Augmenting DiffServ operations with dynamically learned classes of services

Davide Aureli, Antonio Cianfrani, Marco Listanti, Marco Polverini, Stefano Secci

Augmenting DiffServ Operations with Dynamically Learned Classes of Services

Davide Aureli, Antonio Cianfrani, Marco Listanti,
Marco Polverini, Stefano Secci

25th November 2021

Abstract

In this work, we provide a Machine Learning framework for augmenting the Differentiated Services (DiffServ) protocol with fine-grained dynamic traffic classification. The framework is called L-DiffServ. It is composed of two classification algorithms able to detect the QoS classes of incoming packets only looking at three packet header fields; the first algorithm, referred to as *Inter-L-DiffServ*, is a semi-supervised classification procedure able to replicate DiffServ classification; the second one, referred to as *Intra-L-DiffServ*, is an unsupervised algorithm for intra-class classification, useful for classes taking large portions of the overall traffic. We apply the latter to the low priority best-effort class. The performance evaluation shows that our solution is able to dynamically classify packets and to detect new QoS sub-classes hence adapting to traffic aggregate characteristics. We also show that network resource management can be improved exploiting the new generated QoS sub-classes: two active queue management algorithms based on WRED and CHOKe show a reduction of the number of sessions affected by packet losses up to 40% with respect to the legacy DiffServ procedure.

1 Introduction

The use of data mining and machine learning techniques for automated network control represents an important milestone in the technology transfer agenda of Internet Service Providers [1]. The promise is that self-driving networks, with autonomous configuration, management and planning operations, could lead to a significant improvement of network performance, both in terms of user quality of experience and ISP infrastructure efficiency. The interest of the scientific and industrial community towards self-driving networks has lead to developments in the area of Self-Organizing Networks (SON) now becoming a key background for 5G [2, 3]. More recently, the integration of Artificial Intelligence principles in programmable IP networks led to various frameworks sometimes referred to as Knowledge Defined Networking (KDN) [4].

In this work we propose a machine learning based methodology, referred to as Learning-powered DiffServ classification (*L-DiffServ*), to augment the efficiency of the de-facto leading legacy Quality-of-Service (QoS) protocol for IP networks, DiffServ [5]. The contribution of our work is to exploit and adapt existing machine learning mechanisms to improve the network management, and not to define a novel machine learning algorithm. Our methodology consists of performing a dynamic and fine-grained classification of traffic classes, leveraging on a pre-existing DiffServ classification. DiffServ allows to define up to 64 different QoS classes, even if in practice operators merged them into four DiffServ macro classes (also known as PHB in DiffServ terminology). The DiffServ CodePoint (DSCP) is used to identify the DiffServ macro class a packet belongs. In the DiffServ architecture edge nodes are responsible for packets classification and DSCP marking, so that core nodes can perform QoS procedures only looking at DSCP. Our work is meant to dynamically classify packets and up-

*D. Aureli, A. Cianfrani, M. Polverini, M. Listanti are with DIET Department, University of Rome "La Sapienza", Rome, 00184 Italy.

†S. Secci is with Cnam, 75003, Paris, France.

date QoS classes, defining sub-classes within original DiffServ macro classes, while the legacy approach is to let the classification stay static [6].

Our idea is to provide an automatic classification procedure able to finalize a 3-step operation: i) to dynamically identify DiffServ macro classes on the basis of the incoming traffic; ii) to increase the granularity of DiffServ classification, detecting QoS sub-classes; iii) to exploit the data driven classification to improve network resources utilization. An interesting outcome of our investigation is that a limited number of header fields (the Protocol, Source Port number and Destination Port values for the dataset we analyzed), are sufficient to identify DiffServ macro classes / QoS sub-classes; in this way, the association of an incoming packet to a specific DiffServ macro class / QoS sub-class can be executed defining classical field-matching rules in edge nodes, allowing for a line-rate classification of incoming packets.

The framework we propose is composed of two main phases:

- *Classification phase.* We provide two version of *L-DiffServ*. The first one, referred to as Inter-Class L-DiffServ classification (*Inter-L-DiffServ*), allows to perform DiffServ classification after a learning procedure on past traffic traces. *Inter-L-DiffServ* is based on a semi-supervised data mining solution, obtained by combining the Linear Discriminant Analysis (LDA) dimensionality reduction algorithm with the K-means clustering technique: starting from a labelled training set, i.e. traffic traces with packets marked with a DSCP, *Inter-L-DiffServ* provides an automatic mechanism to detect Diffserv macro classes for each incoming packet. The second one, referred to as Intra-Class L-DiffServ classification (*Intra-L-DiffServ*), is based on an unsupervised mechanism, obtained by replicating *Inter-L-DiffServ* and replacing LDA with the Principal Components Analysis (PCA) technique: the aim is to provide a novel classification for a specific DiffServ macro class (e.g., the dominant one in terms of volume) to be exploited by network providers when performing resources management. In our case, taking into account the features of the dataset used for the numerical results, we provide a novel classification of Best Effort traffic, i.e. traffic with no QoS guarantees.
- *Resource management phase.* We demonstrate that the fine grained classification of traffic obtained with *Intra-L-DiffServ* can be used to improve network performance. More in detail we focus on Best Effort traffic, providing a solution able to exploit Active Queue Management (AQM) strategy in network nodes to reduce the number of flows affected by losses. In this way, the network operator has an autonomous solution for concentrating packet losses, in cases of network overload, to a subset of low priority traffic, i.e. a specific QoS sub-classes. It is important to remark that the detection of QoS sub-classes is not static, but it is the outcome of the classification phase performed in a dynamic way, i.e. traffic classes are updated in runtime.

We open source the L-DiffServ code in [7], and we worked on open data.

The paper is organized as follows: in Section II the state of art is described while motivating our dynamic QoS management idea. The two algorithms proposed for the dynamic classification are described in Sections III and IV, respectively. In Section V the QoS aware resource management solution is defined, while in Section VI the performance evaluation is provided. Finally, we conclude in Section VII.

2 Background

The application of standard statistical and data-mining methods to IP traffic classification has been a well established research domain for the last two decades, leading to many industrial outcomes such as deep-packet-inspection engines and expert firewall rules. The research in this area has more recently evolved toward the adoption of machine learning and artificial intelligence methodologies, which become efficient enough to scale with large traffic volumes thanks to increased software quality and computing capacity, and hardware acceleration solutions.

The major difficulty in this field is dealing with the (partial) presence or the absence of prior labels to traffic records. Hence a challenge of-

ten addressed is the evaluation of the behavior of supervised methodologies compared to unsupervised techniques. Semi-supervised methods are also making surface in the literature on traffic management; while supervised learning methods are effective when fine granularity differentiation is already defined, and unsupervised methods detect unknown classes similarities, a semi-supervised approach is a combination of them offering the possibility of pursuing both objectives as in [8–14].

Authors in [8] propose the RTC (Robust Traffic Classification) scheme, building on the first semi-supervised framework developed in [9, 10], with the capability of identifying traffic of zero-day applications as well as accurately discriminating predefined application classes by exploiting the prior knowledge about the label of the traffic and detect the unknown traffic through the K-means. As in our work, this clustering technique is used to find the optimal number of centroids (classes) according to the training dataset, and the label assigned to a cluster is the one having the greater number of elements in it.

The use of the K-means algorithm to discriminate among well known flows and unknown ones is also described recently in [11], with the framework called self-adaptive semi-supervised traffic classification system (SSTCS). However, all these developments enhance clustering techniques combining them with semi-supervised methodologies for classifying traffic and evaluating the performance according to the accuracy; our idea is going beyond the accuracy values and test the performance of our QoS differentiation comparing it with the current management of the service classes in the DiffServ architecture.

A large number of works are completely relying on a unsupervised methodology. In [12] authors apply K-means in unsupervised way for the data exploration part; in the classification part they use k-nearest neigbours (KNN) measuring the performance with the Davies-Bouldin Index (a non normalized coefficient). This is the main difference with respect to our clustering method, as in fact we consider the Silhouette Coefficient (a normalized measure) to evaluate the goodness of clustering outcome, enabling comparison between different daily-traces results.

In [13] the behaviour of different ML algorithms, such as K-means, Gaussian Mixture Model and Spectral clustering, is analyzed. The conclusion is that it is possible to distinguish an application from the observation of the first few packets of a TCP connection. In [14] DBSCAN clustering is compared with the K-Means: the experimental results show that both K-means and DBSCAN work really fast and, although DBSCAN has lower accuracy compared to K-means, DBSCAN produces better clusters since it is able to identify traffic "noise" by not including it in the classification. In this work we make a classification comparison between our proposal and Abacus framework presented in [15]. Abacus is based on a supervised algorithm, i.e. Support Vector Machine. Abacus is able to classify applications starting from session level data, i.e. Netflow ones, considering as traffic features the source and destination IP addresses, the port numbers, and the packets/byte count. The authors show that Abacus correctly classifies applications with an accuracy greater than 90% when a heterogeneous mix of applications is considered. Our solution highly differs from Abacus, since we are using a semi-supervised approach and our solution only exploits packet level features, thus allowing for a line rate classification of packets, not possible with the Abacus approach.

