Agglomerative Clustering of Network Traffic Based on Various Approaches to Determining the Distance Matrix

Vladimir Deart
MTUCI
Moscow, Russia
vdeart@mail.ru

Vladimir Mankov
Training Center Nokia
Moscow, Russia
vladimir.mankov@gmail.com

Irina Krasnova
MTUCI
Moscow, Russia
irina_krasnova-angel@mail.ru

***Abstract*—We are presenting a real-time traffic flow classification model for maintaining QoS in dynamic networks such as Software Defined Networks (SDN). In previous works, we managed to achieve high accuracy (90-95%) on the database of flows known for the model using Machine Learning (Supervised Learning) methods but in a dynamic SDN new network applications and flows appear more often than usual. For detection of new flows it is proposed to use the Agglomerative clustering method, which has never been used to solve the problem of network flow classification, because early approaches to traffic clustering gave insufficient results and the speed of its operation was too low. This paper offers a combination of different Machine Learning methods in such a way that Agglomerative clustering is responsible only for updating the class database, and Supervised Learning methods are responsible for quickly classifying known flows, which solves the problem of model speed. Clustering accuracy is improved by automatically controlling the cluster construction process by determining the distances between flows using the Random Forest and Extra Trees methods. In the experimental part of the study, three more most promising ways of determining distances are given for comparison: Random Trees Embedding, Euclidean and Manhattan distance. Results of clustering of TCP and UDP applications for different number of clusters and different size of the initial sample are presented. Experimental studies confirm the effectiveness of using hierarchical clustering in traffic clustering tasks under the condition of controlled cluster construction.**

I. Introduction

Modern telecommunications networks transmit a large amount of heterogeneous traffic and for each class of traffic it is necessary to provide the corresponding requirements for Quality of Service (QoS). The classification can be based on applications, services, traffic types (voice, video, data) or other characteristics. Early methods of automatic traffic classification were based on defining applications by port, but with the advent of dynamically changing ports, this approach began to lead to a large number of errors. In this regard, the analysis of DPI packets (Deep Packet Inspection) was proposed. For this method, you must have access to the application part of the packet but since networks often use encryption, this approach does not always allow you to identify the application. In addition in dynamically changing networks such as SDN (Software Defined Networking) new applications are sometimes added that are unfamiliar to the DPI tool. Recently, to solve the problem of traffic classification, Machine Learning (ML) methods are increasingly used, which allow not only to effectively classify traffic and expand the database of known flows, but also to analyze flows based only on the statistical properties of flows.

Machine learning methods are divided into Supervised Learning and Unsupervised Learning methods. In Supervised Learning, each sample element has its own feature vector and the label of the class it belongs to. The class label for flows must be known in advance or determined using automatic tools (nDPI, L7 filter, etc.), which introduces some problems in the classification process, such as automatic markup errors and the inability of the model to determine new classes that are not represented in the training sample. For Unsupervised Learning (unsupervised learning or clustering), a class label is not required, and class divisions are based only on flow characteristics. This approach allows you to introduce new previously unexplored classes but is less accurate than Supervised Learning.

We are conducting a series of studies aimed at developing a classifier of traffic flows using ML methods to maintain QoS in a dynamic SDN network in real time [1], [2]. Due to the specifics of the problem, the classifier must have the following distinctive properties:

- Classification by a limited set of features based on the properties of the first N packets, since it is not possible to get information about the entire flow.
- Detection of new classes, i.e. clustering methods should be applied.
- High accuracy of the results obtained.
- Ability to interpret classes for further link them to QoS policies.
- The division into clusters is based on a dynamically changing metric that is not predefined in advance, such as the distance between clusters.

This paper is part of our research work and focuses on clustering traffic flows. When clustering individual elements of the original sample (in our case, flows) are represented on

coordinate axes in N-dimensional space and grouped together in clusters depending on the distance between them. One of the tasks that arise in this case is to determine the method of calculating distance between clusters. In this paper, we consider the five most promising approaches to calculating the distance between clusters, and then conduct experimental studies using these methods to cluster traffic flows from the real network and give a comparative assessment of their performance. Despite the fact that the work is focused on the tasks of the SDN network, the results of the work can be used in other networks as well.

The paper is structured as follows: section I presents the background and goals of the research, section II reviews the most outstanding works of other researchers, and section III provides brief theoretical information about the methods used. Section IV presents the algorithm of our model and the relationship between the methods used in it. Section V presents the results of experimental studies for real network traffic, and section VI presents conclusions and suggestions for future work.

II. Related Work

The article [3] provides an overview of some promising works and approaches to the problems of traffic classification using Supervised Learning methods, and in [4]- Unsupervised Learning.

In the paper [5] instead of the Euclidean distance, a distance matrix based on a Random forest is used, followed by clustering using the K-Medoids method. The study showed a significant improvement in distance-based clustering results using a Random forest compared to Euclidean distance. However, it is worth noting that to perform clustering using the K-Medoids method, you need to have data on the number of clusters to split, which is not possible in a dynamically changing network in real time.

