 24 January 2024.

Digital Object Identifier 10.1109/OJCOMS.2023.3351706

Adaptive In-Network Traffic Classifier: Bridging the Gap for Improved QoS by Minimizing Misclassification

MUHAMMAD SAQIB [1] (Graduate Student Member, IEEE), HALIMA ELBIAZE [1] (Senior Member, IEEE), AND ROCH H. GLITHO [2]

[1] Department of Computer Science, Université du Québec à Montréal, Montreal, QC H3C 3P8, Canada

[2] Concordia Institute of Information Systems Engineering, Concordia University, Montreal, QC H3G 2W1, Canada

CORRESPONDING AUTHOR: M. SAQIB (e-mail: saqib.muhammad@courrier.uqam.ca)

This work was supported by CHIST-ERA Program under the "Smart Distribution of Computing in Dynamic Networks (SDCDN)" 2018 Call.

ABSTRACT In-network traffic classification presents an innovative approach to developing early-stage and accurate traffic classification solutions. However, despite its initial accuracy, the one-size-fits-all Machine Learning (ML) model becomes obsolete as traffic patterns evolve. This evolution in traffic patterns inevitably leads to misclassification, resulting in the erroneous mapping of traffic flows to Quality of Service (QoS) classes. Consequently, misclassification may lead to service quality violations, imposing penalties on Infrastructure Providers (InPs). The impact, however, is not solely tied to misclassification rates, as a multi-path network with paths of varying capacities can redirect traffic from low data rate classes to high data rate paths and vice versa, thereby influencing the overall outcome. Therefore, precisely quantifying the impact of misclassification on network performance, i.e., QoS, is paramount. This research aims to investigate and address the effects of traffic misclassification on Service Level Agreement (SLA) violations within multi-class, multi-path networks. To achieve this, we propose a novel framework to quantify SLA violations caused by misclassification, an economic model to assess its impact on provider profitability, and adaptive ML techniques to enhance traffic classification accuracy continually. The evaluation results reveal that the optimal path allocation for various traffic classes determines the targeted revenue. Meanwhile, the adaptivity of the classifier maintains prediction accuracy, ensuring the integrity of SLA through precise QoS class assignments. Hence, implementing an adaptive traffic classifier can mitigate penalties and sustain profitability. This work provides valuable insights for network operators, enabling effective misclassification management, resource optimization, and the maintenance of SLA integrity.

INDEX TERMS Programmable data plane, in-network traffic classification, traffic misclassification, QoS, network economics.

I. INTRODUCTION

WITH the proliferation of latency-sensitive services and applications within the Internet of Things (IoT) domain, the application of diverse Quality of Service (QoS) policies and efficient utilization of network resources becomes paramount. Network Slicing (NS) has emerged as a promising solution for resource orchestration, enabling QoS isolation through the overlay of multiple virtual networks on a shared network domain [1]. NS facilitates the efficient utilization and management of network resources while offering differentiated services at scale, allowing specific services to leverage dedicated network slices to meet their QoS requirements [2].

In network management, network traffic classification plays a pivotal role. It serves as the foundation for mapping incoming traffic flows to their appropriate QoS slices, thereby ensuring the provision of application-specific QoS guarantees [3]. This simplifies the enforcement of *Service Level Agreements (SLAs)*. Given the dynamic nature of traffic patterns and the growing diversity of IoT behaviors, early-stage traffic classification is indispensable for timely and accurate QoS provisioning. The programmability of

the data plane allows for the definition of customized matching criteria for traffic type identification [4], enabling fine-grain classification at a near-line rate. This concept is known as in-network traffic classification, which involves implementing rule-based Machine Learning (ML) models, such as Decision Trees (DTs), within switches to achieve high-speed processing [5], [6], [7].

In-network traffic classification turned out to be the key enabler in accurate and early-stage traffic classification solutions [8]. However, in the IoT domain, the traffic presently includes a variety of behaviours such as communication types, events, sources, patterns and volumes, etc. [9], [10]. These behaviours considerably impact traffic patterns, management, and control. Based on the number and type of active devices in the network, the devices' behaviour might generate a variety of characteristics, i.e., variability in data transmission period and payload size. Learning-based models have rapidly become a viable option for identifying the source devices and application types from Internet traffic [11]. Despite having a learning model with good accuracy, a single-time trained model becomes outdated as the traffic pattern changes over time [12]. This changing traffic pattern leads to *misclassification*, i.e., incorrect mapping of traffic flows to the QoS classes.

The researchers approached misclassification from a risk-free or cost-aware model training perspective, with minimal inaccuracy risk. For example, the authors in [13] argue that a solution for device identification (or classification) should priorly consider features extraction cost (computational and memory). Similarly, another work in [14] considers a cost-sensitive learning strategy to ensure the robustness of traffic classifiers against unbalanced datasets. They consider the misclassification cost during training and minimize the training model's cost. However, despite having a cost-effective model with good classification accuracy, the one-fit ML model loses its relevance over time as the traffic pattern changes. This loss of accuracy leads to incorrect mapping of traffic flows to the QoS classes, which further results in *SLA* violations and affects customer satisfaction in return. As a result, on the one hand, the service provider tries to increase revenue through priority-based traffic scheduling [15]. On the other hand, incorrect QoS class mapping by the classifier may lead to *SLA* violations, resulting in the addition of penalties that negatively impact the provider's profit. A significant challenge for Infrastructure Providers (InPs) is ensuring multi-priority traffic demands while maintaining maximum profit. To the best of the authors' knowledge, no existing works investigate the implications of traffic misclassification on network performance or network operators.

In our prior work [16], we examined the effect of traffic misclassification on InP profitability. We developed an economic model that directly calculates penalties based on class priority and misclassification rate. We assumed that *SLA* violations are directly proportional to the misclassification rates of traffic classes. However, this is only sometimes the case in multi-class, multi-path networks. In other words, the misclassification rate is not the sole metric for *SLA* violation, as misclassification does not necessarily result in *SLA* breaches. Instead, there is a need for precise measurement of the QoS status of network paths to calculate *SLA* violations. Hence, there remain unanswered questions. For instance, *what if traffic flows from a high data rate class are incorrectly directed to a low data rate path, or vice versa? What if traffic from multiple classes is poorly mapped to a single network path?* These open questions underscore the importance of measuring the misclassification impact on network performance and quantifying *SLA* violations to establish equitable penalties for the InP.

This research aims to investigate and mitigate the impact of traffic misclassification on *SLA* violations within multi-class, multi-path network environments. Our contributions encompass a novel framework for quantifying *SLA* violations caused by misclassification, an economic model to evaluate its impact on provider profitability, and adaptive ML techniques for continuous improvement of traffic classification accuracy. This work provides valuable guidance to network operators for effectively managing misclassification, optimizing resources, and ensuring the integrity of *SLAs*.

The proposed research investigates and mitigates the impact of traffic misclassification through a two-phase system design: the control plane and the data plane. In the control plane, we determine optimal routing paths to maximize revenue and employ adaptive ML techniques to generate updated classification rules for the data plane. The data plane features an in-network traffic classifier and performance monitoring for the quantification of *SLA* violations. By jointly monitoring classifier performance and network path utilization, we precisely measure the impact of misclassification on the QoS of network paths and enable adaptive model updates to improve accuracy, thereby reducing the negative impact of classifier.

The remainder of this paper is organized as follows. Section II provides an overview of the related work. Section III presents the problem definition and network model. Section IV introduces the proposed solution. The validation plan is discussed in Section V, followed by the presentation of experimental results in Section VI. Finally, Section VII contains the concluding remarks.

II. RELATED WORK

This section provides an overview of the most relevant approaches in the literature concerning the traffic misclassification problem, exploring how researchers have addressed this crucial issue from various perspectives.

In [18], the authors present two novel types of online Internet traffic classifiers designed to ensure performance guarantees for false alarm and false discovery rates. These classifiers aim to minimize overall misclassification rates while meeting specific constraints, with one classifier focused on reducing false alarm rates and the other on reducing false discovery rates. The proposed techniques

enhance network monitoring, service quality, and security measures.