3 The network scenario:DiffServ and beyond

The possibility of implementing service differentiation mechanisms in IP networks is a strategic feature for network operators. In this way, the operator can stipulate Service Level Agreements (SLAs) with customers, providing different QoS levels on the basis of monetary revenues. The first step of any stateless QoS differentiation mechanism is packet classification, i.e. the identification of the specific QoS class to be associated to any incoming packet. The de-facto standard solution for QoS support in IP networks is DiffServ [5]. In a DiffServ network, any incoming packet is marked at an edge router with a DiffServ CodePoint (DSCP), representing its QoS class; the DSCP is used by core routers to identify network resources (mainly buffers and bandwidth) to be provided to each crossing packet.

The DSCP in IPv4 is a six bits string, thus it

allows to define up to 64 different QoS classes. In practice, four different QoS classes, referred to as Per Hop Behavior (PHB), are recommended and used by operators [6]: i) Best Effort (BE), providing classical Best Effort service, ii) Expedited Forwarding (EF), providing guaranteed service in terms of delay, jitter and loss, iii) Assured Forwarding (AF), providing assurance of delivering with limitations with respect to EF, and iv) Class Selector (CS), providing a dedicated service for control traffic (such as routing protocols messages) and for backward compatibility. In the following we use the term DiffServ macro class to identify the PHB used by operators.

The DiffServ macro class association is performed in a static way in today networks: the network operator, on the basis of specific rules manually (by human operators) inserted in edge routers, mark each incoming packet with the proper DSCP. The static setting has a main drawback: packets not belonging to manually configured high-priority classes are always treated as low-priority ones. In practice, any time a new traffic flow with QoS requirements appears, a manual configuration in edge routers is needed to install a new classification rule.

In this work we investigate the possibility of implementing a dynamic classification of packets in a DiffServ network. The idea is to exploit a machine learning classification procedure able to:

- detect DiffServ macro classes of incoming packets on the basis of past traffic traces collected in a centralized manner;
- exploit only header fields as traffic features for the learning procedure;
- detect and install classification rules in edge routers to perform classification at line rate.

More precisely, our machine learning framework, referred to as **Learning-powered DiffServ classification (L-DiffServ)**, focuses on two different aspects: i) improving the dynamicity of DiffServ classification and ii) introducing a deeper QoS classification level for a specific DiffServ macro class. The results of our investigation are two different solutions. The former one, referred to as Inter-Class L-DiffServ classification (*Inter-L-DiffServ*), focuses on learning network traffic classification rules and dynamically detect DiffServ macro classes of new traffic flows; *Inter-L-DiffServ* is based on a semi-supervised classification mechanism. The latter aspect is the definition of a higher granularity traffic classification procedure starting from DiffServ one. Our second proposal, referred to as Intra-Class L-DiffServ classification (*Intra-L-DiffServ*) is based on an unsupervised classification mechanism and leads to the definition of novel QoS sub-classes. The novel QoS sub-classes will be used to implement more efficient QoS strategies.

The actual goals of our framework can also go beyond the dynamic classification of new traffic flows without manual flow marking and the improvement of QoS strategies. The possibility of dynamically classifying traffic can be exploited in the following ways: i) L-DiffServ modules can be used for applications to perform classification refinement; ii) it could also allow not using or hiding the DSCP marking in part or the whole network path for confidentiality or security reasons; iii) the detection of additional QoS classes worth being differentiated may so be spotted, as a form of anomaly detection.

From an architectural point of view, our idea is to integrate our L-DiffServ classification procedure into the ISP Management/Control Plane. An high level scheme of the classification function block at Management/Control level is reported in Figure 1:

- traffic traces (actually only few fields of layer 3 and 4 headers are needed) are collected from edge routers, as part of classical management procedures, and are used as training data set for our classification algorithms, i.e. *Inter-L-DiffServ* or *Intra-L-DiffServ*;
- the outcome of the classification algorithm is the set of DiffServ macro-classes/ QoS sub-classes (clusters);
- classification rules, defined as matching conditions on specific header fields, are extracted from DiffServ macro-classes/ QoS sub-classes and are installed in edge routers.

The classification of incoming packets becomes an automatic function, executed in a dynamic way by the Management/Control Plane. The training is performed continuously offline, on the basis of traffic traces collected by edge nodes. The QoS packet marking is performed at line rate by edge

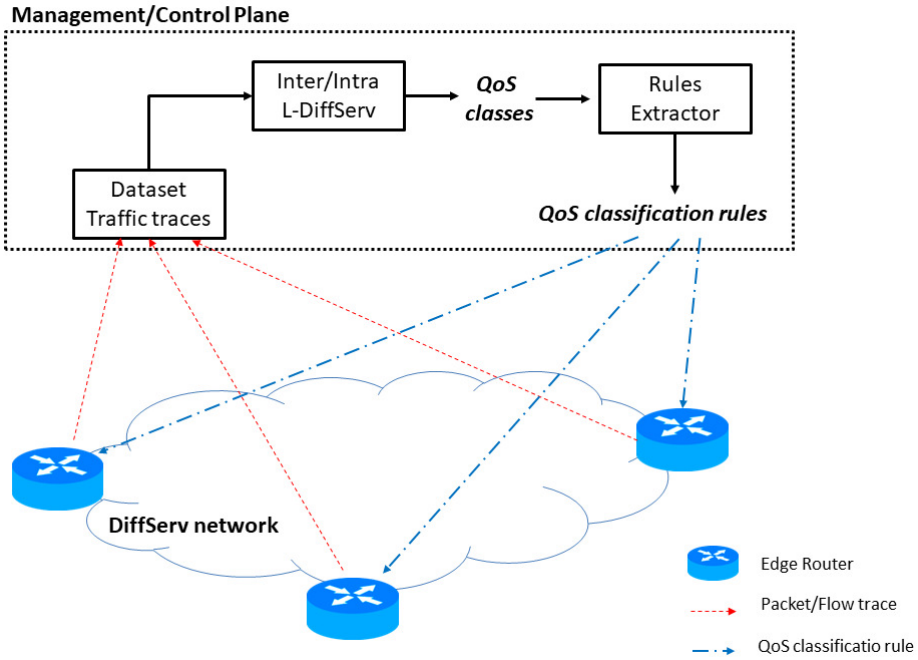


Figure 1: Architectural scheme of our solution in the Management/Control plane of a DiffServ network.

nodes, using matching conditions detected by our *L-DiffServ* solution analyzing past traffic traces. The QoS matching conditions are configured and updated by the Management/Control Plane : the collection of traffic traces allows to feed the classification algorithm with new traffic flows. In this way,*L-DiffServ* is able to adapt QoS classification to traffic behavior, with no need of human operator intervention.

4 Inter-Class L-DiffServ

The aim of the *Inter-L-DiffServ* algorithm is to provide an autonomous traffic classification replicating the standard DiffServ classification. *Inter-L-DiffServ* exploits the availability of a DiffServ labelled training dataset to define header-based classification rules.

The *Inter-L-DiffServ* classification procedure is based on a classical Machine Learning approach: i) an initial training phase is implemented on historical data to detect traffic clusters, ii) DiffServ macro classes are associated to the different clusters, and iii) testing is performed on new packets, not included in the traffic used during the training phase, to evaluate the capability of *Inter-L-DiffServ* in implementing a dynamic DiffServ classification.

The training process is composed of three main phases: i) preprocessing and oversampling, ii) dimensionality reduction and iii) clustering and DiffServ macro classes identification. After the detection of DiffServ macro classes, the testing phase can be executed, i.e. new incoming packets are classified on the basis of obtained clusters.

4.1 Preprocessing and oversampling

The *Inter-L-DiffServ* training procedure is shown in Figure 2. It takes as input a traffic trace with labelled data, i.e. data with DiffServ codepoints available. The traffic trace, also referred to as training dataset, is a packet level trace, i.e. a file captured on wire by a packet sniffer.

The first step of the Preprocessing phase is the characterization of input data in terms of features to be considered for the classification procedure.

For each received packet, our approach performs a multi-layer features evaluation: all the IP and Layer 4 header fields are extracted and evaluated. Investigating the data-set used (more details in Section 6.1), and exploiting the availability of the DiffServ codepoints, we are able to detect the correlation between each header field and the DiffServ classification. The result of this first evaluation was quite surprising: the features really useful to perform a similar DiffServ classification are the Protocol Number, the Source Port and the Destination Port. The previous result allows for a high flexibility of *Inter-L-DiffServ* in terms of input data to be accepted. The possibility of focusing only on Protocol Number, Source Port and Destination Port allows for considering also session level traces, i.e. a Netflow trace, as input data. This aspect represents a first improvement with respect to our previous work [16], where only packet level traces were supported as input file. The capability of working with sessions makes our proposal more flexible, allowing to highly reduce training time (a set of packets is represented as a single element of the session trace), and also avoiding data inconsistency (packets of the same sessions with different DSCPs). It is important to remark that our solution is still valid in a different scenario, i.e. when more header fields are used as data features; anyway, in the rest of the work we consider as data features only the Protocol Number, the Source Port and the Destination Port [1].

In the Preprocessing phase, two further operations are performed:

- transformation of categorical variables into binary variables by means of the One-Hot Encoding tool [17]; it transforms a variable with n observations and d values, to d binary dichotomous variables with n observations, each observation indicating the presence with 1 or absence with 0 of the variable;
- detection of the most significant port numbers; the port numbers taken as features are the ones used by the 90% of the training traffic, while the remaining ones are replaced by the categorical variable 0.