The authors [6] suggested using hierarchical clustering as a way to extract and generalize statistical information about traffic activity in high-capacity networks. In [7] hierarchical clustering methods (AutoFocus and BIRCH) based on IP addresses and other categorical heuristics allowed us to create some basic traffic patterns. These and other works confirm the effectiveness of using hierarchical methods in network traffic analysis tasks.

In this paper, we suggest using Agglomerative clustering to classify an application in real time, since it supports clustering based on the distance between classes and thanks to its structure, it is possible to interpret the resulting clusters in terms of QoS policy (unlike many other methods, e.g. DBSCAN). To compare the accuracy of the results, we consider controlled and uncontrolled ways to obtain a distance matrix for building a model.

III. Background

A. Agglomerative clustering

Agglomerative clustering [8] is one of the types of hierarchical structures and can be represented as a dendrogram that allows you to organize data "from bottom to top": initially, each element of the sample is considered a separate cluster, as the distance between clusters decreases, objects are combined to form new clusters (Fig. 1).

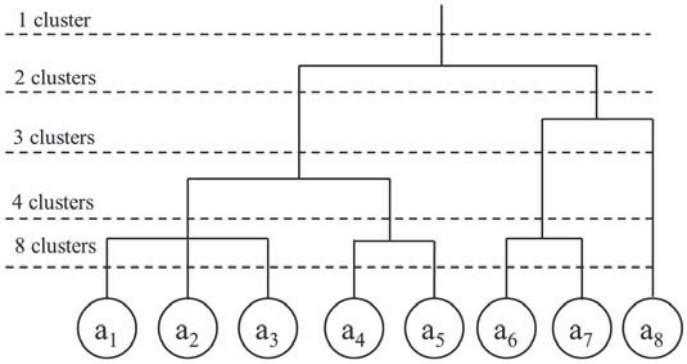

Fig. 1. An example of a dendrogram of the eight samples

To perform clustering, you need to calculate the pairwise distance between all elements and set a measure of the distance between clusters. The matrix of pairwise distances can be defined in different ways, which are conditionally divided into two groups:

1) Unsupervised distance calculation: Euclidean distance, Manhattan distance, Random Trees Embedding. With this method of calculating distances, class labels are not used and distances are calculated only based on the feature matrix.
2) Controlled distance calculation: Random Forest, Extremely Randomized Trees. With this method, the sample is divided into supervised learning classes, and then the distance between the sample elements is calculated.

The paper analyzes standard approaches to determining distances (Euclidean distance, Manhattan distance), since they are effective in many Machine Learning problems and methods for constructing a distance matrix based on Decision Trees, since such algorithms show better results in traffic classification problems compared to others.

To determine the measure of distances between clusters, the **complete linkage** method was used, since it is recommended when using non-standard approaches to determining distances. In this approach, the distance between two clusters is the maximum distance between two elements from different clusters.

Agglomerative clustering allows you to adjust the number of clusters depending on the distance between clusters. Thus, unlike other common methods (K-means, spectral clustering, etc.), you do not need to have knowledge about the number of clusters before clustering. This feature makes it possible not only to determine the optimal number of clusters for the training sample but more importantly, it is possible to introduce new classes into an existing model automatically, adjusting only the distance between clusters.

B. Machine Learning algorithms used to calculate the distance matrix

Decision Tree, DT is one of the basic machine learning algorithms. It is a logical structure consisting of leaves and nodes. A leaf is the terminal end of the tree and uniquely determines whether the selection element belongs to a class. At each node, a decision is made about which next node or leaf to go to in order to classify the current item. To make decisions, a conditional function is set in the node based on some feature of the object. Binary trees are most widely used, i.e. such that the conditional function divides objects into two child nodes and / or leaves.

By itself DT is weak and unstable algorithms and copes well only with simple tasks. However, it is often used in ensemble algorithms, where it helps to achieve high accuracy by working in conjunction with other trees or other models. The paper uses Random Trees Embedding, Random Forest and Extra Trees (Extremely Randomized Trees) as such ensembles.

The **Random Forest (RF)** and **Extremely Randomized Trees (ET)** algorithms are a set of DTs that perform classification in parallel and independently of each other. The final result of the prediction is chosen by a majority vote. The main difference between RF and ET is the significantly greater influence of the randomness factor when using ET. RF performs sample splitting at nodes in the best way, while ET does it randomly. Also, in RF, unlike ET, data is initially loaded using the bootstrap method, which allows you to create many others based on the existing selection and use them to replace elements.

RF shows a more compact distribution of sample objects and the model most often requires fewer trees. In most cases, RF gives better classification results compared to ET. Despite this, the ET construction results in a larger number of leaves, i.e. the data is more sparse, which does not give an advantage in classification, but affects the calculation of distances between elements, which can be used in subsequent clustering.

An ensemble of **Random Trees Embedding (RTE)** is a construction of a forest based on an unsupervised sample, i.e. it does not require the presence of training sequence class labels.

