Going Beyond DiffServ in IP Traffic Classification

Davide Aureli, Antonio Cianfrani, Alessio Diamanti, José Manuel Sanchez Vilchez, Stefano Secci

Going Beyond DiffServ in IP Traffic Classification

Davide Aureli*†, Antonio Cianfrani*, Alessio Diamanti†‡, José Manuel Sanchez Vilchez‡, Stefano Secci†,
*Universita' degli Studi di Roma "La Sapienza", 00185 Rome, Italy, Email: firstname.lastname@uniroma1.it
† Cnam, Paris, 75003 Paris, France. Email: firstname.lastname@cnam.fr
‡ Orange Labs , Orange, 92320 Châtillon, France. Email: firstname.lastname@orange.com

***Abstract*—Quality of Service (QoS) management in IP networks today relies on static configuration of classes of service definitions and related forwarding priorities. Packets are actually classified according to the DiffServ architecture based on the RFC 4594, typically thanks to static configuration or filters matching packet features, at network access equipment. In this paper, we propose a dynamic classification procedure, referred to as Learning-powered DiffServ (L-DiffServ), able to detect the distinctive characteristics of traffic and to dynamically assign service classes to IP packets. The idea is to apply semi-unsupervised Machine Learning techniques, such as Linear Discriminant Analysis (LDA) and K-Means, with a proper customization to take into account the issues related to packet-level analysis, i.e. unbalanced distribution of traffic among classes and selection of proper IP header related features. The performance evaluation highlights that L-DiffServ is able to change dynamically the classification outcome, providing an higher number of classes than DiffServ. This last result represents the first step toward a more granular differentiation of IP traffic.**

I. Introduction

Quality of Service (QoS) technologies commonly address the network link bottleneck issue by introducing protocols to support priority packets to pass first. Differentiated Services (DiffServ) is the de-facto QoS protocol used in IP networks, in most of Internet Service Provider (ISP) networks as well as in layer-3 operated data-center and entreprise networks. Once packets are classified into classes of services at a network access point or router, they can get dropped or delayed in the queuing system of core routers according to class priority. DiffServ was proposed as a stateless alternative to the stateful Integrated Services (IntServ) [1] architecture, suffering from scalability issues by following an end-to-end layer-4 flow virtual circuit resource reservation approach.

Indeed, IntServ revealed to be not well suited for heterogeneous systems as it requires support at the source terminal, destination terminal, all intermediate routers and at the application as well, which made it practically difficult to happen. In addition, IntServ-enabled routers to maintain an internal state for each virtual circuit opened by the resource reservation protocol [2], which makes them vulnerable to failures. Therefore, the community turned to DiffServ as a lighter and stateless approach, not touching at terminals nor applications, and not requiring all routers on the way to implement it. User traffic is mapped by edge routers or access points into the appropriate forwarding class, encoded into the packet header. This information is then used by the intermediate routers to differentiate packet processing, as forwarding classes indicate drop and resource priorities.

In this paper, we propose to go beyond the legacy DiffServ policy of manually setting QoS classes, in order to be able to dynamically learn the appearance of new classes worth being differentiated. Our proposal, called Learning-powered DiffServ (L-DiffServ), is to dynamically refine the set of classes in order to increase the granularity of the macro service classes. We propose a machine learning methodology for determining valuable sub-classes for actual packet classification. Our solution is composed of three main building blocks: i) data pre processing, performing features extraction and oversampling); ii) dimensionality reduction by means of the Linear Discriminant Analysis (LDA) procedure; and iii) clustering and classification, exploiting the K-Means algorithm. We run our experiments against real data from the MAWI dataset [3], containing daily IP traces of a transpacific backbone link.

The paper is structured as follows. Section II gives the necessary background. Section III presents our machine learning methodology. After in Section IV we report the numerical result. Finally, Section V concludes the paper.

II. Background

We provide the necessary background on DiffServ and application of machine learning in traffic classification.