[1] for raw IP packets, such as routing protocols control messages, the Source and Destination port numbers are set equal to value -1, i.e. all raw IP packets are classified only on the basis of the Protocol value

We applied the One-Hot Encoding to the three categorical variables which are transformed into about 500 binary variables (the exact number of final variables depends on the training dataset); the huge number of binary variables shows the great heterogeneity of the traffic analyzed.

The successive step is the evaluation of the traffic distribution among DiffServ macro classes for the training dataset. This is a classical step in any classification algorithm since it is very frequent to have an unbalanced dataset, that could lead to "cover" information related to classes with a limited amount of data. In our cases, as better explained in the Performance Evaluation section, the Best Effort class represents more than the 90% of the dataset.

To cover with traffic unbalance within DiffServ macro classes, it is needed to adopt an Oversampling technique, i.e. to increase the size of underrepresented classes. Two different approaches can be used to perform oversampling: i) Random Oversampling and ii) SMOTE [18]. In the former case, the underrepresented class size is increased by adding new samples obtained by randomly selecting values already present in the dataset and belonging to the same class. In the latter case, used in [16], new samples are obtained by interpolating data values belonging to the same class. Considering the packet/session level traffic data to be processed, we decided to use Random Oversapling technique: i) Random Oversampling maintains the diversity of the training data, i.e. no novel values are added to the dataset; ii) SMOTE could lead to unfeasible data values, i.e. data having fields not feasible in a traffic trace. In particular, this last aspect regards Protocol and Port Number values that, as dichotomous variable, can assume only two values, i.e. 1 (presence) or 0 (absence): with Smote not integer values are possible.

At the end of the Random Oversampling procedure, the training dataset is composed of an equal number of data for each DiffServ macro class.

4.2 Dimensionality reduction

The outcome of the Oversampling phase is a new dataset, where new data is added to the original ones and variables are transformed. As a consequence, the size of the new dataset is higher with respect to the original one and this could be a prob-

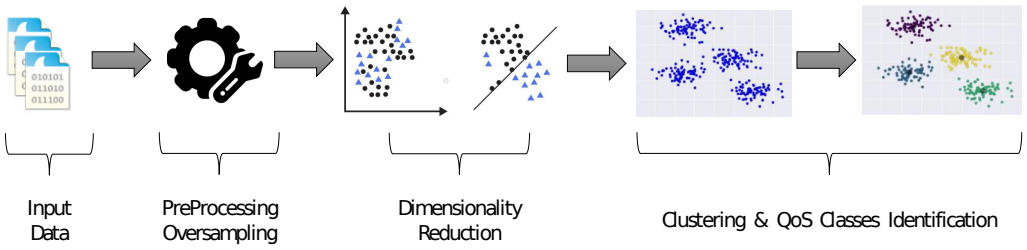


Figure 2: Description of the main steps of *Inter-L-DiffServ*.

lem in terms of complexity. This aspect is known in the Machine Learning field as the "Curse of dimensionality".

The classical approach to overcome this limitation is to extract the most informative variables reducing the space dimensionality, applying a so called dimensionality reduction technique. Exploiting a labelled dataset, it is possible to define a new space where the variance between different classes is maximized: to do that we used the known Linear Discriminant Analysis (LDA) classifier, able to perform space dimensionality reduction. The main advantage with respect to unsupervised approaches, i.e. when no labelled data are available, is that LDA maximizes the variance between different DiffServ macro classes and not within them. In our solution we fix to three the new space dimension, since in this way more than the 90% of the total variance is obtained. Each new axis, referred to as Linear Discriminant (LD), leads to a new variable, obtained as a combination of the original ones. To have an idea about the LDA procedure, we report in Table 1 the composition of the three LDs after the execution of LDA on a reference training dataset.

The final step of the dimensionality reduction phase is to represent the training dataset in the new space defined by LDA.

4.3 Clustering and DiffServ macro classes identification

Starting from the new space representation, it is possible to detect DiffServ macro classes, referred to as clusters, applying a clustering technique. In our solution we use the well-known K-means algorithm, mainly due to its good performance and low complexity. However, the outcome of K-means must be characterized to evaluate its goodness. To do that, a similarity parameter is used to characterize the obtained clusters. In this work, we use the Silhouette Coefficient, i.e. the average value of the Silhoutte index computed for each cluster [19]. The Silhouette coefficient provides a metric evaluation about the goodness in the clustering results. It compares the cohesion and dispersion for each element clustered respect to its own cluster and the others. The range of this metric goes from -1 to 1, higher values of the index suggest an higher performance in the clustering matching.

Thus, the idea is to set the number of obtained classes equal to the number of centroids that maximize the average Silhouette Coefficient. Before proceeding with DiffServ macro classes definition,

Table 1: The 10 most significant variables for each Linear Discriminant (LD) after executing LDA on a reference traffic trace

LD1	
Variable	**Value**
S Port 62208	5.80%
S Port 123	3.69%
D Port 443	3.26%
S Port 50213	2.64%
D Port 53	2.64%
S Port 45198	2.64%
S Port 52404	2.60%
S Port 57651	2.57%
D Port 80	1.58%
S Port 63914	1.27%

LD2	
Variable	**Value**
S Port 123	19.75%
D Port 443	2.79%
S Port 62208	2.28%
S Port 8080	2.15%
S Port 63685	2.15%
S Port 50213	2.13%
S Port 47125	2.13%
S Port 52404	2.13%
S Port 21	1.87%
S Port 1883	1.80%

LD3	
Variable	**Value**
S Port 123	5.79%
Protocol 47	3.58%
Protocol 97	3.58%
D Port 443	2.82%
S Port 21	2.20%
S Port 8195	2.20%
S Port 993	2.20%
Protocol 1	2.13%
S Port 1883	1.89%
S Port 22	1.80%

a refinement step must be executed. Each cluster must be evaluated independently to detect clusters having a low Silhouette Coefficient. These clusters must be removed from the final result, since they represent clusters with mixed traffic, i.e. packets belonging to different DiffServ macro classes. In our case, after empirically evaluating different values, we set the Silhouette Coefficient threshold to 85%: all clusters having a Silhouette Coefficient lower than 85% are removed.

The final step is the DiffServ macro classes definition. Each cluster is marked with the DiffServ macro class having the higher number of packets in the cluster. The outcome of the classification phase is the set of DiffServ macro classes.

4.4 On-line classification

The last phase of any classification procedure is the testing phase: the clusters obtained from the training phase are used to perform classification on new data. In our case, new incoming packets must be classified in edge nodes, with the assignment of a DiffServ macro class. More precisely, each packets is classified considering its "euclidean distance" with respect to the centroids of the clusters obtained from the training phase: the DiffServ macro class having the lowest distance to the packet is selected.

To perform the previous operation in a real ISP network, the classification of incoming packets must be performed on-line in edge routers. In other words, the testing phase must be implemented at line rate in router line cards. The features of *Inter-L-DiffServ* allows for such an implementation since:

- classification is performed only looking at Protocol, Source Port Number and Destination Port Number fields; a router is able to obtain such values looking at the packet header, exploiting fast hardware level lookup mechanisms;

- clusters are defined in a new space where the same three fields are used; as a consequence, each cluster can be identified by rules based on such fields (as a matter of example a cluster can be described with a specific Source Port Number and/or a specific Protocol value).

Exploiting the previous considerations, it is possible to translate DiffServ macro classes into field-matching classification rules to be installed in edge nodes in an SDN-like way. Each entry is a combination of Protocol, Source Port and Destination Port fields.

5 Intra-Class L-DiffServ

The *Inter-L-DiffServ* algorithm is based on the availability of a labelled training dataset and allows to replicate the same DiffServ classification procedure statically realized today by service provider. The practical implementation of the proposed solution has two main requirements: i) the availability of a labelled dataset, and ii) the importance of delivering the correct traffic classification. This last aspect is the most problematic: if a high priority packet is not correctly detected as such by *Inter-L-DiffServ*, the SLA contracted with the customers is not respected, with a negative consequence on the ISP reputation. Moreover, the misclassification cannot be considered as a rare event, since the detection of high priority packets can be based on rules not related to headers fields; in such cases, *Inter-L-DiffServ* is not able to detect high priority packets.

For this reason, we define here a novel application of the L-DiffServ logic, focusing on a given DiffServ macro class. Exploiting the idea of the *Inter-L-DiffServ* algorithm, we propose to detect QoS sub-classes inside a DiffServ macro class, performing an unsupervised classification. We refer to this L-DiffServ variant as Intra-Class L-DiffServ classification (*Intra-L-DiffServ*). In other words, we are providing a new tool for network operators to enhance the classification granularity for a given DiffServ macro-class, a class that may deserve finer classification because of high volume or traffic heterogeneity within its aggregate. Therefore the choice about the DiffServ macro class/classes to be splitted into QoS classes, is a network provider decision: it depends on the provider policies and on the traffic type. In this work, as better explained in Section VII, we focus on BE traffic since the dataset we used is mainly composed of BE traffic.