Some researchers have approached misclassification as an optimal feature selection problem, primarily focusing on minimizing inaccuracy risks. For instance, in [19], the authors select the optimal set of features for IoT device fingerprinting on edge infrastructure to reduce the feature set's size while maintaining classification accuracy. This approach enables efficient device identification on resource-constrained edge nodes. Similarly, another study [13] addresses misclassification in an IoT environment by introducing a new variable called *risk* into the classification algorithm. They emphasize the importance of identifying whether misclassification occurred and determining the misclassified class, as this information can have significant implications for actions and costs. Incorporating the notion of risk aims to improve the accuracy and effectiveness of IoT device classification.

Another perspective on misclassification treats it as an imbalanced class problem. The authors tackle misclassification in [14] by proposing a cost-sensitive deep learning approach. They divide the dataset into partitions and create a cost matrix for each partition based on the data distribution. These costs are applied to the cost function layer to penalize classification errors, ensuring diverse costs for each type of misclassification. By incorporating these costs into the deep learning classifiers, the proposed approach aims to enhance the robustness of classifiers against the imbalanced class problem in network traffic classification.

Similarly, a different approach is presented in [20], where a deep learning model named the Cost Matrix Time-Space Neural Network (CMTSNN) is introduced. The CMTSNN model utilizes a cost penalty matrix and an improved cross-entropy loss function to enhance the classification accuracy of minority categories and overall multi-classification performance. The cost penalty matrix is applied to the cost penalty layer, and the improved cross-entropy loss function is used to calculate the loss, reducing the impact of data imbalance on model classification. This approach helps mitigate the effects of misclassification and improves the identification of encrypted abnormal traffic in the IoT network.

It is worth noting that the term *misclassification* is not new but has been explored in the state-of-the-art literature from risk-free or cost-aware perspectives. Researchers consider the misclassification risk or cost during the training process, focusing on minimizing the risk of inaccuracy or the cost of the model training. However, even with a risk-free or cost-effective model with good classification accuracy, a one-size-fits-all ML model can lose relevance over time as traffic patterns change due to *concept* or *data drift* [12]. This loss of accuracy leads to the incorrect mapping of traffic flows to QoS classes, resulting in *SLA* violations and reduced customer satisfaction. Considering the impact of post-classification processes becomes highly desirable in the context of multi-class, multi-path networks. Surprisingly, the reviewed work has not investigated the implications of incorrectly mapped traffic flows on network performance or InP.

III. PROBLEM FORMULATION AND NETWORK MODEL

A. TOWARD TRAFFIC CHARACTERIZATION

We begin by discussing the motivation for categorizing Internet traffic. Instead of relying on a well-known QoS classifier identifier (QCI) table, which usually offers a predetermined set of values linked to specific QoS characteristics, we aim to address the diverse and evolving nature of emerging applications and services. These entities often have distinct data rates, latency, and priority needs, and the rigid structure of QCI tables may need to be revised to accommodate such variability effectively. Our approach seeks a more flexible and adaptive method for classifying traffic flows to better align with the dynamic requirements of contemporary applications. For example, the chosen applications in TABLE 1 exhibit varying sensitivity to service quality requirements. We translated these service quality metrics, such as latency φ and data rate γ, to various classes having different priorities. Each class is characterized by specific traffic attributes, including high definition *HD*, real-time R_t or non-real-time traffic X_t - indicating the bandwidth and latency requirements; critical R_a or non-critical data rate X_a - to express the delay tolerance level [21].

In Fig. 1, we illustrate the given classes' latency and data rate requirements to establish their priorities. The requirements become more critical as the data rate increases while latency decreases. The higher the data rate and the lower the latency, the more critical these requirements become. Consequently, the priority of each class is determined by its specific service quality requirements. A service priority factor, denoted as α, is assigned to each class based on these requirements. The α value increases as the required data rate and latency become more stringent. Hence, this α is intricately linked to the criticality level, proportionally related to the necessary data rate while inversely associated with latency. The blue line visually emphasizes the expansion of the criticality level for each class, while the black line represents the α value. Given that α reflects the relative importance of the classes, it is utilized to define penalty and revenue generation functions in the proposed method. By delineating unique traffic classes and corresponding parameters for each application, our solution aims to optimize resource allocation based on the specific requirements of these diverse use cases.

B. PROBLEM DEFINITION

Let $c \in C$ be the set of priority classes, each with different service quality parameters, namely the latency φ and data rate γ. Here, the latency φ is defined in terms of end-to-end packet delay on the chosen route, and the data rate γ is given in bits per second (bps) as specified by the user capacity requirement. As a result, each class c is bounded

TABLE 1. Service Quality Requirements of a Few Emerging Applications [17]

Use case / Applications	Traffic Class	Data rate(γ)	E2E Latency(φ)	Priority (α)
Personalized BAN	Non-real-time/Critical rate (X_tR_a)	1-100 Mbps	$\leq$ 1000 ms	α_1
Internet of Everything	Non real-time/Non-critical rate (X_tX_a)	100-1000 Mbps	$\leq$ 100 ms	α_2
Smart Grid 2.0	Real-time/Critical rate (R_tR_a)	100-2000 Mbps	$\leq$ 10 ms	α_3
Live streaming	HD real-time/Highly critical rate (HR_tHR_a)	2000-4000Mbps	$\leq$ 1 ms	α_4

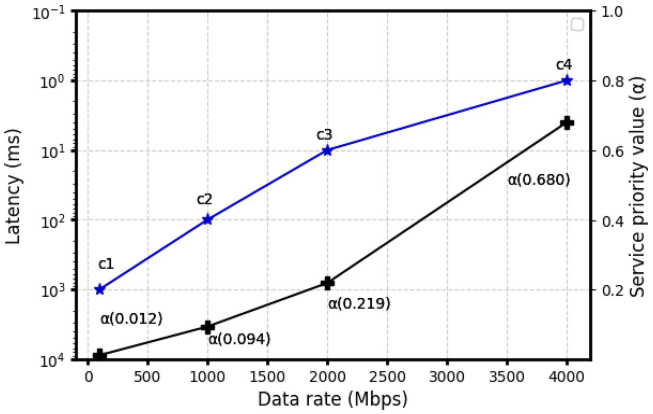

FIGURE 1. Service priority factor calculation.

TABLE 2. Notations

Notation	Description
N	Set of nodes
L	Set of links
$\mathcal{S}$	Set of network slices
K	Set of paths from source and destinations
C	Set of priority class (i.e., QoS group)
F	Set of traffic flows
T	Generated traffic load
$f_i^{(c)}$	Traffic flow i of class c
$\varphi_{p,i}^{(c)}$	Delay of packet p belonging to $f_i^{(c)}$
$\gamma_{l,i}^{(c)}$	Data rate of link l on flow i belonging to class c
b_l, b_k	Bandwidth of link l and path k
$x_k^{(s)}$	Traffic volume of slice s on path k
$\alpha_i^{(c)}$	Service priority of flow i belonging to class c
$\lambda_k^{(c)}$	Traffic portion from class c to path k
$\mu_k^{(c)}$	BW utilization of path k by class c traffic
δ	The selling price for a unit of bandwidth
β	Monetary penalty unit.

by threshold values of latency $\varphi_{max}^{(c)}$ and data rate $\gamma_{min}^{(c)}$. All the defined variables are summarized in TABLE 2.