A. DiffServ Architecture

The DiffServ architecture is described in RFC 2474 [4]. It relies on a mechanism to classify and mark packets as belonging to a specific class of service, using 8 bits for the Differentiated Services (DS) field in the IP header. This field is composed of two parts, the 6 most significant bits identify the DSCP (Differentiated Services Code Point) while the 2 least significant bits define the ECN (Explicit Congestion Notification). Classification is based only on the DSCP field, while the ECN field is used for router communication in congestion detection. Each router is configured to differentiate traffic based on its set of classes. Complex functions, such as packet classification, can be carried out at the edge of the network by access routers, whereas core routers simply apply per-hop behaviour (PHB) treatment that defines the packet forwarding properties associated to a specific traffic class. In theory, a network could have up to 64 different traffic classes using the 64 (2^6) available DSCP values, which can give to

the network operator great flexibility. In practice, DiffServ recommend the following PHB encoding:

- **Default Forwarding** (DF): PHB applied to any traffic that does not meet the requirements of any other service classes. Typically, DF has best-effort forwarding characteristics. The recommended DSCP for DF is 0 [5].
- **Expedited Forwarding** (EF): PHB with characteristics of low delay, low loss and low jitter. These characteristics are suitable for voice, video and other real-time services. EF traffic is often given strict priority queuing above all other classes. The recommended DSCP for EF is 46 [6].
- **Assured Forwarding** (AF): PHB meant to provide assurance of delivery as long as the traffic does not exceed some prefixed rate. Traffic that exceeds the rate faces a higher probability of being dropped if congestion occurs. The AF is composed of four separated classes and, for each each class, three drop precedence levels (high, medium or low) are present. Twelve separated DSCP encodings from AF11 to AF43 are thus defined [7].
- **Class Selector** (CS): it assures backward compatibility with network devices that still use the IP Precedence field. Before the DiffServ architecture, IPv4 networks could use the IP precedence field in the ToS [8] byte of the IPv4 header to mark priority traffic.

B. Machine Learning applied to Traffic Classification

Let us mention relevant works in the literature that apply machine learning to IP traffic classification, anticipating similarities and differences with our approach.

In [9], the average packet size and average duration of the flows are used to classify flows with K-means; our proposal instead works at packet-level and target DSCP marking, ensuring a fine-grained classification. The work was carried out on a dataset created collecting full packet traces from a monitor attached to their campus Internet link. In the analysis on the clustering applications they considered only TCP traffic and not UDP one, whereas we work with both of them detecting the different class of service to which they are related. Their results for the choice of cluster number (k) is interesting, increasing k led to better performance in the evaluation metrics (Precision and Recall) pointing out that flows of the same application level showed differences and therefore the presence of sub-groups within the applications. This idea inspired our way to establish classes.

In [10] K-means, DBSCAN (unsupervised clustering) and AutoClass (probabilistic model-based clustering) are applied to two empirical packet traces. One is the Auckland IV [12], the other is a full packet trace that they collected at the University of Calgary. They compared the three methods in terms of accuracy, showing that both K-Means and DBSCAN work very well and much more quickly than AutoClass. In our work, we consider only K-Means because DBSCAN, even if it obtains excellent performances, is too bound to the parameter choice (ϵ and min_{pts}) parameters: they should be redefined for different contexts.

In [11] authors proposed an automated Class of Service (CoS) mapping. They classified the traffic into four different classes, i.e. Bulk Data, Interactive, Multimedia and Transactional. They worked on 4 sets of data: Auckland-IV [12], and three different traces from Gigascope [13]. They proposed to extract an identifying vector, composed of statistic flows features, for the 4 service classes and then test the behaviour of two classification algorithms, K-nearest neighbors (KNN) and LDA. In our work LDA is used to reduce the dimensionality and to detect the variables that maximize the variance between the classes of our interest ([5]) while for the classification we use the minimum distance with respect to the centroids extracted from the K-Means, to obtain a real-time infrastructure.

