Adjustment of the QoS Parameters on Routers with Neural Network Implementation

Irina Topalova
Department of Information Technology
University of Telecommunications and Post, Bulgaria
Sofia, Bulgaria
itopalova@abv.bg

Pavlinka Radoyska
College of Energy and Electronics
Technical University - Sofia
Sofia, Bulgaria
pradoiska@abv.bg

***Abstract*—Applying Quality of Service mechanisms to modern communications is essential for the efficiency and for the traffic reliability. The various Quality of Service methods are based on queues management depending on the individual traffic parameters. Choosing Quality of Service parameters on the edge network devices defines the management queue and packet discard/queued parameters on the intermediate devices. The proposed research explores the possibility of automatically adapting to the already selected class based Quality of Service policy of new users added to the backbone of the network. In addition, a method for queue adjustment has been suggested and tested, taking into account the current queue of the added user. A neural network is trained to automatically adapt new end users to the quality of service policy, already set by other end users and accepted by intermediate routers. The obtained results show that the automated adaptation of the Quality of Service parameters to the already set ones is possible for the intermediate routers. A software application, implementing the method in a network segment, is presented. The positive consequences of applying the proposed method are discussed.**

Keywords - traffic congestion, Quality of Service, early detection, queue management, neural network.

I. INTRODUCTION

This publication is based on our research reported at the ICAS 2018 conference [1]. Our research is aimed at creating a mechanism for automatically adjusting Quality of Service (QoS) parameters on routers using Neural Networks (NN). The configuration is based on Differentiated Services Code Point (DSCP) and Weighted Random Early Detection (WRED) queue management. The aim of QoS in communication networks is to guarantee the quality of message delivered by congestion management and congestion avoidance. This is achieved by dividing the traffic in queues and managing the queues individually, based on parameters, configured in any intermediate network device (router or switch). The packets are marked in the endpoint devices, according to the QoS model. Any intermediate device must be configured to create and manage queues, based on this model. Synchronized queue management in all devices is important for quality assurance. The purpose of our work is to find a mechanism by which any new device chooses its configuration parameters for queue management, based on the configuration parameters of the neighboring devices. The various QoS methods are based on queues management depending on individual traffic parameters. The chosen QoS parameters on the edge network devices define the management queue and packet discard/queued parameters on the intermediate devices. The proposed research explores the possibility of automatic adaptation to the already selected class based QoS policy of new users added to the backbone of the network. A NN, defined among many other types of neural networks NNs by Graupe [2] is trained to adapt new end users to the QoS policy, already set by other end users and accepted by the intermediate routers. The WRED method, described in Cisco guide [3], was applied to manage and to define the train and the test NN parameters. Additionally, a queue adjustment in a backbone router is proposed, taking into account the current queue of the added user. The automatic adaptation of additional networking devices to existing infrastructure with already-defined QoS policy would lead to the release of human resources and acceleration of the adaptation of traffic parameters in communication management. Properly tracking and setting the backbone queue in accordance to new added users, would improve the efficiency of the congestion management. The experimental results are presented, discussed and a further continuation of the study is proposed.

The rest of this paper is organized as follows. Section II describes the related to the research works. Section III describes and compares differentiated services and weighted random early detection methods. In Section IV the weighted random early detection with extension of explicit congestion notification is explained. Section V describes the proposed method for parameter adjustment and neural network implementation. Section VI gives the experimental results. Section VII introduces a software application for managing the QoS configuration process with using the NN. The conclusion with discussions close the article.

II. RELATED WORKS

Differentiated Services Code Point in IPv4 and Differentiated Services (DiffServ) in IPv6 are advanced instruments for traffic marking and queue management.

Khater and Hashemi [4] propose to use Differentiated Services Queues at the output port and move the flows between queues to prevent increasing delays of the flows. In another work [5] the authors use TSK fuzzy model to generate Differentiated Services Code Point values dynamically and update them in real time to improve QoS. The authors Sahu and Sar [6] have created an intelligent method to recognize incoming congestion problems earlier. They train a feedforward neural network with parameters equivalent to the total drop, average per packet drop, cumulative per packet drop, maximum packets drop and minimum packets drop, for send and receive features. The final solution is not automatically obtained as a result of the proposed method. It is left to the administrator. The results are not clearly represented and discussed, moreover the authors claim that their developed system missed some points of congestion. Within the model proposed in [7], the transmitted packets/traffic were predicted through a neural network, achieving prediction by alternating the input variables (Bandwidth, Congestion Algorithms, QoS, etc.). In this case, in TCP predictions, where one of the most important factors is related to the limitations of this protocol in both the sender and receiver, congestion improvements or methods for QoS were not considered. The different predictions have validity with respect to the real data, obtaining an average error of 4%. The authors in [8] apply a neural network to predict the actual time needed for transmitting the packet to the destination, depending on the number of hops. As neural network input train parameters, the authors use CWND (Congestion Window) as TCP state variable; Round-Trip Time (RTT) as the length of time it takes for a signal to be sent plus the length of time it takes for an acknowledgement of that signal to be received and the time elapsed from the last loss of a packet. However, this study does not use a method of prioritizing the traffic according to different types of priorities and they do not group traffic into classes according to the priority, given by the end routers/ users.