Substrate network and constraints: We represent the underlying physical infrastructure as a directed graph $G = (N, L)$ with a set of nodes $n \in N$ and links $l \in L$, each link with a bandwidth $b_l > 0$ (measured in bps). On top of the substrate network, we consider the co-existence of multiple network slices indexed by $s \in \mathcal{S} = \{1, \ldots, S\}$. For clarity and simplicity, we focus on a scenario where each slice s serves network traffic from a single traffic class c with a single source to a single destination pair [22] having different service parameters in terms of φ and γ. Hence, we represent a slice by a source-destination (S/D) pair (u, v) where the following constraints must be respected for traffic flows between any pair:

$$\varphi_{p,i}^{(c)} \leq \varphi_{max}^{(c)}, \ \forall\, p \in f_i^{(c)} \in F^{(c)} \tag{1}$$

$$\gamma_{l,i}^{(c)} \geq \gamma_{min}^{(c)}, \ \forall\, l \in L, \ \forall f_i^{(c)} \in F^{(c)} \tag{2}$$

The above constraint (1) ensures an end-to-end delay threshold along the multi-hop route. The associated end-to-end delay φ with a packet p across the multi-hop path between source node u and destination node v, in particular, shall not exceed the maximum latency limit for a given class c, i.e., $\varphi_{max}^{(c)}$. Furthermore, constraint (2) assures that the data rate γ at any link $l \in L$ should be sufficient to meet the capacity demand of passing flow i belongs to class c.

In addition, the paths between (u, v) pairs over links L are indexed by $k \in K = \{1., K\}$. We denote the traffic volume each slice s generates as $x_k^{(s)}$ going through path k. Since multiple slices share the same physical network, the bandwidth consumption at each path k may be, at most, the available bandwidth b_k. Thus, we also define the paths' bandwidth constraint:

$$\sum_{s \in S} \sum_{k \in K} x_k^{(s)} \leq b_k \tag{3}$$

Provider's revenue: The service provider generates revenue by optimal resource (i.e., bandwidth) allocation to heterogeneous traffic demands. The revenue mainly depends on the resources requested by the service. That is the product of the selling price of a bandwidth unit and a class's service priority factor. The pricing policy determines the charge per unit bandwidth for each substrate link $l \in L$. A differential pricing policy is considered based on the class's criticality level. Thus, the revenue gained by the provider at the time t by selling the bandwidth resource can be expressed as:

$$\sigma(t) = \sum_{c \in C} \sum_{f \in F^{(c)}} \sum_{s \in S} \sum_{k \in K} \delta * \alpha_f^{(c)} * x_k^{(s)}(t) \tag{4}$$

The selling price for a bandwidth unit is δ. The priority of flow f of class c (i.e., service quality priority) is represented as $\alpha_f^{(c)}$, and $x_k^{(s)}(t)$ stands for satisfying bandwidth over a path k between (u, v) pairs of slices $s \in S$.

Network traffic classifier: The InP increases revenue by allocating resources to various traffic classes. Meanwhile, a traffic classifier at the network's edge assigns the incoming traffic flows to the correct traffic class, i.e., QoS slice. In the case of incorrect traffic class mapping, the InP could not implement the appropriate QoS policies, which would impact the Customer Satisfaction Level (CSL). In order

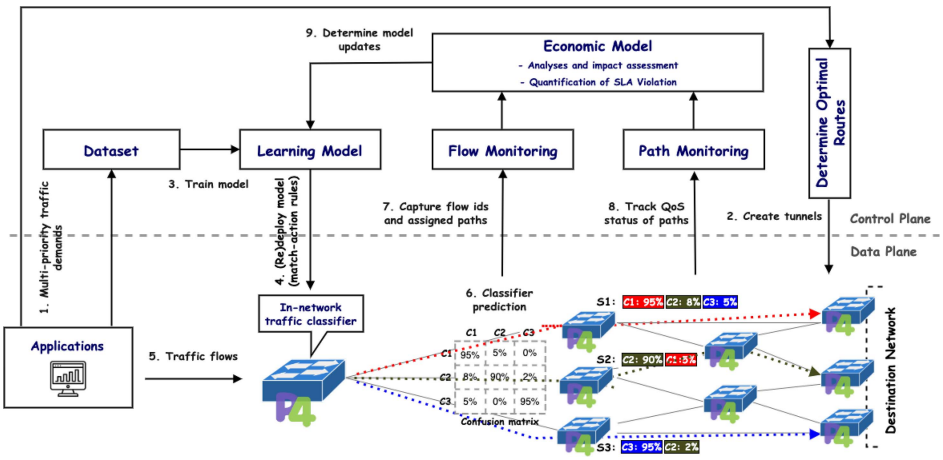

FIGURE 2. System design.

to maintain good classification accuracy, it is crucial to regularly check the classifier's prediction and penalize the classifier for inaccurate mappings.

Let $\mathcal{C}$ be the classifier that classifies the incoming traffic flows f_i to the corresponding class. Any misclassification, i.e., incorrect QoS slice mapping, might result in an *SLA* violation, adding a penalty that can be calculated as the per-class criticality level.

Misclassification rate: We use *f1-score* as a performance metric to measure the per-class misclassification rate. *f1-score* assesses the classification model's performance starting from the confusion matrix; it aggregates *Precision* and *Recall* measures under the concept of harmonic mean [23]. The formula of *f1-score* can be interpreted as a weighted average between *Precision* and *Recall*:

$$f1 = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \tag{5}$$

where

$$Precision = \frac{TP}{(TP + FP)} \tag{6}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{7}$$

TP means the observation is positive, and the sample is predicted to be positive. *FN* is that the observation is positive, but the sample is predicted to be negative. *TN* describes that observation is negative, and the sample is predicted to be negative. And *FP* represents that the observation is negative, but the sample is predicted to be positive.

The *f1-score* reaches its best value at one and the worst score at 0. Hence, the misclassification rate of a particular class c can be defined as:

$$\lambda^{(c)} = 1 - f1^{(c)}, \quad \forall\, c \in C \tag{8}$$

Penalty: The provider must return the penalty incurred due to the misclassification. The penalty ρ for a particular class c over time t can be calculated as a product of the monetary penalty unit, the class's priority, the misclassification rate, and the path utilization rate of the class. The total penalty ρ associated with each class c at the time t can be computed as follows:

$$\rho(t) = \sum_{t \in T} \sum_{c \in C} \beta * \alpha^{(c)} * \lambda_k^{(c)} * \mu_k \tag{9}$$

where β is the monetary penalty unit, $\alpha^{(c)}$ is the class priority, $\lambda^{(c)}$ is the misclassification rate of class c and μ_k is the bandwidth utilization rate of path k.

Objective function: The objectives are to maximize the provider's profit and minimize *SLA* violations. Maximizing the provider's profit can be achieved by maximizing σ and minimizing ρ. The objective function $\mathcal{P}$ can therefore be written as follows:

$$\max_{\mathcal{P}} \sum_{t \in T} (\sigma(t) - \rho(t))$$
$$\text{Subject to constraint } \textbf{(1)}, \textbf{(2)} \text{ and } \textbf{(3)}. \tag{10}$$

IV. PROPOSED SOLUTION

In this section, we introduce a framework designed to investigate the impact of traffic misclassification and an adaptive learning-based approach to mitigate this impact. Fig. 2 shows the high-level system design, encompassing two primary phases: (i) optimizing routing paths and offline ML model training within the control plane and (ii) identifying traffic classes for network slice allocation and performance monitoring within the data plane. In the following subsections, we explore the details of our integrated and smart design for addressing the traffic misclassification problem.

A. CONTROL PLANE

The control plane comprises three essential logical components: route optimization, ML model training, and an economic model for quantifying *SLA* violations and guiding updates to the ML model.