[14] rather than making a priori assumption for the number of service classes, they apply K-Means in unsupervised way for the data exploration part. Here they detect three clusters: Transactional application (DNS), Interactive application (Telnet and FTP) and Bulk Transfer application (HTTP). The reference feature vectors are the mean packet length per connection, connection duration and variance of packet interarrival. For classification they applied KNN evaluating the result according to the Davies-Bouldin Index, which is not normalized. This is the main reason to use the Silhouette Coefficient to evaluate the clustering outcome, making possible to compare results related to different daily-traces.

III. Machine Learning Methodology

This section details our Learning-powered DiffServ (L-DiffServ) solution for dynamic refinement of DiffServ classes of services, including dataset processing, oversampling, dimensionality reduction, clustering and classification steps.

A. The L-DiffServ architecture

Our proposal consists in generating a new set of service classes through a hybrid semi-unsupervised machine learning technique that automatically assesses the new number of service classes, identifying the sub-classes within existing preset classes, based on features extracted from IP packet flows. Figure 1 depicts the L-DiffServ workflow.

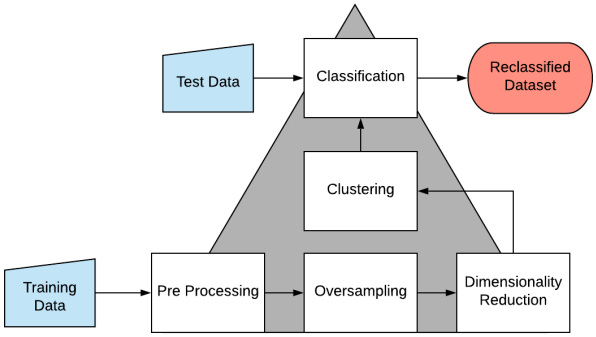

Fig. 1: L-DiffServ Workflow Overview.

Data preparation is accomplished through Pre-processing, Oversampling and Dimensionality Reduction. In the pre-processing phase the numeric values from captured packet

features are normalized, while the categorical ones are transformed into binary; at the end of these operations, we delete the features with zero variance because they have not information for the differentiation between classes. In the oversampling phase, we generate artificial samples of the minoritary class (the one with less occurrences) to balance the data set. It is worth noting that we choose to oversample the information of the less numerous classes rather than undersampling the best effort class because undersampling could lead to significant information losses [15]. Finally, we reduce the space dimensionality maximizing the variance between service classes and projecting the starting dataset into the new dimensional space.

The goal of the Clustering step is to produce a new set of classes, so that the proposed classification is based on a grouping done according to the clustering method outcome. First, we evaluate the clusters obtained through the Silhouette Coefficient index of goodness [16]; it measures the magnitude between cohesion (intra-cluster distance) and dispersion (inter-cluster distance) for each group. Once we obtain the maximum Silhouette value we establish the optimal number of clusters. In this way we can assign a new label for each observation and train the model based on our classification. The end of the Clustering step coincides with the beginning of Classication one. We can reclassify a test trace, with the same structure, applying the transformations used in our methodology, evaluating the minimum euclidean distance between the packet and centroids extracted from the Clustering step.

We validate this methodology on a public dataset provided by MawiLab [3] described hereafter.

B. Dataset and Features

The WIDE project publishes daily IP traces of a transpacific link, called the MAWI Archive [3]. Each file contains 15 minutes of traffic flows, captured between 14:00:00 and 14:15:00 local time. This represents usually between 10 and 20 GB of traffic for one file. Before being released, traces are anonymized to hide any personal information (removing application data and scramble IP addresses with the Crypto-PAn Algorithm [17] following collision-free and prefix-preserving principles). As of our knowledge, there is no other equivalent public dataset, spanning many months, we could exploit.