C. Evaluation of clustering algorithm

The **Adjusted Rand index (ARI)** and the **Adjusted Mutual Information (AMI)** measure the similarity of two assignments (true and predicted), ignore permutations, and normalize all values within [-1; 1], where -1 is bad labeling, 0 is random labels, and 1 is the best match indicator. AMI is based on a measure of Shannon's information, and ARI is based on comparisons between sample elements.

Homogeneity indicates that each cluster contains objects belonging to only one class. By **completeness** you can determine the proportion of a sample of one class belonging to one cluster.**V-measure** is the harmonic mean between **Completeness** and **Homogeneity**. All three characteristics are normalized within [0; 1], where 1 corresponds to the best result.

IV. METHODOLOGY

A. Traffic classification model

Our traffic classification model is shown in Fig. 2. In the

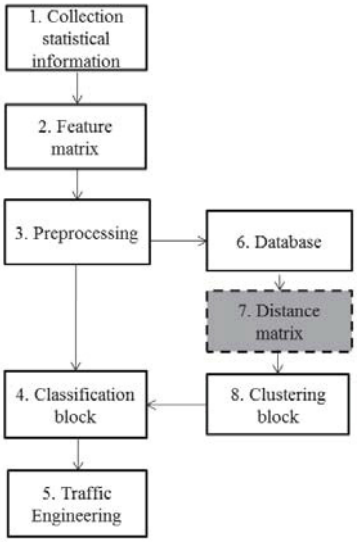

Fig. 2. Traffic classification model

data collection block (1), the time of arrival and its size are recorded for a packet entering the network. After that, packets are grouped into flows based on 5-tuple (IP addresses, port source and destination and protocol). In order for the system to work in real time, only the first 10-15 packets of the flow are taken into account, and the classification results are applied to subsequent packets [9].

Next, the feature matrix (2) is calculated for each flow. The features are statistical properties of the flow: the length of each packet, the interarrival time, and the characteristics calculated based on them (median, variance, STD, minimum and maximum values, bit and packet rates) [10].

The preprocessing (3) block works with outliers, normalizes and scales the resulting feature matrix.

In the next classification block (4), Random Forest (or other advanced Supervised Learning methods) is used to classify traffic according to the classes known for the model [11]. If the application is new and unfamiliar to the model, some errors may occur at this stage, because the Supervised Learning methods can only classify samples by classes that are known to them. This assumption is made consciously in the model, since further work on traffic management (5) is based on known classes. However, the classification block (4) represents the closest class for the new flow that has arrived, and as a result, the corresponding Traffic Engineering (TE) rules apply to it.

While the classification block (4) decides which class to assign the active flow to, and the TE block applies real-time traffic management rules, a copy of the new flow information from block (3) is sent to block (6). this is where the database for all known flows is stored. Each sample is represented by its own coordinates in N-dimensional space in accordance with certain features.

The subject of this article is block (7), which calculates the distances between the available samples. For research purposes, we consider five ways to construct a distance matrix: two of them are standard and generally accepted methods (**Euclidean distance** and **Manhattan distance**), and the other three are based on methods of precomputed distances (**Random Forest, Extremely Randomized Trees** and **Random Trees Embedding**).

After constructing the distance matrix between samples, the clustering block (8) divides the samples into clusters using Agglomerative clustering. This approach makes it possible not only to identify new clusters, but also to determine their nearest neighbors. Information about the nearest neighbors can be used in the TE(5) block.

After analyzing a certain number of new flows and creating additional clusters, the classification block (4) learns new classes, and the TE tools (6) define new rules for managing flows. This approach for building a classifier model allows you to use the speed of classification using Supervised Learning methods and the ability to identify new flows using Unsupervised Learning methods.

B. Distance matrix

The pairwise distance matrix specifies the relative distances between all pairs of its elements. For its construction, Euclidean or Manhattan distance is often used, calculated on the basis of a feature matrix.

Euclidean distance (1)

$$d(p,q) = \sqrt{\sum_{i=1}^{m}(p_i - q_i)^2} \tag{1}$$

where p and q are a pair of feature vectors of two flows, m is number of features.

Manhattan distance (2)

$$d_1(p,q) = \sum_{i=1}^{m}|p_i - q_i| \tag{2}$$

In addition to the standard methods for calculating the distance matrix, the distance matrix obtained using any other methods, the so-called precompiled distance matrix, can also be used. For this purpose classes are placed using the Supervised Learning methods, and the distances between clusters are extracted based on the results obtained. In many situations, this approach allows more efficient clustering than standard methods.

The process of creating a distance matrix based on decision trees (**ET, RF, RTE**) (Fig. 3) can be represented as the following sequence of actions:

1) Calculation of the $n \times m$ feature matrix based on the statistical characteristics of the flow, where n is the number of flows, and m is the number of features.
2) construction of a forest of k trees based on the matrix of features of the training sequence for ET and RF takes place in a controlled mode.
3) Assigning each flow the index of the leave on which it appeared on each of the trees. This step results in an $n \times k$ leaves matrix.
4) Creating a matrix of pairwise proximity $n \times n$, where between each pair of flows, the total number of leaves on which they appeared together among all trees, is indicated. The leaves comparison process is performed using One-Hot Encoding (representing each state using a single trigger) and the product of the encoded index matrix and its transposed form.
5) Getting the distance matrix by normalizing the proximity matrix relative to the maximum value and subtracting its values from one.