All mentioned researches do not apply more productive/ efficient methods, such as WRED in conjunction with Class-Based Weighted Fair Queueing (CBWFQ), proposed in Cisco guide [3]. They do not interpret the task we offer - to automatically adapt new end users to the quality of service policy, already set by other end-users and accepted by the intermediate routers.

III. Differentiated Services and Weighted Random Early Detection

Network congestion occurs when the volume of incoming traffic exceeds the bandwidth of the outgoing channel. Congestion avoidance mechanisms are trying to provoke TCP slow-start algorithm (RFC 2001), implemented in end devices. WRED and differentiated services, implemented in routers, become the most effective approach to prevent the congestions.

A. *Active Queue Management congestion avoidance mechanisms*

Congestion avoidance in routers is implemented by Active Queue Management (AQM) congestion avoidance mechanisms. Extra packets coming on the inbound interfaces are queued in buffers. The length of the queue is maintained within defined limits by dropping the packets. One of the first effective AQM mechanism is RED (Random Early Detection), proposed by Floyd and Jacobson [9] in the early 1990s. Two critical thresholds for the queue are defined: minimum queue length (***min***q) and maximum queue length (***max***q) and three queue management phases: no drop, random drop, and full drop, shown in Fig. 1. No drop phase is executed only for queue length from 0 to ***min***q. All packets are buffered. Random drop phase is for queue length from ***min***q to ***max***q. Some packets are dropt. Full drop phase is for queue length above ***max***q. All packets are dropped. The packet drop probability (random drop phase) is calculated based on the average queue length and the MPD (Mark Probability Denominator), Floyd and Jacobson [9]. MPD is the number of dropped packets when the queue size is equal to ***max***q. RED algorithm gives a decision for congestion avoidance problem but has some disadvantages. First, this mechanism does not affect non-TCP protocols. There are risks by insensitive protocols to embezzle the queue. Second, the packets from different TCP sessions are not dropped equally and there is a risk of global synchronization problem. Third, the number of dropped packets sharply jump to 100% when the queue size achieves ***max***q size. Different algorithms for the improvement of active queue management are proposed in [10]. WRED is a kind of class based queue management algorithms. It uses the same parameters as RED, but it has the ability to perform RED on traffic classes individually. Several traffic classes can be defined within a single queue. Each class has a specific level for the ***min***q, ***max***q and MPD. Packets are classified and joined to a specific class. Drop probability for each packet is calculated according to its class parameters. The packets with lowest ***min***q and/or the highest MPD are dropped preferentially. Every class has the same three phases as the RED algorithm. WRED management queue with three classes: AF1, AF2 and AF3 is presented in Fig. 2. AF1 and AF2 have the same ***max***q and MPD parameters. The AF1 ***min***q parameter has a lower value then the AF2 ***min***q parameter. Obviously the most packages are dropped from AF1 class, then from AF2 class and finally from AF3 class. The network traffic is divided in several queues to improve fairness in packet dropping. Each queue is managed by the RED, WRED or a similar algorithm. Weighted Fair Queue (WFQ), discussed by

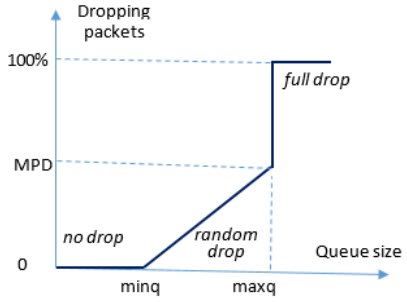


Figure 1. Random Early Detection phases.