In our analysis, we work on traces of multiple days across multiple weeks for training and classification. We consider the trace belonging to the period which goes from 3rd April 2019 to 10th May 2019, considering only Wednesday. The average size of the traces is 19240.36 MB and the average number of packets is 252,046,154. In the traffic analysis, we work only with IPv4 packets along one forwarding direction by filtering packets through the MAC address. We consider for each trace 3,000,000 packets of the total trace because of memory limits; we make a random sample splitting the trace with 200,000 packets maintaining the percentage of packets for each service classes. We focus the attention on a specific day in this section (April 3, 2019).

The following characteristics are the features extracted from every packet header: Internet Header Length (IHL), Differentiated Services Code Point (DSCP), Explicit Congestion Notification (ECN), Total Length, Flags, Fragment Offset, Time To Live (TTL), Protocol, Source address, Destination address, and from the TCP header part we extract Source Port and Destination Port.

DSCP Value	DSCP Class	Class Label
48, 56	CS6, CS7	Network and Internetwork Control
40, 46	CS5, EF	Critical RTP Voice
32, 34, 36, 38	CS4, AF4	Flash Override
24, 26, 28, 30	CS3, AF3	Flash Voice
16, 18, 20, 22	CS2, AF2	Immediate
8, 10, 12, 14	CS1, AF1	Priority
0	CS0	Best Effort

TABLE I: Mapping between DSCP and class labels, from: [5]

In Table I we report the mapping between the DSCP value and the service class name we adopt during the analysis as packet labels. Observing the column *Class Label*, clearly we unify the CS class of service both with the AF and with EF because there are few observation for the backward compatibility classes.

In the following we show each step of our methodology.

C. Pre Processing

We process the dataset composed of the packet features with the DSCP marking as label. We do not to consider the features with zero variance because not important for packets differentiation. Such features revealed to be the Internet Header Length (IHL), the Flags except for the DF (Do-not-Fragment) flag, and the Fragment Offset.

For the four categorical variables, i.e., Source and Destination Address and Source and Destination Port, [17] uses a different key for each day to anonymize the trace, so we cannot include IP addresses in our model; instead, we can handle the Source and Destination Port - our idea is to determine an identification port for each packet, trying to cover a high density of information. We cover the ports that allow to keep 90% of packet volume, while the remaining ports with a small occurrence are transformed into an artificial port of number 0. Moreover for the packets whose protocol has no defined port, as ICMP packets, we assign the port number -1; the values -1 and 0 have not a numerical importance but they affect the presence or the absence of such port within the packet. In this way we create the summary variable called *Heavy Port*. Finally, in the data Pre-Processing we transform the categorical variables into binary variables applying *One-Hot Encoding* [18] (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) while the numerical variables are normalized. For data normalization we use the *MinMaxScaler* function, with the following formula:

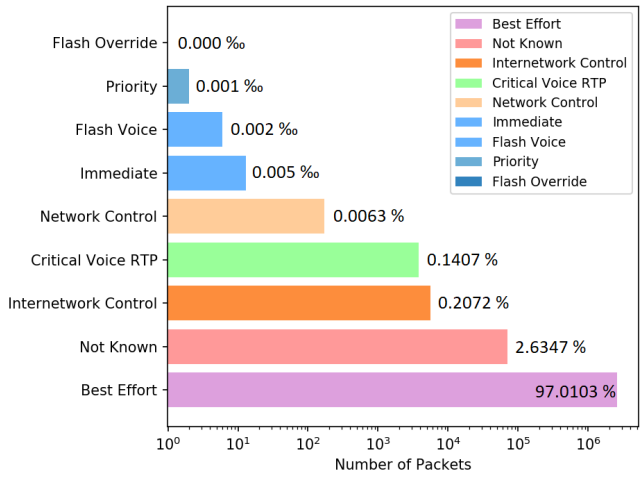

Fig. 2: DSCP Distribution in the considered MAWI dataset.

$$\frac{x_i - min(x)}{max(x) - min(x)} \tag{1}$$

However, until now, our methodology considers all packets in an undifferentiated way regarding the DSCP marking. Nevertheless, observing the service class occurrences there is a great imbalance in favour of the 0 DSCP label (best-effort class). Thus, we apply the analysis for the Source and Destination Port and the normalization part splitting the beginning data frame into two components: the first one with only the best-effort data, while in the second one we consider everything not marked as 0 for DSCP value. In this way we maintain the information related to best-effort and non best-effort classes, otherwise the few observations of the non best-effort classes could disappear under the magnitude of the best-effort packets. This last consideration opens the door to the main challenge for our analysis, the data unbalance.