1) DETERMINE OPTIMAL ROUTES

The control component first utilizes a routing module to identify the most efficient paths for accommodating heterogeneous traffic demands. Subsequently, it sets the targeted revenue, determined by the revenue generation function, as defined in Eq. (4). The function considers traffic demands and available routes with bandwidth capacity as inputs, resulting in the calculation of the maximum achievable revenue. To solve the allocation problem for traffic demands across multiple classes and paths in the network, we formulate this optimization problem as an Integer Linear Program (ILP). Our ILP can be generalized as a 0/1 Multi-Knapsack Problem (MKP) [24]. The complete formulation is detailed below:

$$\max_{\sigma} \sum_{c \in C} \sum_{k \in K} \delta \cdot \alpha^{(c)} \cdot x_k^{(c)} \tag{11}$$

subject to

$$\sum_{k \in K} x_k^{(c)} \leq 1 \quad \forall c \tag{12}$$

$$\sum_{c \in C} \gamma^{(c)} \cdot x_k^{(c)} \leq b_k \quad \forall k \tag{13}$$

$$\sum_{c \in C} \varphi_k \cdot x_k^{(c)} \leq \varphi_{max}^{(c)} \quad \forall k \tag{14}$$

$$x_k^{(c)} \in \{0, 1\} \quad \forall c, k \tag{15}$$

where

$$1x_k^{(c)} = \begin{cases} 1 \text{ if class } c \text{ allocated to path } k \\ 0 \text{ otherwise} \end{cases} \tag{16}$$

Equation (13) guarantees that traffic demand from a specific class c is accommodated by being assigned to at most one path k. Constraint (14) ensures that the allocation adheres to the capacity of each designated path, while constraint (15) ensures that the cumulative latency on the allocated path does not surpass the predetermined limit for each traffic class. Finally, the binary decision variables represent the pivotal choices in allocating traffic demands to the network paths. We utilized the Gurobi optimizer to solve the ILP (see Section V-B).

2) LEARNING MODULE

Additionally, the control plane employs a learning module to gain insights into multi-priority traffic patterns from a given dataset. The dataset, denoted as S, comprises packets p_j from sub-flows $(f_i(1:j))$ and is divided into two subsets: a training set, S_T, and a testing set, S_P. The learning module initially trains the ML model on S_T. Subsequently, it translates the resulting output, represented as *if-else* conditions, into *Match-Action Tables (MATs)* within a P4-enabled programmable switch [25] for real-time inference. This translation process is facilitated through a control plane API called P4Runtime [26].

Numerous supervised learning approaches are available in the literature for traffic characterization, but not all are suitable for implementation in P4 [6]. Our objective is to seamlessly integrate the ML model's output into the data plane, necessitating compatibility with the available operations in P4. Consequently, we opted for a classical decision tree algorithm to circumvent the limitations of P4. Given the current primitives in the P4 language [27], a *Decision Tree Classifier (DTC)* proves to be the more suitable choice for this task. It requires a comparison operation to classify an element x, which can be readily expressed in P4 using *match-action* rules.

B. DATA PLANE

The data plane comprises an in-network traffic classifier and performance monitoring logic responsible for detecting *SLA* violations.

1) IN-NETWORK TRAFFIC CLASSIFIER

After embedding the learning-based model's output into the data plane, the next step is predicting the incoming packet class. Our in-network traffic classifier is designed using a novel and effective method capable of quickly and accurately classifying diverse traffic classes. What makes our approach unique is its reliance on a single stable feature, specifically the sequential packet size information that can be directly extracted from the packet's header.

The switch maintains registers to track flow IDs, packet sizes, and packet counters for the initial packets of each flow. For incoming packets, the switch's parser extracts the flow ID and relevant features, such as the five-tuples (i.e., IPs, Ports and protocol) and packet size, from the header and stores these feature values in the pipeline's metadata. The flow ID register records all classified flows, allowing the switch to handle packets belonging to these classes efficiently. This means that packets from the classified flows do not need to undergo the decision tree process and are processed at a line rate, according to the identified class or QoS policy.

When a flow is not classified, the switch checks the packet counter for that flow. The parser extracts packet sizes and stores them in a size vector until the packet counter reaches a certain threshold. Once the packet counter reaches the threshold, the *MATs* are used for classification based on the size vector. Subsequently, the switch directs subsequent flow packets to the appropriate slice (i.e., QoS group). We refer to [8] for more details about the traffic flow classification process inside a programmable data plane.

2) PERFORMANCE MONITORING

The next step involves calculating the misclassification rate for each class based on the classifier's predictions. We continuously monitor the classifier's performance by computing the *f1-score* from the data plane's predictions. Due to evolving traffic patterns, the deployed model's accuracy diminishes over time, resulting in flows being incorrectly mapped to the wrong QoS classes.

In our design, traffic flows from a set of classes, denoted as C, are mapped to a corresponding set of slices, denoted as S. As illustrated in Fig. 2, the colored arrows, such as *red, green,* and *blue,* represent distinct slices with varying bandwidth capacities and propagation delays. For instance, misclassification causes approximately 8% of flows from class $c2$ and 5% of flows from class $c3$ to be routed through slice $s1$. Similarly, about 5% of flows from class $c1$ and 2% of flows from class $c2$ are routed through $s2$ and $s3$, respectively.

Due to these misclassifications, the intended QoS policies may not be applied to a fraction of the flows, resulting in degrading service quality. The switch captures the flow IDs and associated routing paths to address this issue. Based on the generated traffic load $T^{(c)}$ for each class and the classifier's predictions, we calculate the traffic load on each network path k and determine the bandwidth utilization rate u_k for each path.

Since the available bandwidth capacity varies across network paths, some misclassified traffic flows from a particular class may be directed to paths with either higher or lower capacity. This can lead to underutilization or overutilization of certain paths, affecting service quality. Measuring the precise impact on service quality, such as how each traffic class receives its allocated resources, is a complex task in a multi-class and multi-path network. Therefore, we employ a joint approach, periodically monitoring the classifier's predictions and the QoS status of the paths to assess the impact of traffic misclassification on service quality.

C. INTEGRATED AND ADAPTIVE DESIGN

The entire process for assessing and mitigating the impact of traffic misclassification is outlined in Algorithms 1-3. Algorithm 1 iterates T times, invoking path scheduling and penalty calculation algorithms at lines 7 and 8, respectively. Algorithm 2 aims to maximize revenue through optimal path allocation, while Algorithm 3 calculates the misclassification rate for each traffic class, the utilization rate of each network path, and the associated penalties. The InP profit is periodically determined by deducting the estimated penalties from the generated revenue at line 9.

Algorithm 2 utilizes a solver to iterate over traffic classes and network paths. For each item (i.e., traffic demand from a class c), we have K choices (paths) for allocation. Therefore, we must allocate C items to K choices over each period t. For each feasible allocation, where a network path can adequately meet the service quality requirements of a traffic class, the revenue is calculated by multiplying the class's priority factor with the price for a bandwidth unit and the satisfiable bandwidth. The generated revenue is temporarily assigned to a variable σ at line 5. The solver aims to maximize the revenue through optimal allocation and returns the maximum generated revenue as a result of the optimal assignment of C to K. The algorithm's computational complexity is inherently tied to the performance characteristics and algorithms implemented within the solver. However, in the worst case, the algorithm may need to explore all possible combinations of C and K. Hence, the worst-case complexity can be approximated as $\mathcal{O}(K^C)$.

Algorithm 1: Maximizing InP's Profit

Input: *C, F, T, K*
1 *C: A list of classes with varying QoS requirements*
2 *$F^{(c)}$: Network traffic flows of class c*
3 *$T^{(c)}$: Generated traffic load for each class c*
4 *K: A list of capacity-varying routes*
Output: *$\mathcal{P}$: Profit*
5 $\mathcal{P} = 0$;
6 **for** $t \in T$ **do**
7 $\quad \sigma(t) =$ `PathScheduling(C, K)` ; `// Algorithm 2`
8 $\quad \rho(t) =$ `PenaltyCalculation(C,F,T,K)` ; `// Algorithm 3`
9 $\quad \mathcal{P}(t) = \sigma(t) - \rho(t)$; `// Profit over time t`

Algorithm 2: PathScheduling

Input: *C, K*
Output: *max_revenue*
1 *max_revenue* $= 0$;
2 **for** $c \in C$ **do**
3 $\quad$ **for** $k \in K$ **do**
4 $\quad\quad$ **for** *each feasible solution* **do**
5 $\quad\quad\quad \sigma = \sum_{c \in C} \sum_{k \in K} \delta \cdot \alpha^{(c)} \cdot x_k^{(c)}$;
6 $\quad\quad\quad$ **if** $\sigma >$ *max_revenue* **then**
7 $\quad\quad\quad\quad$ *max_revenue* $= \sigma$;
8 **return** *max_revenue*

Algorithm 3 is designed to monitor network performance, specifically path utilization rates, and precisely calculate the penalties for InP as a result of *SLA* violations. The first nested loop from lines 1 to 8 runs for $K \times C$, calculating the network path utilization rates. Initially, the classifier's predictions, denoted as $\lambda_k^{(c)}$, are derived from classification results where columns represent paths, and rows represent classes. The network path utilization rate, i.e., μ_k at line 4, is then determined by multiplying the generated traffic load $T^{(c)}$ for a specific class c by the classifier's prediction $\lambda_k^{(c)}$ and dividing it by the network path's capacity b_k. This utilization rate reflects how efficiently each path is being used. Furthermore, to ensure fairness and uniformity in the penalty calculation, we scale the utilization rate for each path by dividing the over-utilization by 100 at line 7, allowing for a consistent assessment.