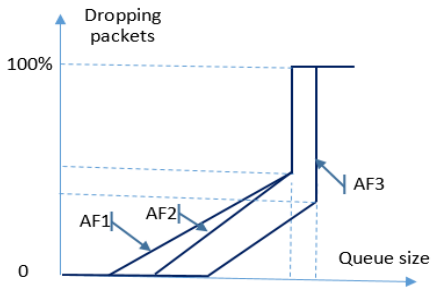


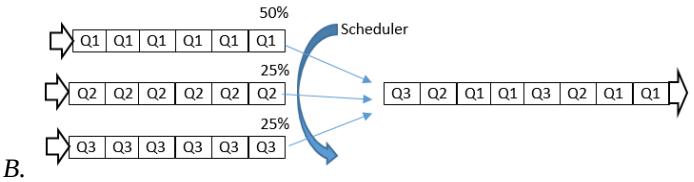
Figure 2. WRED phases.

Vukadinović and Trajković [11] is a data packet scheduling algorithm. All the queues share outbound bandwidth equally or by predefined ratios. The queues are visited one by one in the cycle period. Every queue sends the amount of packets, according to its share part of the outgoing capacity. The simple WFQ example is presented in Fig. 3. Q1 gets 50% of the outgoing capacity, Q2 – 25% and Q3 – 25%. The Scheduler visits Q1 and passes over 2 packets to the output. After that it visits Q2 and passes over 1 packet to the output; visits Q3 and passes over 1 packet to the output, and the cycle is rotated again.

B. Differentiated Services Quality of Service model

There are three main models for providing QoS in a network: Best Effort; Integrated Services (IntServ); Differentiated Services (DiffServ). DiffServ is called soft QoS model and uses WFQ and WRED algorithms. This model is based on user defined service classes and Per-Hop-Behavior (PHB). The flows are aggregated in traffic classes. The network service policies are defined for each class on any single node. Priorities are marked in each packet using DSCP for traffic classification.

The fields Type of Service (ToS) in IPv4 header (RFC 791) and Traffic Class (TC) in IPv6 header (RFC 2460) are predefined as Differentiated Services Field (DS Field) in RFC 2474. The first six bits of the DS field are used as a code point (DSCP) to select the PHB packet experiences at each node. DSCP values are described in RFC 2475. They determine the PHB of a packet. Four conventional PHBs are available: two border marks; Class-Selector PHB and Assured Forwarding (AF). DSCP = 000000 marks best effort behavior. All packets with this mark will be dropped when congestion occurs. This is the default PHB. DSCP = 101110 (46 in decimal) marks Expedited Forwarding (EF). EF PHB provides a virtual leased line and is used for critical traffic class as voice traffic. EF PHB provides low-loss, low-latency, low-jitter and assured bandwidth service. DSCP values of "xxx000" ("xxx" are the class selector bits) mark Class-Selector PHB and are used to assure backward compatibility with IP ToS model. DSCP values of "xxxyy0" mark Assured Forwarding (AF) PHB. "xxx" is for user defined AF class and "yy" is for drop precedence of a packet. "01" denotes low drop precedence, "10" – middle and "11" - high drop precedence. AF PHB classes are the subject of this paper.



B.

Figure 3. Weighted Fair Queue

C. DiffServ model configuration steps

1) Network traffic classification

Performs predominantly on the edge for QoS domain router - Cisco Guide [12]. The traffic type is defined by Access Control Lists (ACL) and joined to the specific AF class. Every class is associated with specific DSCP value. Inbound packets are marked with corresponding DSCP value on the edge routers of QoS domain and it is not recommended to change it in the intermediate routers.

2) Queue building

One or more AF classes can be aggregated in one queue, based on PHB parameters. The Queues can be three types: Strict priority queue (LLQ – Low latency queue); Class based queues (managed by WRED algorithm) end best effort queue.

3) Defining queue parameters

The WRED parameters are defined for every queue. For the Strict priority queue, the defined outbound bandwidth is guaranteed. The rest of outbound bandwidth is distributed between all other queues. For every class based queue, the following parameters are defined:

a) The portion of the bandwidth in percentage;

b) For each AF class (DSCP value) in the queue: min-threshold; max-threshold; MD (Mark-denominator).

Successful congestion avoidance depends on the proper execution of the above three steps. Especially on proper queue management definitions, described in 3) b).