Intra-L-DiffServ is composed of the same steps of *Inter-L-DiffServ*, with two significant differences. The first one regards the training dataset: it is composed only of low priority packets. The second one regards the Dimensionality Reduction phase; considering that there is no difference among packets in terms of DSCP, it is completely unsupervised: the algorithm used is the Principal Component Analysis (PCA) one. The new variables computed by PCA, referred to as Principal Components, are linear combinations of the initial variables; PC computation is based on the Covariance matrix, composed of the correlations between all the possible variables pairs. The new space is obtained from the eigenvalues and eigenvectors of the Covariance matrix, ordering the eigenvectors in a decreasing order with respect to the eigenvalues.

The outcome of *Intra-L-DiffServ* is the set of subclasses related to the initial DiffServ macro class X (where X can be BE, EF, AF, etc..), simply QoS sub-classes in the following. In the next section we provide a resource management solution that allows to exploit the QoS sub-classes availability to improve the efficiency of resources utilization.

6 QoS-aware Resource Management

The resources management in a QoS aware network is based on the differentiation of resources assignment/access between QoS classes. The resources to be managed are the links bandwidth and the routers buffers. More in detail, the output links of routers represent the core elements of the QoS differentiation strategies: the availability of multiple output buffers, i.e. a queue for each QoS class, and the definition of a proper scheduling algorithm allow to differentiate the network services in terms of bandwidth assignment, end-to-end delay and packet loss probability.

In this work, we provide a simple yet efficient mechanism to manage routers buffers so that to exploit the dynamic *Intra-L-DiffServ* classification defined in previous Section. We use the Weighted Random Early Detection (WRED) solution [20] for the management of the buffer assigned to the DiffServ macro class "partitioned" by *Intra-L-DiffServ*. We implement WRED so that each QoS sub-class i can be associated to a specific threshold value, referred to as τ_i. Considering that the QoS subclasses are dynamically computed, the association

between a QoS sub-class and its threshold varies in time.

WRED functioning depends on the buffer occupancy, referred to as B_0. The queue of BE sub-class i starts the dropping procedure when the buffer occupancy reaches threshold τ_i; more precisely, the probability of dropping a packet of BE sub-class i increases linearly from 0, for $B_O = \tau_i$, to 1, for B_O equal to the buffer size.

The outcome of the proposed solution is a service differentiation among QoS sub-classes in terms of packet loss. The QoS sub-classes having lower thresholds experience a higher packet loss with respect to QoS sub-classes with higher thresholds. In our case, we propose to implement WRED mechanism on a single QoS sub-class; in this way the losses are concentrated on a limited amount of sessions, i.e. the ones belonging to the selected QoS sub-class. This idea can be exploited by service providers to differentiate sessions belonging to the same DiffServ macroclass: the lower priority sessions of the selected DiffServ macroclass can be dynamically detected and an improved service, in terms packet losses, can be provided to higher priority sessions of the same DiffServ macroclass. The selection of the QoS sub-class to be marked as low priority one between the ones detected by *Intra-L-DiffServ* remains an operator choice.

Moreover, we consider different threshold values τ_i. It is clear that decreasing the threshold value can lead to a better service differentiation; on the other side, a low threshold value increases the overall packet loss, since BE packets are to be discarded next, i.e. when the buffer still have available space. Moving to high values of the threshold reduces the overall packet loss but it does also reduce the service differentiation among BE sub-classes.

It is important to remark that our WRED based proposal represent one of many possible solutions to exploit *L-DiffServ* classification. Different approaches, considering more complex buffer management strategies and advanced scheduling algorithms, can be considered as future directions of our research work.

7 Numerical results

We numerically evaluate our proposal as follows. After describing the real dataset we used, we evaluate the performance of *Inter-L-DiffServ*, showing the capability of our solution in detecting the proper QoS class when new traffic flows are managed by the network. Finally, we evaluate *Intra-L-DiffServ* showing how our unsupervised solution is able to define new QoS sub-classes to improve network performance. Our code is available at [7].

7.1 Dataset description

The dataset, referred to as MAWI archive [21], is an ongoing collection of daily Internet traffic traces captured within the WIDE backbone network (identified by AS 2500). WIDE interconnects universities and research institutes in Japan to the Internet, and peers with major Autonomous Systems. The analysis provided in [22], showed that traffic crossing WIDE includes both academic and commercial traffic. Each IP trace is a PCAP file containing 15 minutes of traffic and its size is 20 GB, corresponding to about 200 millions packets. Traffic traces are anonymized to hide any personal information, removing application data and scrambling IP addresses with the Crypto-PAn Algorithm [23].

In our analysis, we consider traces of multiple days across multiple weeks. The traffic traces are captured at samplepoint-G, that monitors a 10 Gbps link connecting WIDE network with a peer AS. We work on the period from Apr. 10, 2019 to May 10, 2019; for that period MAWI provides file traces for a single day of the week.

In the traffic analysis, we handle only IPv4 packets. For each trace we consider as training dataset the inial part of the trace, corresponding to 10 Million Packets [2]; then, the classification is performed on the last 200,000 packets of the daily-trace.

Observing the DSCP of packets in our dataset, we decided to modify the recommended DiffServ macro classes [6] in the following way:

- adding a new DiffServ macro class, i.e. the Scavanger one, used to mark a not negligible number of packets in our trace; Scavanger class is used by network provider to identify services with a lower priority than BE class (in some enterprises, applications such as music download are marked with Scavanger label);

[2] This is essentially a computing resource constraint

- renaming the CS macro class as Network and Internetwork Control (NIC) class, since no other CS classes are present in our dataset.

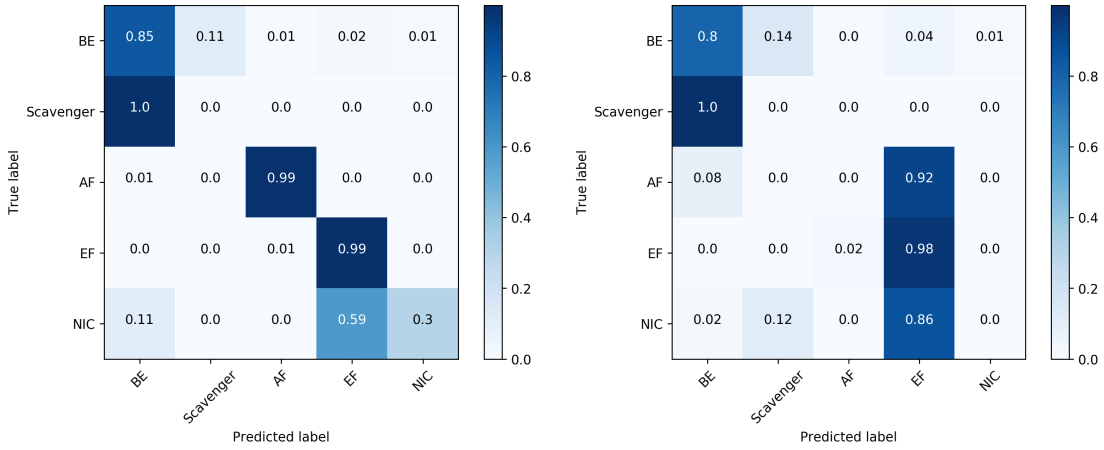
7.2 Inter-L-DiffServ results

We compare the *Inter-L-DiffServ* classification outcome to the DiffServ classification one. The DiffServ classification is available in the dataset since each packet is marked with its DSCP. As already described before, the training dataset is composed of an initial part of the trace, while the last 200 thousand packets represent the testing dataset. To provide a meaningful evaluation, we only focus on new sessions in the training part, i.e. packets belonging to sessions not already present in the training dataset: in this way, the capability of *Inter-L-DiffServ* to apply learned rules to new traffic sessions is really investigated.

In Figure 3 we report the confusion matrices for four different days, focusing on DiffServ macro classes (we group our QoS sub-classes in the related DiffServ macro class). The confusion matrix associates to each DiffServ macro class (True label in the figure) the true positive rate, i.e. the percentage of packets assigned to a specific QoS class by our algorithm (Predicted label). A first important observable outcome is that *Inter-L-DiffServ* is able to properly detect BE traffic: the true positive rate is always higher than 80%; moreover, this value increases to 94% if we also considered the Scavanger class, i.e. traffic with lower priority than BE. Another aspect regards the low performance related to high priority traffic. Looking at AF and EF QoS classes, our solution has high true positive rate in some days (Figure 3(a)), while it has low ones in different days (AF for Figure 3(c)).