Thus, the result of this sequence of actions will be a matrix of pairwise distances $n \times n$, where its elements are distributed within [0;1] and 1 is the maximum distance between flows, and the diagonal consists of 0.

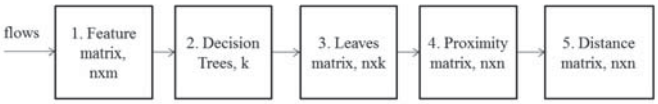

Fig. 3. Building a distance matrix based on Decision Trees

V. Experimental Setup

A. Internet traffic dataset

For the experimental part of the study, we used 15-minute traffic routes collected in January-February 2020 by the MAWI research group as part of the WIDE project on a real network section [12]. Traffic was divided into unidirectional flows by 5-tuple fields (source and destination IP addresses and ports, TCP/UDP Protocol). Using the nDPI tool, it was possible to perform the initial markup of traffic flow classes, which were represented by application names. TCP and UDP traffic were studied separately from each other. The TCP database consisted of 1500 flows for each of the applications: DNS, Apple (protocol for accessing Apple iCloud), IMAPS, HTTP, SSH, Telnet. The UDP traffic database contains 250 flows for each application: DNS, NTP, Quic, SNMP, STUN, and UPnP. The first 15 packets of the flow were used to create the feature matrix, and the arrival time and packet size were recorded for each of them. Flows less than 15 packets long were not considered. For more information about how to collect individual statistical characteristics of packets without creating an additional load on the network, see [9], [11].

The received data flows for each application were divided into training (train) and test (test) arrays in the ratio of 80% to 20%. Controlled methods for constructing the distance matrix (ET and RF) used a training array to build trees, and the test array was built on the basis of existing trees without using class labels. Unsupervised methods (RTE, Euclidean, and Manhattan distance) did not use class labels when creating distance matrices but for comparison with ET and RF, the results of clustering the test and training arrays are shown separately from each other.

B. Test 1. Visualization of clusters using the tSNE method

The widely used ML visualization method t-SNE (t-distributed Stochastic Neighbor Embedding) allows data to be dimensioned down and displayed in two-dimensional space so that points from one sample are most likely to be closer to each other, and from different ones - farther away from friend. The method is very dependent on random components and is not an accurate determination of the distances between points but it manages to get an idea of the possibility of clustering and the dataset. In Fig. 4, 5 , the selected applications were visualized using the t-SNE method. The circles mark the train sequence samples, and the crosses mark the test. The figure shows that the sample can be divided into clusters, although it has intersections in some places - for example, for TCP - the intersection of IMAPS, HTTP and Apple. The appearance of such disputed zones can be caused not only by the difficulties of clustering but also by the peculiarities of the operation of the protocols, as well as possible errors in automatic marking.

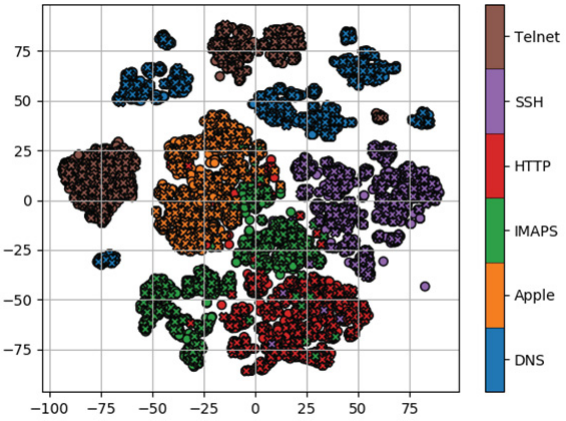

Fig. 4. Visualization of TCP clusters using the tSNE method

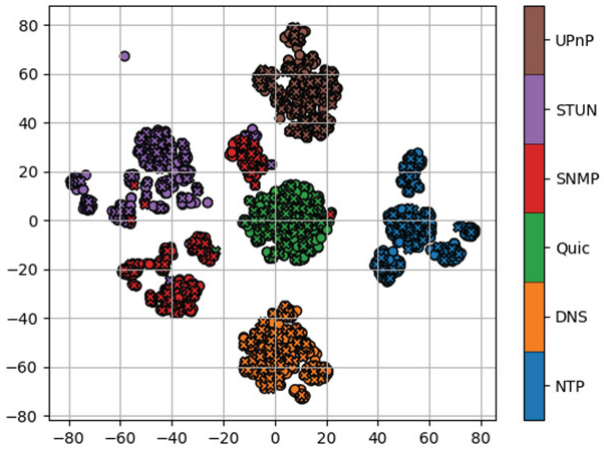

Fig. 5. Visualization of UDP clusters using the tSNE method

C. Test 2. The dependence of the results of clustering on the number of clusters for different distance matrices

Based on distance matrices obtained by five different methods (ET, RF, RE, Euclidean and Manhattan distance), Agglomerative clustering was performed with the number of clusters varying from 6 to 45. Based on the results of clustering, it was evaluated using the AMI, AMI, Completeness, Homogeneity, and Measure methods (Fig. 6, 7). Clustering was performed using the Scikit-learn library [13] of the Python programming language.