D. Oversampling

Let us comment on the the DSCP distribution from our trace, focusing on the percentage of packets for each service class in Figure 2. More than 97% of the traffic is best-effort traffic, the other services covering the remaining 3%. We report in the histogram the label *Not Known*, not listed in the Table I; it corresponds to DSCP values not recommended by RFC 4594 [5], most of them being values between 0 and 8. This traffic is known as Scavenger, i.e., traffic with lower priority than the best-effort class and to which is allocated the lowest amount of bandwidth.

Furthermore it is interesting to observe the percentage of packets related to the other service classes, to have a complete picture of the available information. The Scavenger class is the second service with the greatest usage 2.63%. The Expedited Forwarding (CS5, EF) class and Network & Internetwork Control (CS6 and CS7) have a good percentage of occurrences: the first with 0.14% and the second with 0.21%. However, the four Assured Forwarding (AF) classes have almost no packets. In fact the packets related to the Priority (CS1, AF1), Immediate (CS2, AF2), Flash Voice (CS3, AF3) and Flash Override (CS4, AF4) are only the 0.0008%.

Such unbalanced data distribution represents one of the most discussed problems in the Machine Learning literature, as it strongly influences the behaviour of classification algorithms in favour of a specific class. This problem is known as the *Paradox of Accuracy* [19]. In fact, if we limit ourselves to identify best-effort packets from non best-effort, any classification algorithm trained on this data will classify everything as best-effort getting always more than 90% of Accuracy.

To countermeasure this issue, we exploit the Smote over-sampling technique presented in [20]. The Smote algorithm can oversample the service classes for which we have very few samples, without losing information from the best-effort class. In this way we obtain a balanced dataset, where each service class has the same number of occurrences of the best-effort class, hence we can extract the characteristics which maximize the differentiation between service classes.

E. Dimensionality Reduction

The dataset matrix after the over-sampling procedure has $10,611,286$ rows, representing the packets, and 42 columns, as number of features (including the DSCP label). Our purpose is to reduce the features only to the ones that allow to identify the service class each packet belongs to. We reduce the dimensionality of the space working with the LDA (Linear Discriminant Analysis) technique [21]: it allows to specify the number of axis for the new space, thus obtaining a 3D graphic distribution of the packets. Moreover it works in a supervised way, being also a classification algorithm; in this way it can maximize the variance between classes of services; in fact, we set it to use a variance of 97.8%. In Table II, for each one of the new 3 LDA axis, we report the rate of correlation between the original features and the corresponding axis; for each axis, we report only the top ten components.

Variable	Value	Variable	Value	Variable	Value
Port 123	30.6%	Port 123	27.1%	Protocol 47	12.5%
Protocol 17	20.3%	Protocol 47	10.6%	Protocol 97	6.1%
Port 443	6.5%	Protocol 97	4.6%	Protocol 4	5.9%
Protocol 47	2.6%	Protocol 4	3.2%	Protocol 17	5.2%
Protocol 97	1.9%	Protocol 6	3.1%	Protocol 6	4.9%
Protocol 6	1.8%	Protocol 17	3.0%	Port 22	4.9%
Protocol 89	1.7%	Port 22	2.8%	Port 53855	3.7%
Port 22	1.7%	Port 53855	2.8%	Port 443	3.2%
Protocol 80	1.6%	Port 53469	2.8%	Port 89	2.9%
Port 8080	1.6%	Port 8080	2.7%	Port 8080	2.9%