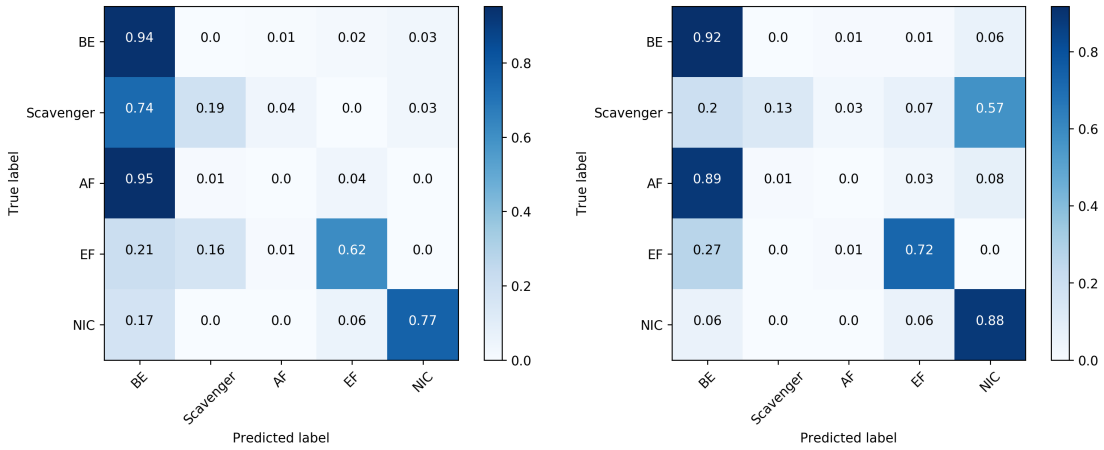
To better understand the low accuracy in classifying high priority traffic, we performed a detailed characterization of AF and EF traffic in our dataset. This outcome of this evaluation was unexpected: in some cases, AF/EF packets belonging to the same session (same IP source/destination addresses, source/destination ports and protocol) have a different QoS classification. The observed behavior can be the consequence of extra traffic generated by a specific high priority customer/service, that overcomes the allowable AF/EF traffic agreed and then receives a lower priority treatment. The previous outcome suggests that the network operator performs high priority traffic classification following commercial rules that cannot be inferred from packet header fields. As a consequence, in many cases *Inter-L-DiffServ* fails in detecting the classification rules behind the DSCP assignment for high priority traffic.

The results obtained with *Inter-L-DiffServ* are compared with a different traffic classification algorithm, referred to as Abacus [15]. With respect to our solution, Abacus acts on session level information, i.e. Netflow data, and it also considers IP addresses and packets/bytes number as classification features. In other words, Abacus has a greater action space, taking into account additional features with respect to *Inter-L-DiffServ*, and cannot be implemented online at edge routers, since session related features (packets/byte count) must be computed on the overall set of packets of the same session. For this reason, we considered Abacus as a benchmark for *Inter-L-DiffServ*. The results of Abacus for the same traces of *Inter-L-DiffServ* are reported in Figure 4. Besides the higher number of features evaluated and the possibility of classifying considering the flow level information, the results obtained by Abacus are in line with the ones obtained by *Inter-L-DiffServ*: i) best effort traffic is detected with a good accuracy, even if it is lower with respect to *Inter-L-DiffServ*, while ii) high priority traffic shows a different classification accuracy according to the PHB: for the AF traffic, we can observe a very poor performance, while for the EF traffic Abacus is able to obtain an accuracy greater than or equal to 95%. The difference in EF accuracy between Abacus and *Inter-L-DiffServ* is due to two main reasons. Firstly, performing a deeper investigation on EF traffic, we found that it is characterized by flows generating a limited number of packets. Abacus is able to better classify EF traffic since it exploits the amount of packets/bytes generated by the flow as classification feature. A second aspect is a consequence of data preprocessing for Abacus: the different DiffServ classification of AF/EF packets of the same session, discussed previously. Considering that Abacus is a session-based classification, we removed from the input data all sessions having packets with more than a unique DSCP: in this way the Abacus performance reported in Figure 4 is not affected by the "unusual" multiple classification of a session performed by network operators, that degrades *Inter-*



(a) 10 April 2020.

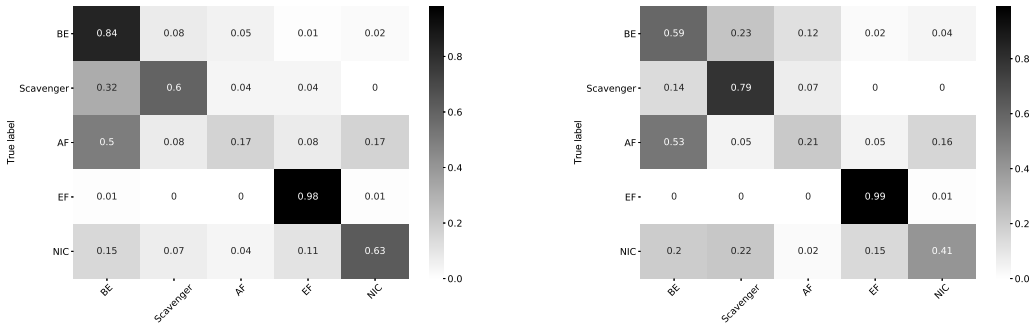
(b) 17 April 2020.



(c) 1 May 2020.

(d) 8 May 2020.

Figure 3: Confusion Matrices related to *Inter-L-DiffServ* are reported for four different days. The obtained QoS class (Predicted label) is compared with DiffServ class (True label)



(a) 10 April 2020.

(b) 17 April 2020.

(c) 1 May 2020.

(d) 8 May 2020.

Figure 4: Confusion Matrices related to Abacus [15] are reported for four different days. The obtained QoS class (Predicted label) is compared with DiffServ class (True label).

L-DiffServ performance. Anyway, the results obtained confirm that such dynamic classification procedures have poor performance in detecting high priority traffic due to the intrinsic nature of the classification strategy performed by network operators.

A second analysis regards the capability of *Inter-L-DiffServ* to dynamically adapt to network changes. In Figure 5 we report the number of QoS sub-classes computed by *Inter-L-DiffServ* in five different days. We show both the total number of QoS sub-classes and the number of sub-classes for each DiffServ macro class. Figure 5 shows that *Inter-L-DiffServ* is able to dynamically change the classification outcome following the traffic features: the number of total sub-classes vary from 5 to 25, while the number of BE sub-classes varies from 1 to 7 in five different days. This result represents also a confirmation of our intuition that analyzing a "few header values" it is possible to adapt QoS classification to traffic features.

The final outcome of the *Inter-L-DiffServ* performance evaluation is that our solution has the capability of characterizing low priority packets features in a dynamic way. For this reason we decided to focus our attention only on BE traffic, which represents also the great part of ISP traffic.

7.3 Intra-L-DiffServ results

The characterization of *Intra-L-DiffServ* performance is based on two different aspects: i) the evaluation of the classification outcome, i.e. the QoS sub-classes detected, and ii) the use of WRED buffer management mechanism to reduce the number of traffic sessions affected by packet losses. Considering that the traffic traces analyzed are mainly composed by BE traffic (more than 90% of packets), we decided to implement *Intra-L-DiffServ* on BE class: in this way, we are splitting the BE macro-class in BE sub-classes.

7.3.1 BE sub-classes evaluation

It is important to remark that *Intra-L-DiffServ* is based on a different Dimensionality Reduction algorithm (PCA), with respect to the *Inter-L-DiffServ* one (LDA). For this reason, the outcome of *Intra-L-DiffServ* differs than the one of *Inter-L-DiffServ* in terms of BE sub-classes detected. In Table 2, we report the number of BE sub-classes computed by *Intra-L-DiffServ* and *Inter-L-DiffServ*, respectively. It is clear that *Intra-L-DiffServ* provides a more granular classification, detecting always more BE sub-classes than *Inter-L-DiffServ*. The reason is that *Intra-L-DiffServ* works only with BE traffic with no need of detecting high priority clusters, and then the BE classification procedure is more accurate. A further outcome is that the classification outcome of *Intra-L-DiffServ* changes over time: once again we proved that traffic QoS features are not static and that our solution is able to adapt to their variability.

Table 2: Number of BE sub-classes for *Intra-L-DiffServ* and *Inter-L-DiffServ*.

	# BE sub-classes	
Day	***Intra-L-DiffServ***	***Inter-L-DiffServ***
10-th April	4	1
17-th April	8	4
24-th April	9	4
1-th May	10	6
8-th May	10	7

To provide a more accurate evaluation, we focus on the obtained BE sub-classes, showing their distinctive features. In Tables 3 and 4 the BE sub-classes defined for the 10^{th} April and 17^{th} April are reported, respectively. Looking at the features description, it is clear that some BE sub-classes are present in both days: for instance, BE 0 sub-class characterized by packets having protocol number equal to 1 or to 50. At the same time, the classification outcome related to the 17^{th} April trace highlights the presence of new BE sub-classes, such as BE 3 characterized by source port 53 and protocol 17.

Table 3: BE sub-classes for the 10-th April trace

BE sub-class	**Feature Description**
BE 0	Protocol 1, Protocol 50
BE 1	Src Port 443 , Src Port 80
BE 2	Dst Port 443 , Dst Port 80
BE 3	Src Port 22 , Dst Port 22

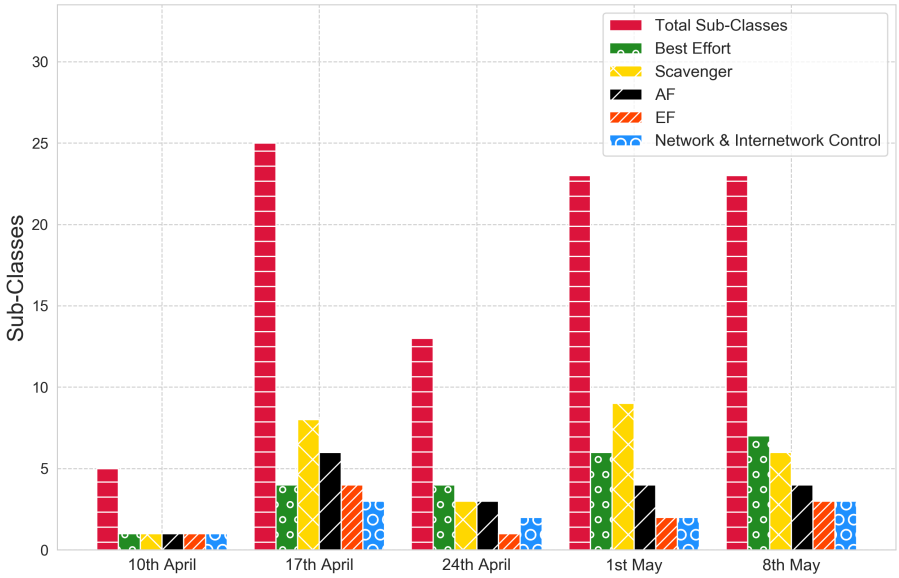


Figure 5: Number of QoS sub-classes computed by *Inter-L-DiffServ* in five different days.