Fig. 6, 7 shows the ARI and AMI dependencies on the number of clusters for TCP applications. It can be seen that the best ARI and AMI indicators for the training sequence of TCP applications are achieved by the ET method (close to 1). The RF performance is slightly worse but the RF model looks less retrained, because the results for the test array for RF are higher (0.8) than for ET (0.75). The ARI for RTE for both arrays is zero and the AMI is 0.12, therefore, the RLE method failed to correctly perform clustering. When calculating the distances of Manhattan and Euclid, the results are not much better – they are in the range of 0.15-0.25, which also shows the unsuitability of these methods in this situation.

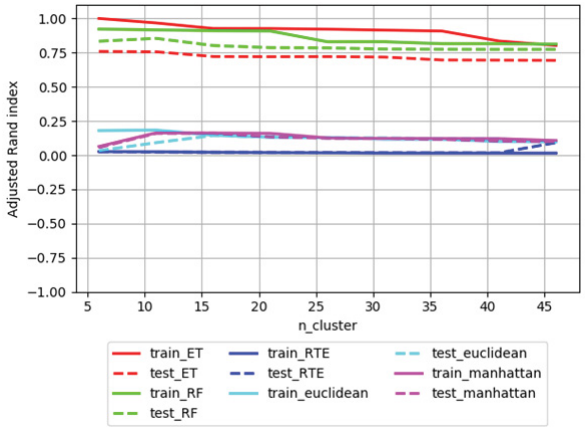

Fig. 6. The dependences ARI of the number of clusters for TCP applications

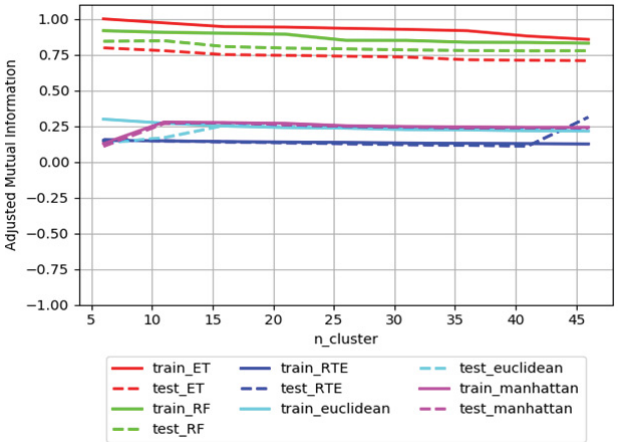

Fig. 7. The dependences AMI of the number of clusters for TCP applications

All the same conclusions are confirmed by the graphs of Homogeneity, Completeness and V_measure (Fig. 8-10). Such a significant difference in the results of clustering by ET, RF and RTE methods, Euclidean and Manhattan distances is explained by the presence of cluster construction control for ET and RF. For UDP applications the ET and RF methods show results of ARI and AMI within 0.95-1 (Fig. 11-12). Similarly to the situation with clustering of TCP applications, methods based on Manhattan and Euclidean distances do not cope with the task and are within 0-0.25. ARI RTE for UDP is slightly higher than for TCP-at the level of 0.3 - 0.4, and AMI-0.65 - 0.75.

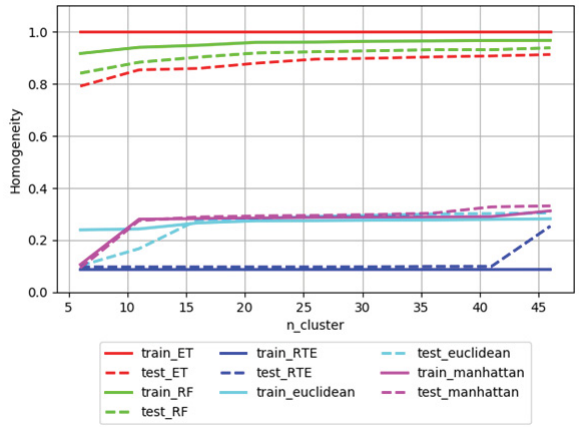

Fig. 8. The dependences Homogeneity of the number of clusters for TCP applications