The second nested loop from lines 9 to 14 runs for $C \times K$, calculating penalties for each traffic class based on traffic misclassification and path utilization rates. The sub-loop iterates over network paths, assessing penalties for traffic incorrectly mapped to paths other than its intended route. Line 11 indicates that if the traffic is directed to a path other than its planned route, then calculate the penalty for that portion of traffic on the assigned path. The penalty is computed as the product of a penalty unit, class priority, path utilization rate, and misclassification rate at line 12. At

Algorithm 3: PenaltyCalculation

Input: C, F, K, T
Output: ρ
// Calculate path utilization
for $k \in K$ **do**
  for $c \in C$ **do**
    $\lambda_k^{(c)} = pr(F_k^{(c)})$;
    $\mu_k += \left(\frac{T^{(c)} \cdot \lambda_k^{(c)}}{b_k}\right) \cdot 100$;
    $c = c + 1$;
  if $\mu_k > 100$ **then**
    $\mu_k = \frac{\mu_k}{100}$;
  $k = k + 1$;
// Calculate penalty
for $c \in C$ **do**
  for $k \in K$ **do**
    if $c \neq k$ **then**
      $\rho^{(c)} += \beta \cdot \alpha^{(c)} \cdot \mu_k \cdot \lambda_k^{(c)}$;
    $k = k + 1$;
  $c = c + 1$;
$\rho = \sum_{c \in C} \rho^{(c)}$;
return ρ

the end of the iterations, the algorithm calculates and returns the accumulated penalty for each time (i.e., days).

The combined performance monitoring of the classifier and network paths enables us to measure and quantify the misclassification effects accurately. Based on this quantification, the economic model determines the appropriate updates to adapt to newly obtained traffic patterns and improve the classifier's performance. Consequently, we incrementally regularly incorporate freshly acquired data into the existing model to keep an up-to-date ML model within the data plane. This approach ensures that flows are predicted correctly and misclassification rates are minimized. The complete computational complexity of the proposed algorithms can be approximated as:

$$\mathcal{O}\Big(a_(T) \times \Big(a_2\Big(K^C\Big) + a_3(K \times C + C \times K)\Big)\Big)$$

- $a_1(T)$: Represents the computational complexity of running the solver for periods.
- $a_2(K^C)$: Signifies the solver's complexity, linked to its internal logic. In the worst case, the algorithm explores all possible combinations, resulting in an approximate worst-case complexity of $\mathcal{O}(K^C)$.
- $a_3(K \times C + C \times K)$: Represents the complexity of calculating the path utilization rates and penalty for each class on the associated path.

The overall worst-case complexity depends on the number of traffic classes C and network paths K.

V. VALIDATION METHODOLOGY

This section presents the dataset employed, describes the experimental setup, and outlines the distinct performance analysis cases covered in Sections V-A–V-C, respectively.

TABLE 3. Dataset Summary

Class	Device type	# of flows	# of packets
C1(XR_a)	Sensor	6411	233329
C2(XX)	Appliance	5439	23561
C3(R_tR_a)	Controller	16788	270840
C4(HR_tHR_a)	Camera	1601	144187

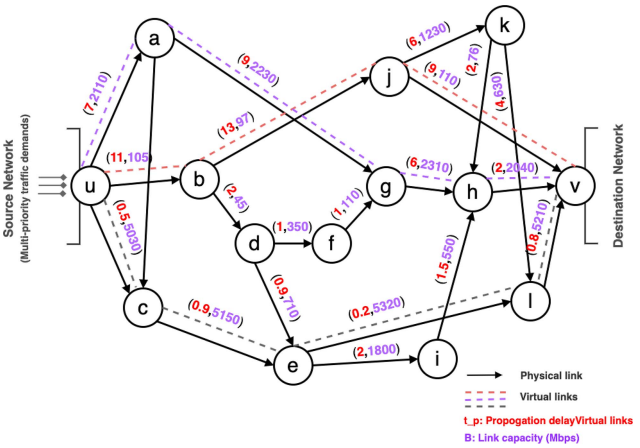

FIGURE 3. Network topology.

A. DATASET DESCRIPTION

We have utilized packet capture (PCAP) traces of IoT devices from [9] as our dataset. Among the available instances in the dataset, we consider the PCAP files spanning seven days, from September 23 to September 29, 2016. These files contain flows associated with four distinct applications and involve various IoT devices.

These IoT devices have been categorized into different classes, each with varying degrees of priority. To ensure diversity, we selected devices capable of being assigned to various QoS groups, ranging from high bandwidth and low latency to best effort. Devices within the same class exhibit similar traffic characteristics. Consequently, for validation, we select a single device from each class. An overview of the dataset about these chosen devices is shown in TABLE 3.

B. EXPERIMENTAL SETUP

We organized our experiments into three steps. First, we define the network topology for finding optimal paths using a solver in Section V-B.1. Then, we discuss the model training and deployment in Section V-B.2. Finally, we present the simulation setup for the in-network traffic classifier and performance monitoring in Section V-B.2.

1) NETWORK TOPOLOGY

Since we consider the performance requirements for various classes of traffic defined in TABLE 1, the InP generates revenue by satisfying traffic demands with varying service quality parameters. We define a network topology with diverse paths (see Fig. 3) and employ Gurobi solver to identify the optimal paths for fulfilling the traffic demands.

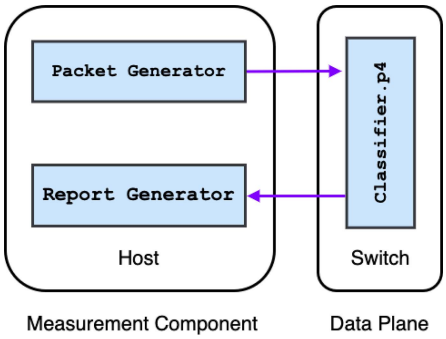

FIGURE 4. Simulation setup.

2) MODEL TRAINING AND DEPLOYMENT

We relied on Python's *scikit-learn*[1] implementation for model training, and we used cross-validation to choose the best hyperparameters (e.g., tree height and the minimum number of items per leaf) to avoid overfitting. The *DTC* classifier produces output translated into *MATs* in the form of *match-action* rules using P4Runtime API. A single table handles a single packet's size range from the first few packets' sequences of an individual flow. The number of tables equals the input length plus a class prediction table. Before applying *MATs*, the P4 code extracts the payload sizes of the sequential packets from an individual flow and stores them as a size vector in *SRAM*. Once the input length's threshold is reached, the switch traverses the size vector through the *MATs* to predict the traffic class.

3) SIMULATING DATA PLANE

The logical components of the simulation setup are shown in Fig. 4. The Measurement Component (on the left) is responsible for generating, collecting, and analyzing the network traffic, whilst the Data Plane Component (on the right) is the target to be evaluated. The data plane is implemented in P4, compiled with a target of behavioural model version 2 (BMv2) [25]. The structure of the simulation setup is organized as follows:

- *Host* send the packets with Pktgen-DPDK[2] and also receives the packets back already classified and timestamped. This component reports the classifier's predictions and network performance for the packets.
- *Switch* contains the traffic classification algorithm responsible for classifying and mapping incoming traffic to the appropriate QoS slices.