Table 4: BE sub-classes for the 17.th April trace

BE sub-class	**Feature Description**
BE 0	Protocol 1, Protocol 50
BE 1	Src Port 53 , Protocol 6
BE 2	Src Port 80
BE 3	Src Port 53 , Protocol 17
BE 4	Dst Port 443 , Protocol 6
BE 5	Src Port 443, Dst Port 443 , Protocol 17
BE 6	Src Port 443, Protocol 6
BE 7	Dst Port 80

7.3.2 AQM solutions based on *Intra-L-DiffServ*

The second *Intra-L-DiffServ* evaluation aspect regards the possibility of improving network performance exploiting BE sub-classes differentiation. The aim of our WRED modified solution is to exploit the BE sub-classification to reduce the number of session affected by packet losses due to buffer overflow. In particular, as described in previous Section, a specific BE sub-class is selected and a specific WRED threshold τ is associated to it; for all the remaining BE sub-classes, the buffer is managed using a DropTail approach. The criterion to select the BE sub-class with WRED management can be considered as a service provider strategic decision. Looking at the different BE sub-classes, the one having "lower importance" can be detected by applying a customized policy: it could be the BE sub-class where applications with very low priority are present, or the BE sub-class with a limited amount of traffic. In our case we select as BE sub-class with WRED management the one having the lower number of sessions: in this way, we try to minimize the number of sessions impacted by packet losses.

To evaluate packet losses, we simulate the network scenario using an events-based simulator we have developed; we made the code available at [7]. The simulator implements the WRED mechanism and allows to simulate the traffic on the basis of the MAWI traces. The aim of the simulation is to characterize network performance in terms of packet

losses. More precisely, we evaluate the number of traffic sessions affected by packet losses (a session is affected by packet losses if at least one packets is lost). Our solution is compared with the DiffServ one, where the BE traffic buffer is managed with a classical DropTail mechanism with no BE sub-classes. To take into account also high priority packets, we assign network resources to AF, EF and NIC traffic on the basis of traffic trace values (using a Linear Regression model), while the remaining resources are available for BE traffic.

To highlight the improvements of *Intra-L-DiffServ* with respect to standard DiffServ, we defined a generic parameter Δ_x as follows:

$$\Delta_x = \frac{n'_x - n_x}{n_x} \quad (1)$$

where n'_x and n_x represent a specific performance parameter of *Intra-L-DiffServ* and DiffServ respectively.

Session-level evaluation. The first evaluation reported in Figure 6 regards the number of sessions affected by losses for DiffServ and *Intra-L-DiffServ* when different WRED thresholds (τ) are used for the selected low priority BE sub-class. Starting from Eq. (1), we define Δ_1, where n'_1 and n_1 are the number of sessions affected by packet losses with *Intra-L-DiffServ* and DiffServ, respectively. Δ_1 is reported as a percentage value in Figure 6. Given the previous definition, negative values for Δ_1 represent an improvement for *Intra-L-DiffServ*, i.e. our solution reduces the number of traffic sessions affected by packet losses. Figure 6 shows that *Intra-L-DiffServ* outperforms DiffServ in almost all cases: the number of affected sessions decrease up to 40%. Only for two days, when the τ is equal to 0.8, *Intra-L-DiffServ* experiences an increase in the number of sessions affected. This behavior is due to an early dropping of packets in unnecessary cases: packets of the selected BE sub-class are discarded when the buffer exceed the 0.8 threshold even if the buffer occupancy does not reach the buffer size value; in other words, when fixing the WRED threshold to a low value, given the traffic behavior, unnecessary drops could be experienced. A similar situation can be seen looking at Δ_1 behavior with respect to τ in the different days: i) in three days, increasing τ leads to a performance improvement reduction (i.e. Δ_1 increases), while in the remaining ones, increasing τ leads to a performance improvement increase. When traffic is high, i.e. the buffer occupancy does reach the buffer size, having a low WRED threshold allows to improve service differentiation with no performance losses, i.e. unnecessary dropping is not experienced; on the other side, when buffer occupancy is not reaching the buffer size, performing packet dropping with a lower buffer occupancy leads to unnecessary losses. Anyway, when $\tau \geq 0.9$ a performance improvement with respect to DiffServ is always obtained.

BE sub-classes level evaluation. A more detailed evaluation of packet loss distribution among BE sub-classes is reported in Figure 7, where the percentage of BE sub-classes sessions affected by packet losses is reported for DiffServ and for *Intra-L-DiffServ* with different τ values. As expected, *Intra-L-DiffServ* is able to concentrate losses on a lower number of BE-subclasses with respect to DiffServ. More precisely, *Intra-L-DiffServ* results show two different outcomes: i) in Figures 7(a) and 7(b) the losses are concentrated on a single BE-subclass, the one managed with WRED; in ii) Figures 7(c) and 7(d) losses are experienced in three BE-subclasses. Anyway, also in the last case, the BE sub-class implementing the WRED mechanism is the one having the highest packet loss, and the remaining BE sub-classes affected by packet loss have better performance with respect to the classical DiffServ approach. Thus, we can conclude that *Intra-L-DiffServ* allows to focus the losses on a restricted number of BE sub-classes. Moreover, the use of lower τ values improves the service differentiation, i.e. losses are mainly experienced in the selected BE sub-class with low priority.

Comparing the results of Figure 7 with the ones reported in Figure 6 about the number of affected sessions, it is clear that a balance must be chosen between lower threshold values, allowing for a greater service differentiation but leading to higher losses, and higher ones, reducing the service differentiation but improving network performance in terms of packet losses. In our case case, choosing a threshold value equal to 95% seems a good compromise, even if it cannot be considered as a general rule: the threshold value setting is strictly related to the traffic features and to the network

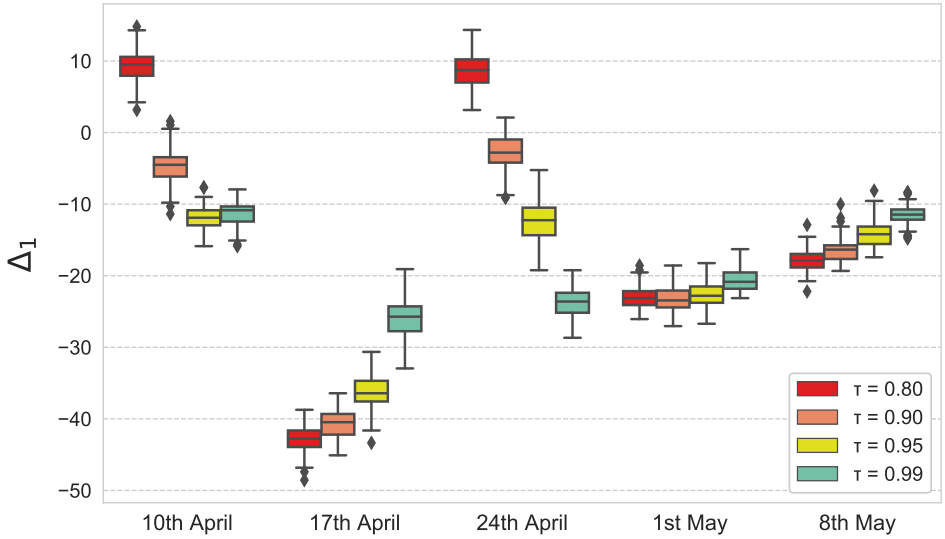
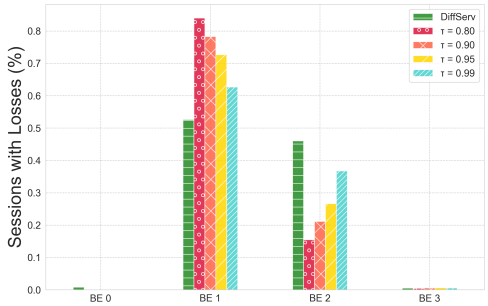


Figure 6: Comparison between the number of sessions affected by packet losses in *Intra-L-DiffServ* and in DiffServ with RED.

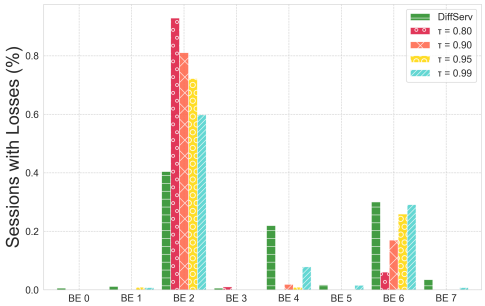
operator policies.