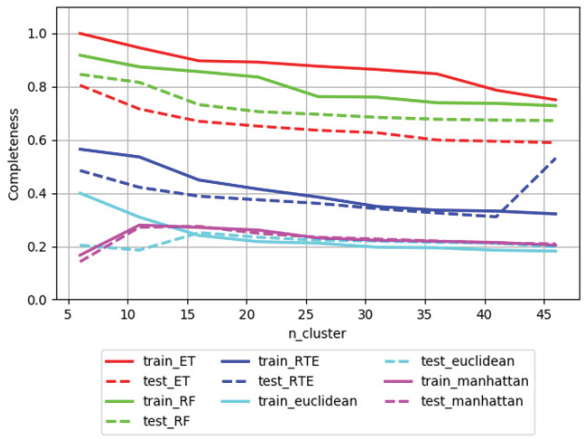

Fig. 9. The dependences Completeness of the number of clusters for TCP applications

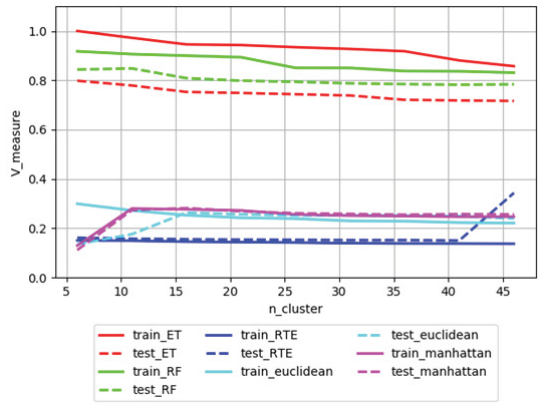

Fig. 10. The dependences V_measure of the number of clusters for TCP applications

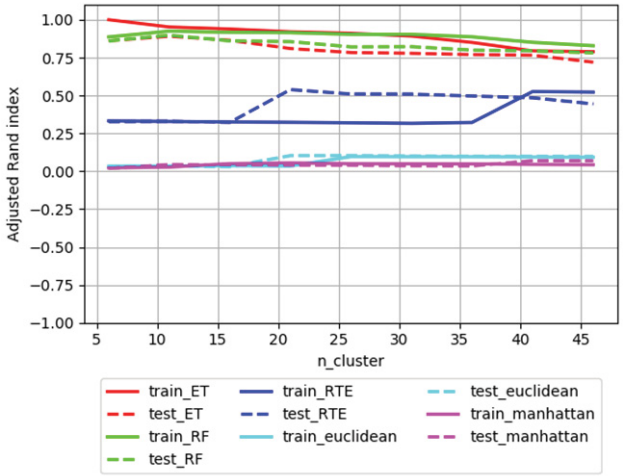

Fig. 11. The dependences ARI of the number of clusters for UDP applications

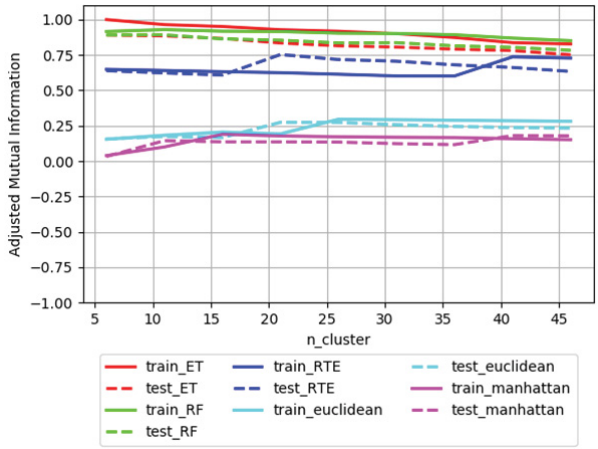

Fig. 12. The dependences AMI of the number of clusters for UDP applications

The uniformity of completeness and V-measure is also higher in this case than for UDP (Fig. 13-15). This result is explained by the cluster composition itself: in the case of UDP, clusters are more distinguishable from each other than in the case of TCP (Fig.8-10).

As the number of clusters increased, the ARI and AMI decreased for RF and ET (from 1 to 0.75), while for the other methods they increased slightly (0-0.2). As expected the indicators of uniformity began to increase, and completeness decreased. An excessive number of clusters is undesirable to use in the model, because it causes situations when one cluster is allocated for a single sample. In addition, the excessive number of clusters is difficult to interpret from the point of view of network management methods.

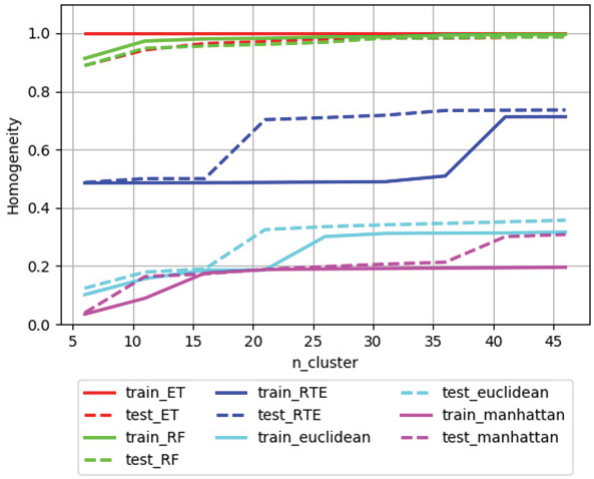

Fig. 13. The dependences Homogeneity of the number of clusters for UDP applications