TABLE II: Linear Discriminant Components

F. Clustering

At this point, the dataset space is projected into the new dimensional space defined by LDA, and the classification step begins. We have to cluster packets even if they have already an assigned label. Our purpose is to increase the granularity of the current service classes. The starting point is the recommended

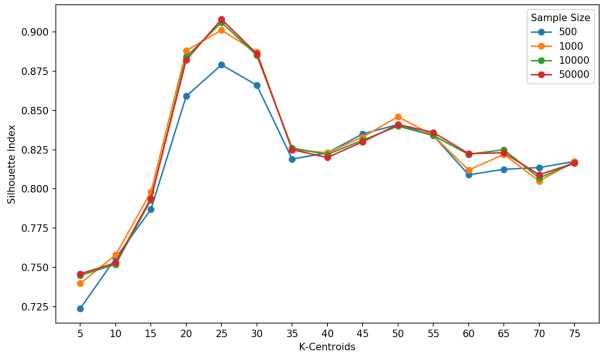

Fig. 3: Silhouette as a function of the number of centroids.

classification in RFC 4594 [5], which represents the macro-classification currently used for the DSCP marking:

- Best-Effort (BE);
- Not Known;
- Assured Forwarding (AF);
- Expedited Forwarding (EF);
- Network & Internetwork Control.

The problem we are going to face is a clustering problem, where data do not have a label and we cannot observe the correctness of the results obtained. However, it is essential to analyze the results according to an index of goodness about the clustering. Therefore, the fundamental tools are the clustering algorithm and the measure to evaluate the results. These tools lead us to determine the optimal number of centroids (k), which are the new available service classes. For the clustering algorithm we work with the well-known K-Means [22], while for the evaluation index we use the Silhouette Coefficient [16].

The Silhouette index is a measure of how similar a cluster is to its own cluster (*Cohesion*) compared to other ones (*Separation*). The Silhouette ranges from -1 to $+1$, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighbouring clusters. The formula and notations used to compute the Silhouette index are as follows:

$$S(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \quad if \quad |C(i)| > 1 \tag{2}$$

- $S(i)$: Silhouette index for cluster $C(i)$;
- $a(i)$: Cohesion;
- $b(i)$: Separation;
- $|C(i)|$: Number of elements in $C(i)$.

We evaluate the optimal number of clusters (k) to establish our final proposal for the service classes. The analysis considers a range of possible values for k, from 5 to 75 with a step of 5. Figure 3 shows the results of the Silhouette Coefficient.

In each k we compute (2) 100 times for each possible sample size; in this way we capture the real behaviour of our packets population without considering all the packets. In Figure 3 we show the trend according to the variation of the sample size. At the beginning we have a dramatic increase for

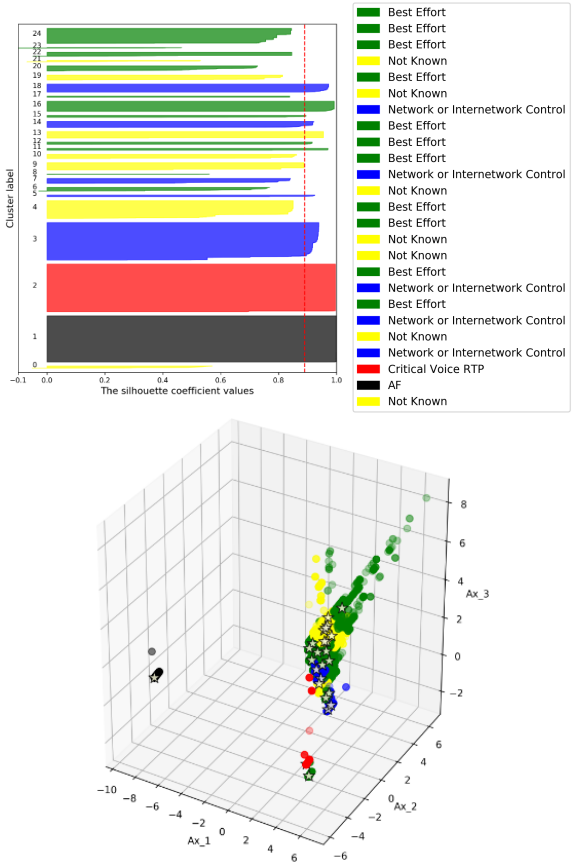

Fig. 4: Silhouette Coefficient & 3D K-Means.