Focusing on the 1-st May trace (similar results are obtained for different days), we report in Figure 8 the evaluation of session-level losses of selected BE sub-classes. More precisely, Figure 8 shows for sub-classes BE 8 (Figure 8(a)), BE 0 (Figure 8(b)), BE 4 (Figure 8(c)) and BE 1 (Figure 8(d)), the distribution of losses among sessions (in percentage): the losses for each sessions are computed as the ratio among the number of packets lost and the overall number of packets generated by the session. The results are reported for two WRED threshold values, i.e. $\tau = 80\%$ and $\tau = 90\%$, and for standard DiffServ. Figure 8(a) shows the packet loss distribution for the low priority sub-class BE 8. The obtained results show a similar behavior for the three cases: there is a negligible higher number of session with lower packet loss for DiffServ with respect to *Intra-L-DiffServ*; the number of sessions with higher losses increases moving from $\tau = 90\%$ to $\tau = 80\%$. This result is a consequence of the *Intra-L-DiffServ* behavior, that tries to focus losses in BE 8 sub-class and is able to increase QoS differentiation with lower WRED threshold values. Looking at different BE sub-classes, in Figures 8(b), 8(c) and 8(d), the difference between *Intra-L-DiffServ* and DiffServ is considerable. *Intra-L-DiffServ* is able to reduce the number of sessions experiencing a significant packet loss, i.e. more than 20% of packet loss, with respect to DiffServ: the percentage reduction is about 40%, 85% and 95% for sub-classes BE 0 (8(b)), BE 4 (Figure 8(c)) and BE 1 (Figure 8(d)), respectively. Regarding the impact of the WRED threshold, decreasing τ allows for better performance in all BE sub-classes except for the low priority one, as expected.

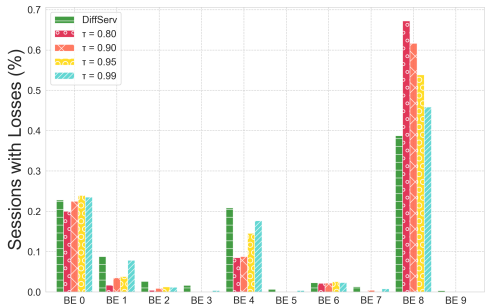
Packet loss and WRED threshold. As a final aspect, we also show in Figure 9 the overall packet loss in DiffServ and in *Intra-L-DiffServ* with different WRED threshold values. Starting from Eq. (1), we define the parameter Δ_2, where n_2' and n_2 are the number of packets lost when using *Intra-L-DiffServ* and DiffServ, respectively. Figure 9 highlights that the difference in terms of packets lost is negligible for $\tau = 95\%$ and $\tau = 99\%$. On the other side, Δ_2 can increase from 3% to 11% when



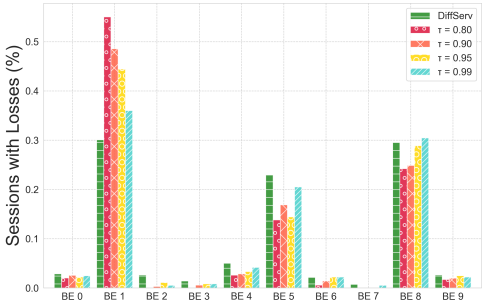
(a) Trace 10 April 2019.



(b) Trace 17 April 2019.

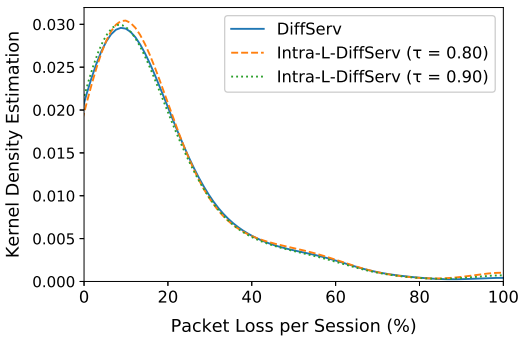


(c) Trace 1 May 2019.

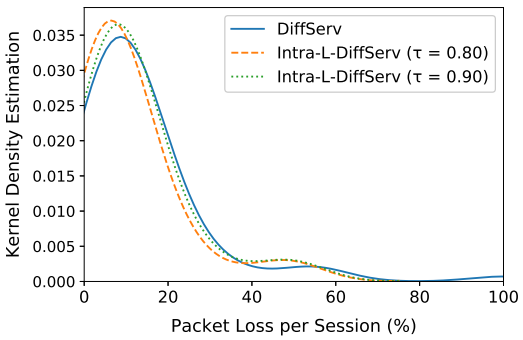


(d) Trace 8 May 2019.

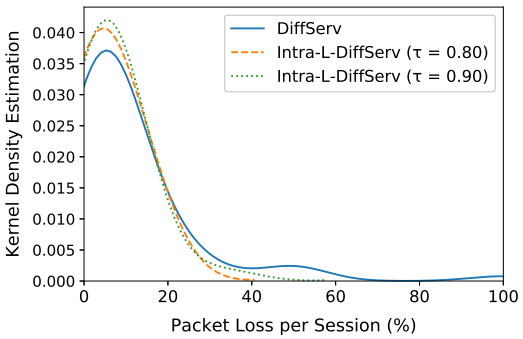
Figure 7: Distribution of losses between BE sub-classes in four different days.



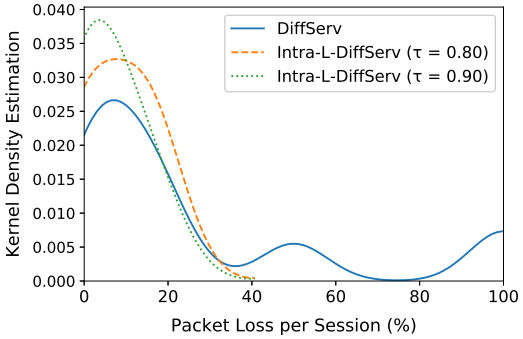
(a) Sub-class BE 8 - 1 May 2019.



(b) Sub-class BE 0 - 1 May 2019.



(c) Sub-class BE 4 - 1 May 2019.



(d) Sub-class BE 1 - 1 May 2019.

Figure 8: Distribution of packet losses at session level for *BE-L-DiffServ* and Diffserv.

$\tau = 90\%$ and from 12% to 30% when $\tau = 80\%$. This is a consequence, as explained above, of an early packets drop in WRED queue: a trade-off between service differentiation and packet losses must be carefully evaluated considering the traffic behavior. We can conclude that the service differentiation provided by *Intra-L-DiffServ* with respect to the number of sessions affected by packet losses is not leading to an overall performance degradation if WRED threshold is properly set. As a general consideration, the network operator should set the WRED threshold to a value not lower than 90%, avoiding a significant packet loss increase, choosing low values (close to 90%) or higher ones (close to 99%) if service differentiation should be enforced or not, respectively.

Comparison with CHOKe mechanism. To show the flexibility of our *Intra-L-DiffServ* solution in providing an efficient sub-classification of BE traffic, we consider an additional active queue management mechanism, i.e. CHOKe [24]. CHOKe improves WRED mechanism by differently penalizing unresponsive sessions, i.e. sessions not modifying their rate when packet losses occur. The penalty consists in the increase of the dropping rate for these sessions. More in detail, when the buffer occupancy is greater than a pre-fixed threshold, an arriving packet and a queued packet randomly selected are both discarded if they belong to the same session. We propose to use CHOKe only for the BE sub-class with lower priority.

The comparison between DiffServ and *Intra-L-DiffServ* with CHOKe is reported in Figures 10 and 11. In Figure 10 the number of sessions affected by packet losses with *Intra-L-DiffServ* and basic DiffServ is shown for different traces. Comparing the results with the ones obtained when WRED is used (Figure 6), it is clear that a similar behavior is present. *Intra-L-DiffServ* allows for a reduction of sessions affected by packet loss in quite all cases: only for two traces, when the CHOKe threshold is equal to 0.8, a slight performance degradation is visible. In all other cases, the number of affected sessions is decreased up to 45%, showing that *Intra-L-DiffServ* classification can be efficiently exploited by CHOKe to provide service differentiation.

In Figure 11 the overall number of packets lost in *Intra-L-DiffServ* with CHOKe with respect to DiffServ is shown. The obtained results are in line with the same ones obtained with WRED (Figure 9). The previous analysis highlights the capability of *Intra-L-DiffServ* in detecting a dynamic sub-classification of BE traffic to be exploited by a general active queue management strategy to provide an effective traffic differentiation in DiffServ networks.

8 Conclusions

In this work, we provide a Machine Learning based solution for the classification of incoming traffic in a service provider network supporting QoS differentiation. We defined the L-DiffServ framework composed of two engines: i) *Inter-L-DiffServ*, an algorithm able to learn an existing DiffServ classification and replicate it properly, and ii) *Intra-L-DiffServ*, providing a novel classification of a selected DiffServ macro class. In both cases, we show that the use of only three header fields (Protocol, Source Port, Destination port) as features is sufficient for the classification. We combine two preprocessing steps before feeding data into our Machine Learning solution: i) an oversampling strategy, to balance the training dataset, and ii) a dimensionality reduction algorithm, to reduce the problem complexity. Once obtained the new dataset, we evaluate the clustering solution to detect QoS subclasses. An outcome of our work is that using our framework it is possible to provide a dynamic and thinner classification of best effort traffic following traffic changes. Moreover, we show how hard it is to classify high priority packets, mainly due to the limitation of header fields as algorithm features.