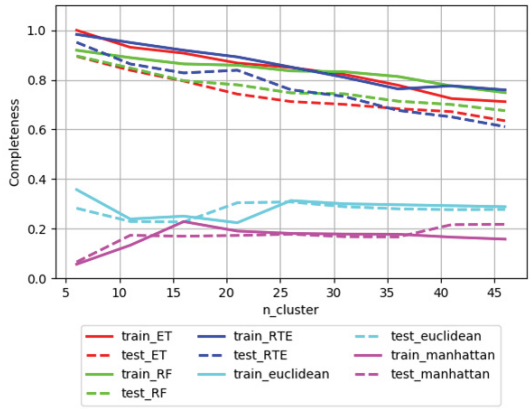

Fig. 14. The dependences Completeness of the number of clusters for UDP applications

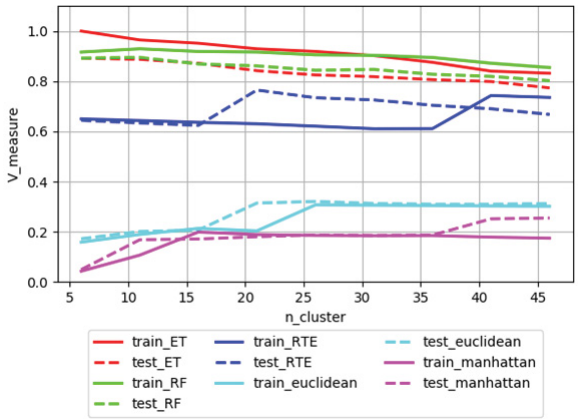

Fig. 15. The dependences V_measure of the number of clusters for UDP applications

D. Test 3. The dependence of the clustering results from the total number of flows for different distance matrixes

Fig. 16-17 shows the evaluation of application clustering results depending on the total number of flows. Due to the weak dependency, only ARI graphs are presented. For TCP applications, the total number of flows varies from 600 to 9000, and for UDP - from 200 to 2700. The number of flows from a single application in the samples is evenly distributed, and the train and test arrays are divided in the ratio of 80: 20, respectively. The number of clusters in this experiment remained fixed: 12 for TCP and 8 for UDP flows. The ET

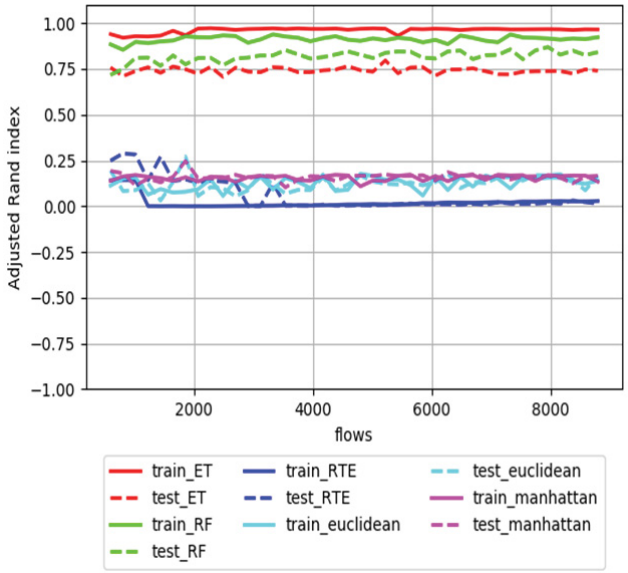

Fig. 16. The dependences ARI of the number of flows for TCP applications

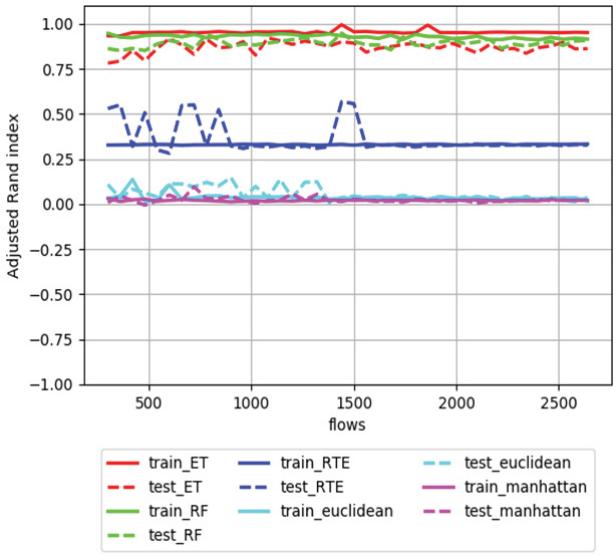

Fig. 17. The dependences ARI of the number of flows for UDP applications

and RF methods show stable results as the number of flows increases. This indicates that the clusters are well distinguishable from each other by these methods. Uncontrolled methods show some fluctuations in the results, but due to their low accuracy, the use of these methods in the model is unjustified.