the Silhouette Index passing from 5 to 25 centroids. Then there is a decrease until 35 centroids and then, for the remaining part of the line chart, we have a steady behaviour around 0.825. So we select 25, the peak of our analysis, as the new number of service classes. It is interesting to check the association among subclasses and macro classes. In the Figure 4 we show the result of K-Means with 25 centroids; analyzing each cluster according to the Silhouette index. In the upper plot, on the left side, the dotted red line identifies the average value between all the clusters for the Silhouette Coefficient, equal to 0.908. In the right side we have the legend related to the 3D-plot about K-Means clustering. In this legend we can see the sub-classes identified within the main classes. The best-effort class is differentiated into 11 sub-categories, while the Not Known service is divided into 7 different sub-classes. For both Assured Forwarding (AF) and Critical Voice RTP (EF) we see that our clustering algorithm finds only one subclass, without sub-divisions.

G. Classification

For classification we use the K-means algorithm, exploiting the centroids built during the clustering analysis: in this way we can obtain the reclassification in real time. The low complexity of this step is proven by the following result: the time to reclassify 1 million packets is about 0.51 seconds. In this way, we can obtain any part of the trace used as a

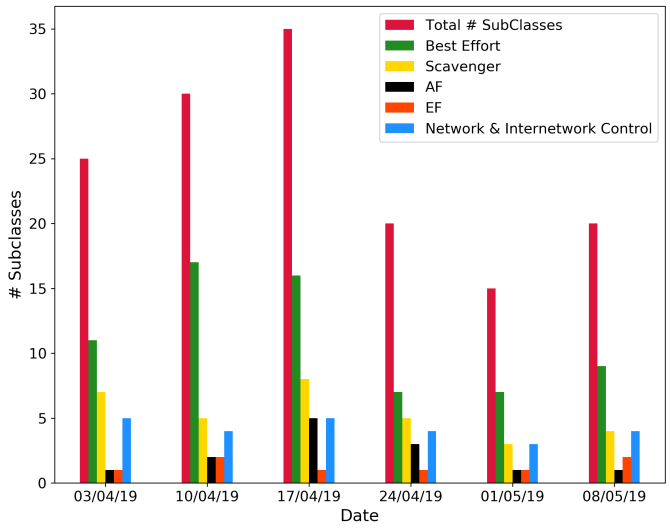

Fig. 5: L-Diffserv Dynamical Behaviour.

test reclassified according to our new distribution in service classes.

IV. Numerical Analysis

We can summarize the results in terms of generated subclasses fragmentation, for the whole period considered (Apr. 3 to May 8, 2019, one day per week), through the bar chart in Figure 5. For each of the given 5 classes, we report the number of subclasses generated - the total number of subclasses is given by the first column for each date. It is relevant to observe that the best effort service class is always the most differentiated one in subclasses, followed by the Scavenger class. The other service classes, especially AF and EF, show a low fragmentation; traffic from these classes is therefore not showing specific behaviors within the class, likely because existing features are already used in general to distinguish them (e.g., Port number or Protocol) from the other classes.

An important aspect to highlight is that the number of subclasses generated for the best effort and Scavenger classes have a high variability in time - e.g., the number of best effort subclasses in Apr. 4 is twice the same number in Apr. 24, 2019. This shows that L-DiffServ is able to capture the dynamicity in traffic patterns and to adapt traffic aggregation in subclasses.