IV. WRED FUNCTIONALITY EXTENDED WITH ECN

WRED drops packets, based on the average queue length exceeding a specific threshold value, to indicate congestion. Explicit Congestion Notification (ECN) (RFC 3168) provide end-to-end lossless communication between two endpoints over an IP routed network as given in [13] [14]. The ECN is an extension to WRED in that ECN marks packets instead of dropping them when the average queue length exceeds the *min-threshold* value. If there is a risk of congestion in a device (*min-threshold* < *queue* < *max-threshold*), instead of dropping the packages, they are marked and forwarded. When a marked packet arrives to the recipient, it sends a confirmation to the sender informing it of the available traffic congestion. As a result, the sender reduces his TCP window and the congestion decreases. This increases the bandwidth of the network because no unnecessary packets are ejected. This mechanism can be built into both - intermediate and end devices. There are also adaptations of ECN to UDP protocol explained in [15] - [17]. Two protocols which support ECN width UDP are defined: Datagram Congestion Control

Protocol (DCCP) (RFC5681) and Stream Control Transmission Protocol (SCTP) (RFC4960). The receiver sends a small special message to the sender, recommending to slow down the sending speed, because of congestion on the route. The next effective congestion avoidance is eXplicit Congestion Control Protocol (XCP), given in [18]. It works on the end and intermediate network devices (switches and routers) width TCP and UDP traffic. In addition, they use end-to-end bandwidth evaluation, to get high congestion estimation. Some of the open problems in Internet congestion control. Are discussed in RFC 6077.

ECN uses two bits - the ECN-capable Transport (ECT) bit and the CE (Congestion Experienced), which are the two least significant bits in the ToS field in the IP header. The four combinations of these bits have the following meaning: “0 0” indicates that a packet is not using ECN, “0 1” and “1 0” are set by the data sender to indicate that the endpoints of the transport protocol are ECN-capable and “1 1” indicates congestion to the endpoints i.e. packets reached a *max-threshold* of a router will be dropped. When ECN is enabled, the packets are treated as given in by Cisco Systems, Congestion Avoidance Configuration Guide, [19] and summarized by us, as follows:

1) If the number of packets in the queue is below the *min-threshold*, they are forwarded, whether or not ECN is enabled, and this is identical to the treatment a packet receives when WRED is only used on the network.

2) If the number of packets in the queue is between the *min-threshold* and the *max-threshold*, one of the following four cases is possible:

a) ECN field is “0 1” or “1 0” on the packet indicates that the endpoints are ECN-capable and the WRED algorithm determines that the packet should have been dropped based on the drop probability. In this case, the ECT and CE bits for the packet are changed to 1 and the packet is transmitted. So that, *the packet gets marked instead of dropped.*

b) *If the ECN field on the packet indicates that neither endpoint is ECN-capable* (that is, the ECT bit is set to 0 and the CE bit is set to 0), the packet may be dropped based on the WRED drop probability. This is the identical packet treatment when WRED is applied without ECN enabled.

c) If the ECN field on the packet indicates that the network is experiencing congestion (that is, both the ECT bit and the CE bit are set to 1), the packet is transmitted. No further marking is required.

3) If the number of packets in the queue is above the *max-threshold*, packets are dropped based on the drop probability. Such a treatment of a package is the same as when the router works only with WRED, without the ECN being set. The properly selected value of *min-threshold* is essential for the proper functioning of the network and congestion avoidance mechanism.

V. Proposed Method for WRED Parameter Adjustment Extended with ECN

In this study, we apply the WRED method for QoS in a network having end routers, a central/backbone router and an ad-hoc “New” router. The first task is to force the new added router to comply with the QoS requirements, which were pre-set in the central router. For this we propose a NN, intended to work in the central router, aiming to adjust the parameters of the “New” to the existing ones. The second task is to propose a method for appropriately determining the average queue and the *min-threshold* in the central router, when applying ECN, taking into account the current average queue of the added “New” router.

A. Investigated topology

We apply the WRED method for QoS, because it gives relation between AF classes and the most important queue traffic parameters. The topology shown in Fig. 4 is considered. It consists of two edge routers (Remotes 1 and 2), an intermediate router(Central) and an edge router "New", which is added later after the QoS parameters are set in the edge routers. WRED is implemented at the central/core routers of a network. Edge routers assign IP precedence to packets as the packets enter the network. With WRED, core routers then use these precedencies to determine how to treat different types of traffic *[18]*. The idea is to train a neural network (NN), implemented in the Central router with WRED parameters: AF class, min-threshold; max-threshold and MD, according to the IOS command random-detect. When an ad-hoc edge router "New" is added with its configured WRED (DSCP) requirements of its network, the already trained NN will approximate/adjust its MD to that already learned by the NN. This adjustment will be performed automatically without the need for any operator intervention. The new added router will have to comply with the pre-set QoS requirements.