C. CASES FOR PERFORMANCE ANALYSES

Traffic misclassification doesn't solely equate to *SLA* violations. In other words, misclassification doesn't necessarily lead to *SLA* breaches. For instance, traffic from a low-priority and low-data-rate class can be erroneously mapped to a high-priority and high-data-rate path without impacting that class significantly. Conversely, misclassifying traffic from a high-priority, high-rate class to a low-priority, low-data-rate path can severely affect *SLA* compliance. Therefore, *SLA* violations for a specific class may or may not be linked to misclassification.

In a multi-class, multi-path network, numerous possibilities exist, necessitating precise measurement of *SLA* violations for equitable calculation of per-class penalties. We've defined three validation and performance analysis cases to simplify our approach, facilitating a foundational understanding of *SLA* violation quantification.

- *Case 1:* When traffic from a low-priority class is incorrectly mapped to multiple high-priority paths;
- *Case 2:* When traffic from a high-priority class is incorrectly mapped to multiple low-priority paths;
- *Case 3:* When traffic from multi-priority classes is incorrectly mapped to a single path.

VI. EXPERIMENTAL RESULTS

The experimental results are divided into three main parts: firstly, determining optimal paths to achieve the targeted revenue by satisfying diverse traffic demands; secondly, assessing the impact of traffic misclassification; and thirdly, evaluating the adaptivity of the machine ML to mitigate the effects of misclassification. These results are presented in Sections VI-A–VI-C.

A. SATISFYING MULTI-PRIORITY TRAFFIC DEMANDS

The InP generates revenue by meeting traffic demands while considering distinct service quality parameters. The selling price for a single unit of bandwidth is set at 5$. The provider allocates the optimal routing paths the solver determines to meet diverse traffic demands. Based on these satisfiable traffic demands, the provider calculates the targeted revenue using the revenue generation function defined in (Eq. (4)). The generated revenue, as depicted in Fig. 6, is shown for the specified classes from Table 1. The bar graph illustrates the correlation between bandwidth demand and revenue. This correlation is particularly robust for classes with a high criticality level, such as *class-4*, while *Class-2* exhibits the opposite behavior due to its lower criticality level. Consequently, the InP identifies the optimal route for each traffic class to maximize revenue while meeting the demands.

B. IMPACT OF TRAFFIC MISCLASSIFICATION

The next step involves assessing the impact of misclassification on the provider's profit. Since network traffic patterns evolve rather than remain constant, an ML model's accuracy naturally degrades over time as traffic patterns change. We employ three distinct cases to measure the impact of misclassification and quantify *SLA* violations (see Section V-C).

We divided the chosen dataset into multiple chunks for analysis, each representing daily streaming data. The initial data chunk was dedicated to training, while subsequent

1. https://scikit-learn.org/
2. https://pktgen-dpdk.readthedocs.io/en/latest/

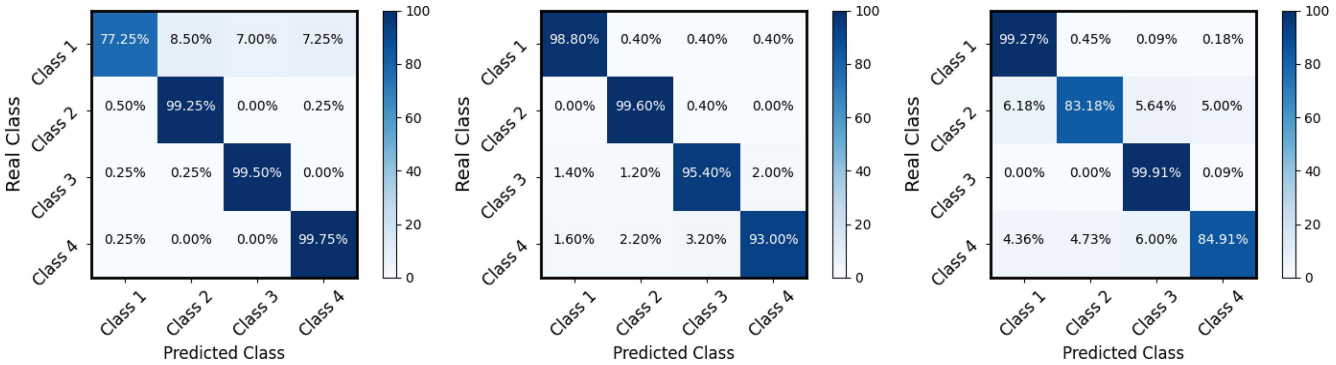

FIGURE 5. Confusion matrix for case 1 (left), 2 (middle), and 3 (right).

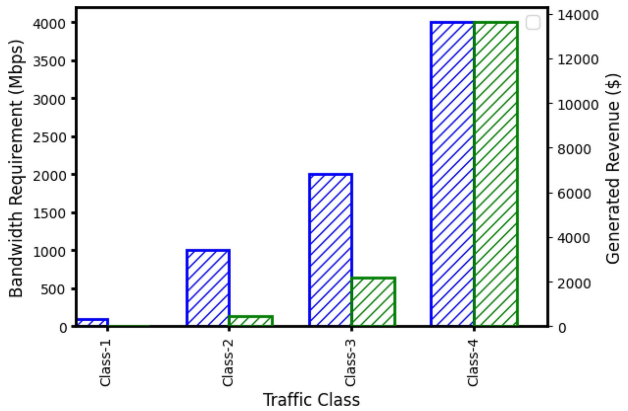

FIGURE 6. Per-class revenue determined by optimal path allocation.

fragments were used for testing. We examined new data patterns on the *2nd, 3rd, and 4th day* to identify the ML model drift, validating the cases mentioned earlier. We then periodically monitored the model's performance on contemporary data patterns.

1) CLASSIFICATION RESULTS

The following day's confusion matrix for each case can be seen in Fig. 5. In the first case, a substantial portion of traffic from a low-priority class (i.e., class 1) is erroneously mapped to high-priority classes. In contrast, the second case involves a minor portion of traffic from a higher-priority class being inaccurately assigned to low-priority classes.

The third case illustrates a scenario where there is no misclassification within specific classes (i.e., classes 1 and 3), but traffic from other classes is directed towards their paths due to misclassification. Fig. 7 draws the *f1-score* for the complete simulation encompassing all these cases. Given that incoming traffic patterns are unknown to the existing model, the *f1-score* of the classifier's predictions diminishes in the subsequent days. This underscores the necessity for model adaptivity to address the evolving traffic patterns and tackle the misclassification problem.

2) IMPACT ON NETWORK PERFORMANCE

Fig. 8 illustrates the network path capacities for each traffic class. Dotted lines without markers represent the path's capacity (i.e., available bandwidth), while dotted lines with markers indicate the network traffic load on each path. Fig. 9 presents the path utilization rates from traffic mapping to the predicted network paths.

In the first case, despite a higher misclassification rate from class 1, the traffic load on all network paths remains below their capacities. Consequently, the influence of misclassification in case 1 on network performance is almost negligible, as depicted by the network traffic load on each corresponding path in Fig. 9. This is attributed to the incorrect mapping of traffic from low to high data rate classes, which imposes minimal overhead on high-capacity network paths and can be accommodated by the network.

Conversely, a minor portion of traffic flows (approximately 1.6%) from a high-priority class is mistakenly mapped to the low-priority class in case 2, resulting in a significant increase in the utilization rate of the corresponding path (i.e., path 1).

The final case visualizes the impact of traffic misclassification from another perspective. In this scenario, there may be no misclassification within specific classes (e.g., class 1 or 3). Still, traffic from other classes may be erroneously mapped due to misclassification from other classes (e.g., 2 and 4). This can lead to over-utilization of the associated paths.

The three presented cases demonstrate the impact of traffic misclassification on network performance, which results from the incorrect mapping of traffic to designated network paths. It's important to note that this impact isn't solely determined by the rate of traffic misclassification but also hinges on the specific QoS requirements of the traffic classes and the QoS status of the associated network paths. Therefore, in addition to considering factors such as the penalty unit, class priority, and misclassification rate [16], the final penalty calculation for traffic classes also incorporates network path utilization.