The capacity to aggregate traffic flows in subclasses can be beneficial in bottleneck management; large amount of traffic (of best effort and Scavenger classes) that by default would all be assigned to two priority levels, can instead by discriminated by L-DiffServ in up to 23 priority levels, while being reassured that subclasses are ordered with respect to each other based on L-DiffServ ordering. Hence traffic loss can be concentrated to few flows of few subclasses instead of a high number of flows spanning a large number of subclasses.

V. Conclusions

This work aims to provide a methodology, named L-DiffServ, for reaching dynamic class of service generation in IP networks. In fact, L-Diffserv differentiates further the traffic based on an existing high-grain classification, giving us advices to improve the resource allocation (buffer and bandwidth) according to this new service classification. As further work, we want to test the behavior of L-DiffServ in real-time systems, increasing the amount of packets analyzed. Finally we want to evaluate the behaviour of the network under congestion according to the different service classification methods, describing advantages and disadvantages of our proposal and the current DiffServ method. We do also envision extending the methodology for service-level agreement management in network slicing.

Acknowledgements

This work was partially supported by the ANR CANCAN project (ANR-18-CE25-0011).

References

[1] D. Clark, R. Braden, S. Shenker, "Integrated Services in the Internet Architecture: An Overview", *RFC 1633*, 1994.
[2] L. Zhang et al., "Resource ReSerVation Protocol (RSVP)Version 1 Functional Specification", *RFC 2205*. 1997.
[3] C.S.L. Sony, K. Cho, "Traffic data repository at the WIDE project", *In Proceedings of USENIX 2000 Annual Technical Conference: FREENIX Track*, pp. 263-270, 2000.
[4] K. Nichols et al. "Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers", *RFC 2474*, 1998.
[5] J. Babiarz, K.H. Chan, F. Baker, "Configuration guidelines for DiffServ service classes", *RFC 4594*, 2006.
[6] B. Davie et al., "An Expedited Forwarding PHB (Per-Hop Behavior)", *RFC 3246*, 2002.
[7] J. Heinanen et al., "Assured forwarding phb group", *RFC 2597*, 1999.
[8] D. Grossman, "New Terminology and Clarifications for Diffserv", *RFC 3260*, 2002.
[9] J. Erman et al., "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*, pp. 883-892, 2007.
[10] J. Erman, M. Arlitt, A. Mahanti, "Traffic classification using clustering algorithms", *In Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pp. 281-286, 2006.
[11] M. Roughan et al., "Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification", *In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 135-148, 2004.
[12] J. Micheel, I. Graham, N. Brownlee, "The Auckland data set: an access link observed", *In Proceedings of the 14th ITC specialists seminar on access networks and systems*, pp. 19-30, 2001.
[13] C. Cranor et al., "Gigascope: a stream database for network applications", *In Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp. 647-651, 2003.
[14] T. M. Chen, Y. Zeng, "Classification of traffic flows into QoS classes by unsupervised learning and KNN clustering", KSII Trans. on Internet and Information Systems, 2009, 3.2: 134-146.
[15] A. Fernndez et al., "SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary", *Journal of artificial intelligence research*, 2018.
[16] P.J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis", *Journal of computational and applied mathematics*, 1987.
[17] J. Fan et al., Prex-Preserving IP Address Anonymization, *In Computer Networks 46*, pp. 253272, 2004.
[18] S. Raschka, "Python machine learning", *Packt Publishing Ltd*, 2015.
[19] T. Afonja, "Accuracy Paradox", *Towards Data Science*, 2017.
[20] N.V. Chawla et al., "SMOTE: synthetic minority over-sampling technique", *Journal of artificial intelligence research 16*, pp. 321-357, 2002.
[21] A. Tharwat et al., "Linear discriminant analysis: A detailed tutorial", *AI communications 30*, No. 2, pp.169-190, 2017.
[22] D. Arthur, S. Vassilvitskii, "k-means++: The advantages of careful seeding", *In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027-1035, 2007.