VI. Conclusion

The paper offers a new traffic classification model that differs from other models in the following properties:

- Real-time operation due to a unique feature matrix based on the first 10-15 packets.
- Classification of any applications, including UDP and not just TCP as in most works [11].
- Collection of statistical information about the flows is performed without introducing additional load on the network [9].
- High classification accuracy (above 90% for TCP and above 95% for UDP) by improving accuracy at each stage of the classifier operation.
- High speed of the model (about 2 ms for processing a new request) due to the Supervised Learning methods.
- Discover new applications using Unsupervised Learning methods.
- Information about the nearest neighbors of the new application for the purpose of maintaining QoS during the TE stage.
- Minimum distance as a dynamically changing metric of cluster division for Agglomerative clustering as opposed to works where the criterion for dividing into clusters is their number.
- High accuracy of clustering due to the pre-calculated distance matrix using Random Forest and Extremely Randomized Trees methods.

Since the goals, conditions, experimental dataset, and features of the model are significantly different from other works, comparing the results with other authors may seem incorrect. Instead, a comparison of the five most promising methods for calculating the distance matrix is presented. It can only be noted that the methods using the pre-calculated distance matrix (ET and RF) showed better results compared to the uncontrolled construction of the distance matrix, as well as for network security problems in [5].

In future papers, it is planned to present results on automatic addition of new clusters, analysis of the operating time of various stages of the model, and TE methods for classified flows.

References

[1] V. A. Mankov and I. A. Krasnova, "Algorithm for dynamic classification of flows in a multiservice software defined network," *T-Comm*, vol. 11, pp. 37–42, 2017, *(in Russian)*. [Online]. Available: https://cyberleninka.ru/article/n/algoritm-dinamicheskoy-klassifikatsii-potokov-v-multiservisnoy-sdn-seti/viewer

[2] ——, "Traffic management task with dynamic qos detection in multiservice sdn networks," *Proceedings of the XI international industrial scientific and technical conference "information society Technologies"*, pp. 67–68, 2017, *(in Russian)*. [Online]. Available: https://www.elibrary.ru/item.asp?id=29802048

[3] Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang, and Y. Sun, "A survey of networking applications applying the software defined networking concept based on machine learning," *IEEE Access*, vol. 7, pp. 95 397–95 417, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2928564

[4] K. Takyi, A. Bagga, and P. Goopta, "Clustering techniques for traffic classification: A comprehensive review," *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pp. 224–230, 2018. [Online]. Available: https://doi.org/10.1109/ICRITO.2018.8748772

[5] Y. Wang, Y. Xiang, and J. Zhang, "Network traffic clustering using random forest proximities," *2013 IEEE International Conference on Communications (ICC)*, pp. 2058–2062, 2013. [Online]. Available: https://doi.org/10.1109/ICC.2013.6654829

[6] A. Mahmood, C. Leckie, R. Islam, and Z. Tari, "Hierarchical summarization techniques for network traffic," *2011 6th IEEE Conference on Industrial Electronics and Applications*, pp. 2474–2479, 2011. [Online]. Available: https://doi.org/10.1109/ICIEA.2011.5976009

[7] M. G. Salimath, "Network traffic analysis of hierarchical data using clustering," *International Journal of Science and Research (IJSR)*, vol. 6, pp. 1996 – 1999, 2017. [Online]. Available: https://www.ijsr.net/search_index_results_paperid.php?id=ART20172014

[8] W. Zhang, X. Wang, D. Zhao, and X. Tang, "Graph degree linkage: Agglomerative clustering on a directed graph," *ECCV*, p. 428–441, 2012. [Online]. Available: https://doi.org/10.1007/978-3-642-33718-5_31

[9] V. A. Mankov and I. A. Krasnova, "Collection of individual packet statistical information in a flow based on p4-switch," *Advances in Intelligent Systems and Computing*, vol. 1127, pp. 106–116, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-39216-1_11

[10] V. Deart, V. Mankov, and I. Krasnova, "Development of a feature matrix for classifying network traffic in sdn in real-time based on machine learning algorithms," *2020 International Scientific and Technical Conference Modern Computer Network Technologies (MoNeTeC)*, pp. 30–38, 2020. [Online]. Available: https://doi.org/10.1109/MoNeTeC49726.2020.9258314

[11] V. A. Mankov and I. A. Krasnova, "Classification of traffic flows in sdn networks using real-time machine learning methods," *Information technologies and mathematical modeling of systems 2019*, pp. 65–68, 2019, *(in Russian)*. [Online]. Available: https://doi.org/10.36581/CITP.2019.31.51.016

[12] K. Cho, K. Mitsuya, and A. Kato, "Traffic data repository at the wide project," *USENIX Annual Technical Conference, FREENIX Track*, 2000. [Online]. Available: https://www.iijlab.net/~kjc/papers/freenix2000.pdf

[13] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "Api design for machine learning software: experiences from the scikit-learn project," *ArXiv*, vol. abs/1309.0238, 2013. [Online]. Available: https://arxiv.org/pdf/1309.0238.pdf