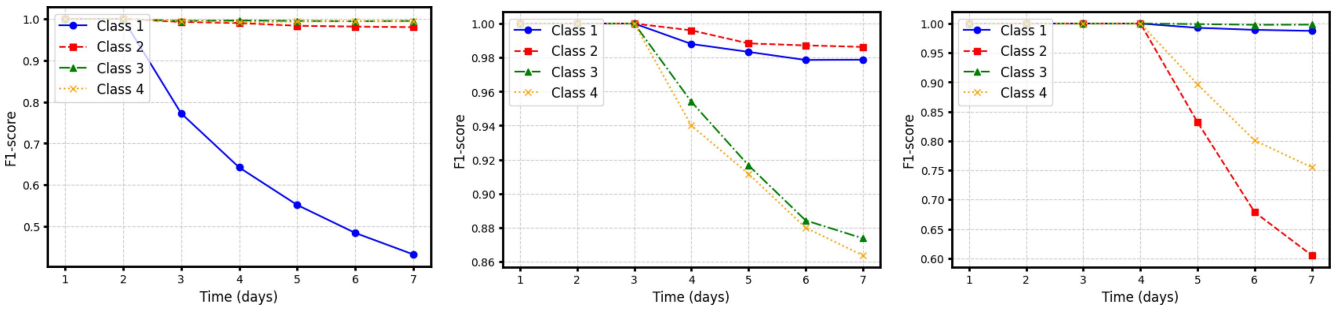

FIGURE 7. Classification result (i.e., f1-score) from complete simulation for cases 1 (left), 2 (middle), and 3 (right).

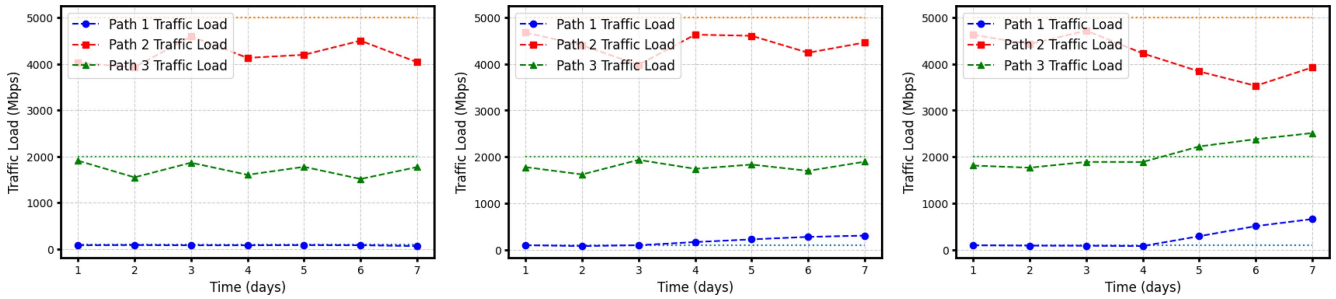

FIGURE 8. Network path capacities and traffic load for cases 1 (left), 2 (middle), and 3 (right).

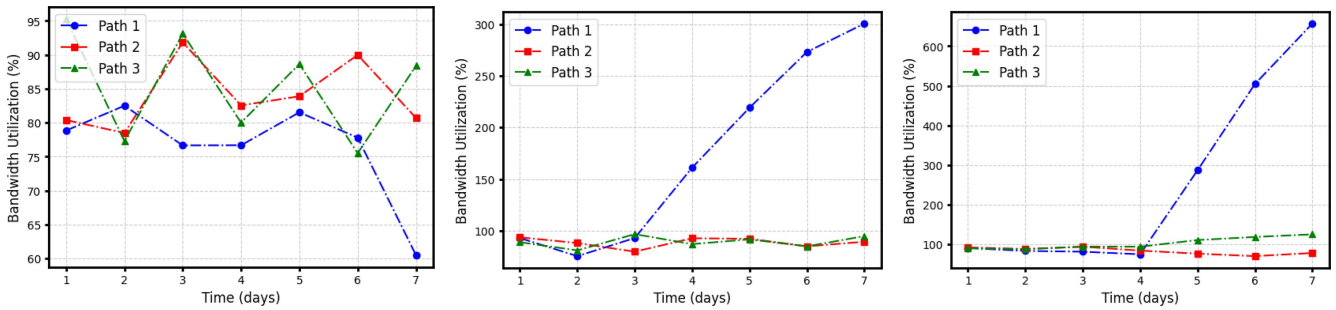

FIGURE 9. Network path bandwidth utilization rates for cases 1 (left), 2 (middle), and 3 (right).

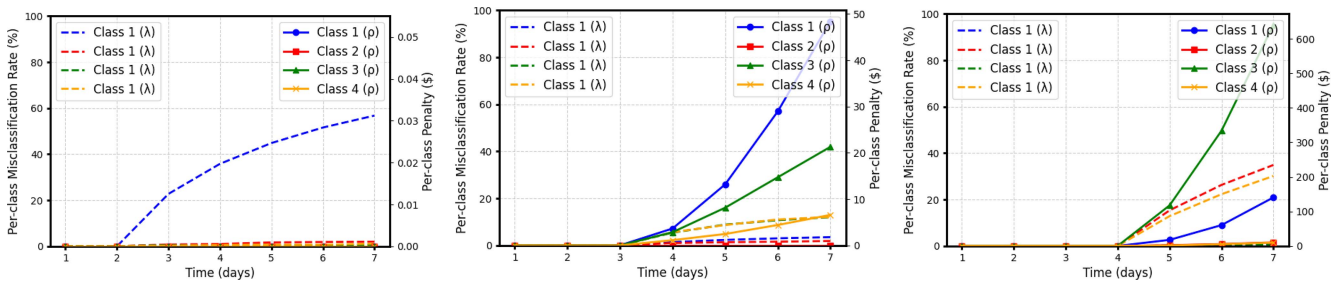

FIGURE 10. Improved penalty calculation for cases 1 (left), 2 (middle), and 3 (right).

3) IMPACT ON PENALTY

The impact of traffic misclassification on penalties is illustrated in Fig. 10. In the initial scenario, despite a heightened rate of traffic misclassification, no supplementary penalty is imposed, as the network path utilization rate stays below 100%. Conversely, a marginal portion of erroneously mapped traffic from class 4 leads to the imposition of a penalty, even for class 1, owing to QoS degradation on the path of class 1 caused by traffic congestion. Similarly, class 3 traffic was correctly mapped to its designated path, but the associated path became congested due to misclassification from other classes, leading to a higher penalty for class 3.

Hence, the impact of traffic misclassification on the final penalty is not solely determined by the misclassification rate

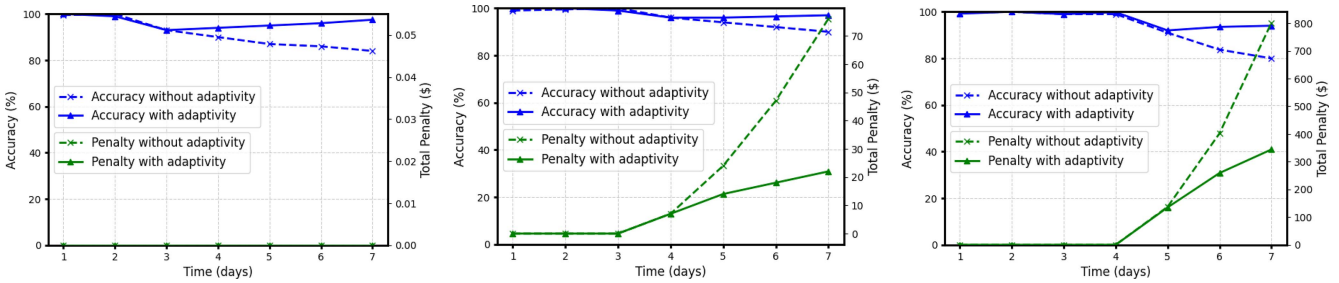

FIGURE 11. Adaptive classifier results for cases 1 (left), 2 (middle), and 3 (right).

or class priority but also depends on the QoS status of the network paths.