We show that the method we propose to enhance the QoS classification can be exploited to improve network resource utilization. We defined two QoS aware active queue management solutions: the first one is based on WRED, the second one on CHOKe; in both cases, we show that it is possible to reduce the number of sessions affected by packet losses with respect to classical DiffServ classification. The definition of novel resource management strategies based on *L-DiffServ* represents the next step of our research activity.

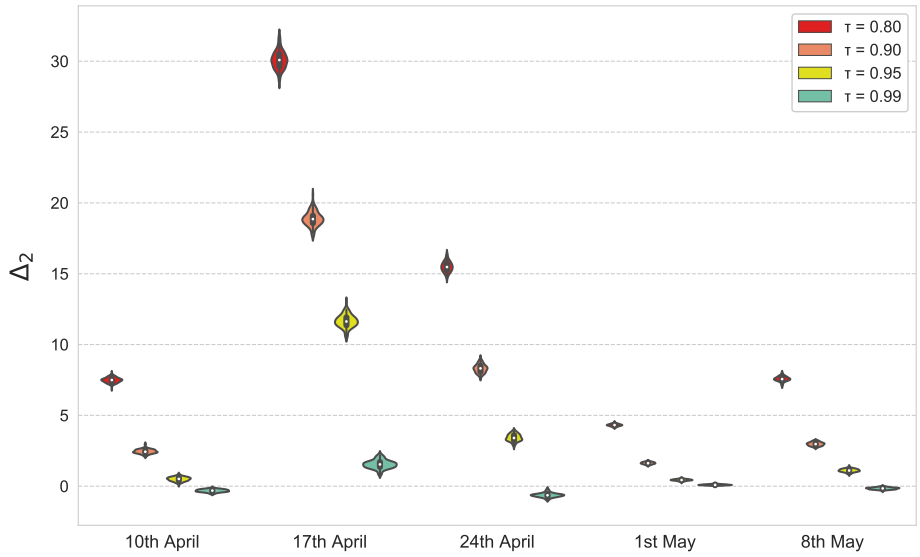


Figure 9: Comparison between *Intra-L-DiffServ* and DiffServ with WRED in terms of overall packets lost.

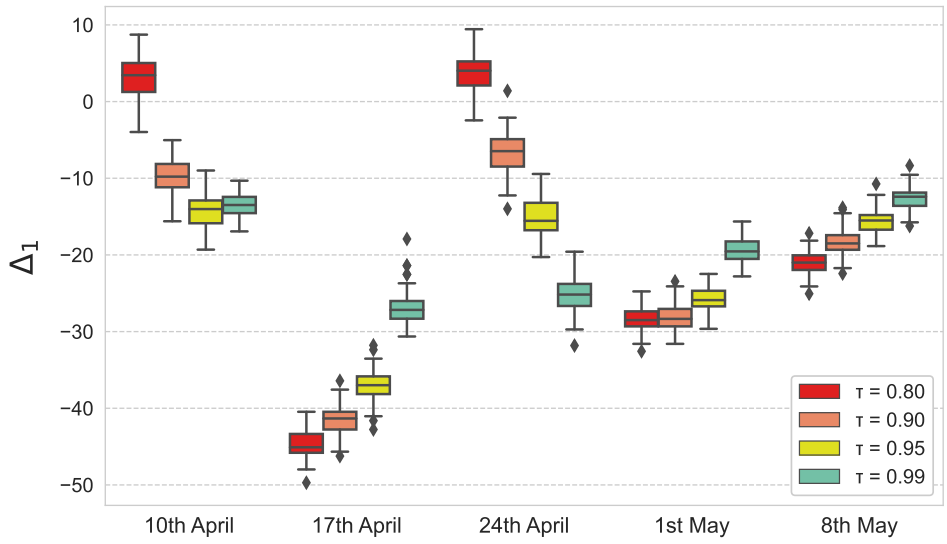


Figure 10: Comparison between the number of sessions affected by packet losses in *Intra-L-DiffServ* and in DiffServ with CHOKe.

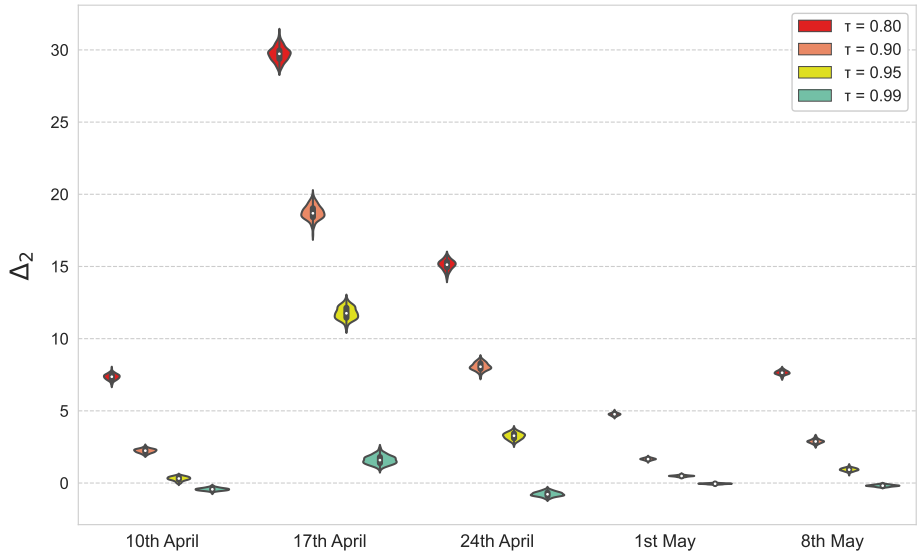


Figure 11: Comparison between *Intra-L-DiffServ* and DiffServ with CHOKe in terms of overall packets lost.

References

[1] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2432–2455, 2017.

[2] J. Moysen and L. Giupponi, "From 4g to 5g: Self-organized network management meets machine learning," *Computer Communications*, vol. 129, pp. 248 – 268, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366418300380

[3] C. Benzaid and T. Taleb, "Ai-driven zero touch network and service management in 5g and beyond: Challenges and research directions," *IEEE Network*, vol. 34, no. 2, pp. 186–194, 2020.

[4] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett, G. Estrada, K. Maruf, F. Coras, V. Ermagan, H. Latapie, C. Cassar, J. Evans, F. Maino, J. Walrand, and A. Cabellos, "Knowledge-defined networking," *SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 3, pp. 2–10, Sep. 2017. [Online]. Available: https://doi.org/10.1145/3138808.3138810

[5] S. Blake et al., "An Architecture for Differentiated Services," IETF RFC 2475, 1998.

[6] J. Babiarz et al., "Configuration guidelines for DiffServ service classes," IETF RFC 4594, 2006.

[7] D. Aureli and M. Polverini, "L-diffserv," 2020. [Online]. Available: https://github.com/davaureli/L-DiffServ

[8] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM transactions on networking*, vol. 23, no. 4, pp. 1257–1270, 2014.

[9] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, "Identifying and discriminating between web and peer-to-peer traffic in the network core," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 883–892.

[10] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and A. V. Vasilakos, "An effective network traffic classification method with unknown flow detection," *IEEE Transactions on Network and Service Management*, vol. 10, no. 2, pp. 133–147, 2013.

[11] J. Ran, X. Kong, G. Lin, D. Yuan, and H. Hu, "A self-adaptive network traffic classification system with unknown flow detection," in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*. IEEE, 2017, pp. 1215–1220.

[12] T. Chen and Y. Zeng, "Classification of traffic flows into qos classes by unsupervised learning and knn clustering," *KSII Trans. on Internet and Information Systems*, vol. 3, no. 2, pp. 134–146, 2009.

[13] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proceedings of the 2006 ACM CoNEXT conference*, 2006, pp. 1–12.

[14] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, 2006, pp. 281–286.

[15] D. Rossi and S. Valenti, "Fine-grained traffic classification with netflow data," in *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, ser. IWCMC '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 479–483.

[16] D. Aureli, A. Cianfrani, A. Diamanti, J. M. Sanchez Vilchez, and S. Secci, "Going beyond diffserv in ip traffic classification," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–6.

[17] S. Raschka, *Python Machine Learning*. Packt Publishing, 2015.

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, p. 321–357, Jun. 2002.

[19] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[20] *Buffer Management*. John Wiley & Sons, Ltd, 2002, ch. 6.

[21] K. Cho, K. Mitsuya, and A. Kato, "Traffic data repository at the wide project," in *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ser. ATEC '00. USA: USENIX Association, 2000, p. 51.

[22] R. Fontugne, P. Abry, K. Fukuda, D. Veitch, K. Cho, P. Borgnat, and H. Wendt, "Scaling in internet traffic: A 14 year and 3 day longitudinal study, with multiscale analyses and random projections," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2152–2165, 2017.

[23] J. Fan, J. Xu, M. H. Ammar, and S. B. Moon, "Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme," *Comput. Netw.*, vol. 46, no. 2, p. 253–272, Oct. 2004. [Online]. Available: https://doi.org/10.1016/j.comnet.2004.03.033

[24] R. Pan, B. Prabhakar, and K. Psounis, "Choke - a stateless active queue management scheme for approximating fair bandwidth allocation," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 2, 2000, pp. 942–951 vol.2.