4) RESULTS FROM ADAPTIVE CLASSIFIER

The final set of results offers valuable insights into the impact of adapting to evolving traffic patterns and how this adaptation diminishes the influence of misclassification on network performance. We periodically incorporated newly received traffic patterns into the existing ML model to achieve this. The adaptation process was designed to alleviate the impact of misclassification on the network's QoS.

The results, as illustrated in Fig. 11, clearly demonstrate that enhancing the accuracy of our classifier has a direct and positive effect on the InP profit. Specifically, we observe a reduction in the total penalty as accuracy improves. In simpler terms, as we enhance the classifier's accuracy, it becomes more proficient at precisely assigning incoming traffic to the appropriate QoS classes. This, in turn, leads to diminished penalties while sustaining higher profits.

In summary, adapting to changing traffic patterns empowers the proposed approach to minimize the adverse effects of misclassification, ultimately maximizing the InP's profit and overall service quality.

VII. CONCLUSION

This paper presents an approach that investigates the network performance and economic implications of traffic misclassification, introducing an adaptive classification method to address this challenge effectively. Our validation process involved characterizing multi-priority traffic, optimizing path allocation, and mapping incoming flows to various QoS classes. We found that in a multi-class, multi-path network, the impact of misclassification on penalty charges varies due to several factors, including the criticality level of each class, the misclassification rate, and the associated paths' utilization rate. Furthermore, we observed that the classifier's adaptivity significantly improves prediction accuracy, ensuring precise QoS class assignments and thus reducing penalties while increasing profits. This research contributes to a better understanding of the complex nature of traffic misclassification and its impact on *SLA* violations within intricate network environments.

REFERENCES

[1] Y. Wu, H.-N. Dai, H. Wang, Z. Xiong, and S. Guo, "A survey of intelligent network slicing management for industrial IoT: Integrated approaches for smart transportation, smart energy, and smart factory," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 2, pp. 1175–1211, 2nd Quart., 2022.

[2] J. J. A. Esteves, A. Boubendir, F. Guillemin, and P. Sens, "Optimized network slicing proof-of-concept with interactive gaming use case," in *Proc. 23rd Conf. Innov. Clouds, Internet Netw. Workshops (ICIN)*, 2020, pp. 150–152.

[3] H. Yao, P. Gao, J. Wang, P. Zhang, C. Jiang, and Z. Han, "Capsule network assisted IoT traffic classification mechanism for smart cities," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7515–7525, Oct. 2019.

[4] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends," *IEEE Access*, vol. 9, pp. 87094–87155, 2021.

[5] Z. Xiong and N. Zilberman, "Do switches dream of machine learning?: Toward in-network classification," in *Proc. 18th ACM Workshop Hot Topics Netw.*, 2019, pp. 25–33.

[6] B. M. Xavier, R. S. Guimarães, G. Comarela, and M. Martinello, "Programmable switches for in-networking classification," in *Proc. IEEE Conf. Comput. Commun. INFOCOM*, 2021, pp. 1–10.

[7] G. Xie, Q. Li, Y. Dong, G. Duan, Y. Jiang, and J. Duan, "Mousika: Enable general in-network intelligence in programmable switches by knowledge distillation," in *Proc. IEEE Conf. Comput. Commun. INFOCOM*, 2022, pp. 1938–1947.

[8] M. Saqib, Z. A. Hmitti, H. Elbiaze, and R. H. Glitho, "An accurate & efficient approach for traffic classification inside programmable data plane," in *Proc. IEEE Global Commun. Conf.*, 2022, pp. 6331–6336.

[9] A. Sivanathan et al., "Classifying IoT devices in smart environments using network traffic characteristics," *IEEE Trans. Mobile Comput.*, vol. 18, no. 8, pp. 1745–1759, Aug. 2019.

[10] I. Žliobaitė, M. Pechenizkiy, and J. Gama, "An overview of concept drift applications," in *Big Data Analysis: New Algorithms a New Society*. Cham, Switzerland, Springer, 2016, pp. 91–114.

[11] A. Azab, M. Khasawneh, S. Alrabaee, K.-K. R. Choo, and M. Sarsour, "Network traffic classification: Techniques, datasets, and challenges," *Digit. Commun. Netw.*, to be published.

[12] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019.

[13] B. Chakraborty, D. M. Divakaran, I. Nevat, G. W. Peters, and M. Gurusamy, "Cost-aware feature selection for IoT device classification," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11052–11064, Jul. 2021.

[14] A. Telikani, A. H. Gandomi, K.-K. R. Choo, and J. Shen, "A cost-sensitive deep learning-based approach for network traffic classification," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 1, pp. 661–670, Mar. 2022.

[15] Y. Xu and D. Xu, "Maximizing profit of network InP by cross-priority traffic engineering," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2018, pp. 1–6.

[16] M. Saqib, H. Elbiaze, and R. Glitho, "A profit-aware adaptive approach for in-network traffic classification," in *Proc. IEEE Int. Conf. Commun.*, 2023, pp. 3351–3356.

[17] C. De Alwis et al., "Survey on 6G frontiers: Trends, applications, requirements, technologies and future research," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 836–886, 2021.
[18] D. Nechay, Y. Pointurier, and M. Coates, "Controlling false alarm/discovery rates in online Internet traffic flow classification," in *Proc. IEEE INFOCOM*, 2009, pp. 684–692.
[19] S. S. Wanode, M. Anand, and B. Mitra, "Optimal feature set selection for IoT device fingerprinting on edge infrastructure using machine intelligence," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2022, pp. 1–6.
[20] S. Zhu, X. Xu, H. Gao, and F. Xiao, "CMTSNN: A deep learning model for multi-classification of abnormal and encrypted traffic of Internet of Things," *IEEE Internet Things J.*, vol. 10, no. 13, pp. 11773–11791, Jul. 2023.
[21] B. Afzal, S. A. Alvi, G. A. Shah, and W. Mahmood, "Energy efficient context aware traffic scheduling for IoT applications," *Ad Hoc Netw.*, vol. 62, pp. 101–115, Jul. 2017.
[22] M. Leconte, G. S. Paschos, P. Mertikopoulos, and U. C. Kozat, "A resource allocation framework for network slicing," in *Proc. IEEE Conf. Comput. Commun. INFOCOM*, 2018, pp. 2177–2185.
[23] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: An overview," 2020, *arXiv:2008.05756*.
[24] L. Zhang and S. Geng, "The complexity of the 0/1 multi-knapsack problem," *J. Comput. Sci. Technol.*, vol. 1, no. 1, pp. 46–50, 1986.
[25] "bmv2." Accessed: Nov. 12, 2022. [Online]. Available: https://github.com/p4lang/behavioral-model/,
[26] "P4 language consortium 2019. P4Runtime specification. P4 language consortium. rev. 1.3.0." Accessed: Nov. 1, 2023. [Online]. Available: https://github.com/p4lang/p4runtime
[27] P. Bosshart et al., "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.

MUHAMMAD SAQIB (Graduate Student Member, IEEE) received the master's degree in computer science from the University of Engineering and Technology, Taxila, Pakistan, in 2019. He is currently pursuing the Ph.D. degree in computer science with the Université du Québec à Montréal, Montreal, Canada. His research interests include traffic classification and quality of service management in next-generation networks.

HALIMA ELBIAZE (Senior Member, IEEE) received the Ph.D. degree in computer science from Telecom Sud-Paris, France, in 2002. Since 2003, she has been with the Department of Computer Science, Universite du Quebec a Montreal, Montreal, QC, Canada, where she is currently a Full Professor. Her current research interests include network performance evaluation, traffic engineering, and quality of service management in next-generation networks.

ROCH H. GLITHO is a Full Professor with Concordia University, where he holds the Ericsson/ENCQOR Industrial Research Chair in Cloud/Edge for 5G and Beyond. He has held a Canada Research Chair from 2010 to 2020, and prior to joining academia in 2010, he has held several senior technical positions with Ericsson. He is also a Professor Extraordinaire with the University of Western Cape, South Africa.