B. Neural Network strategy

To conduct the experiment, we chose a neural network of Multi-Layer-Perceptron (MLP) type, training it with a BP (Backpropagation) algorithm. It was trained with the DSCP

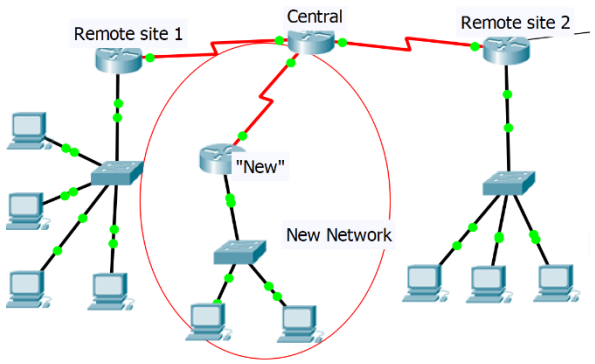


Figure 4. Investigated topology with edge routers (Remote site 1 and 2), intermediate (Central) router and the ‘New’ added router

values, corresponding to AF Classes 1,2,3 and 4, where Class 1 represents the 'worst queue', for low priority traffic and Class 4 – the 'best queue', for high priority traffic as first parameter. The second and third parameters in the input training set are *min-threshold* and *max-threshold*, defined by the command random-detect in the Central router. If the *min-threshold* is reached, Central router randomly drops some packets with the specified IP precedence. If the *max-threshold* is reached, Central router drops all packets with the specified IP precedence. The MLP has one output neuron and it represents the desired *MD*, where *MD* represents the fraction of packets dropped when the average queue depth is at the *max-threshold*. It means that one out of every *MD* packets will be dropped. Table I represents the correspondence between AF classes, DSCP values and drop precedence. After the NN was trained, a combination of different DSCP values with proposed bandwidth percent for each AF class was provided at its input layer, in order to simulate these parameters, send by the 'New' router. According to the "New" requirements/parameters, the Central router generates new *min-threshold* and *max-threshold* and forwards the new information to the NN inputs.

TABLE I. AF CLASSES AND CORRESPONDING DSCP VALUES

Assured Forwarding	Low Drop (DSCP)	Medium Drop (DSCP)	High Drop (DSCP)
Class 4	AF41 (34)	AF42 (36)	AF43 (38)
Class 3	AF31 (26)	AF32 (28)	AF33 (30)
Class 2	AF21 (18)	AF22 (20)	AF23 (22)
Class 1	AF11 (10)	AF12 (12)	AF13 (14)

As a result, the trained NN gives an output with approximated *MD* value, which is near the value defined initially by the Central. In this way, the 'New' router will be forced to "comply" with the chosen QoS policy.

C. Queue adjustment

The average queue size is based on the previous average and current size of the queue, as given in equation (1) [19]:

$$Q_{avr} = \left(old_{avr} * (1 - \frac{1}{2^n})\right) + (curr\_queue\_size * \frac{1}{2^n}), \quad (1)$$

where Q_{avr} is the calculated value of the average queue size, old_{avr} is the previous value of the queue, *curr_queue_size* is the current queue and *n* is the exponential weight factor, a user-configurable value. The analysis of this equation shows that for high values of *n* the previous average queue size becomes more important. At the same time for low values of *n* the average queue size Q_{avr} will closely track the current queue size.

In our case we propose a change in the given equation (1) in order to accommodate C:Q_{avr} of Central router, taking into account also the current queue size $N: curr\_queue\_size$ of

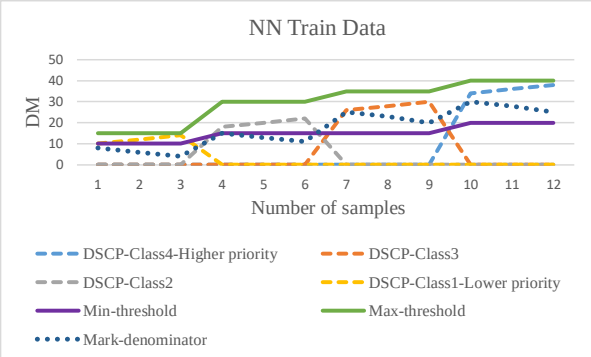


Figure 5. NN train data with initial QoS parameters. The ordinate represents the number of packets in the queue and DM

the New router. We choose the critical moment when the previous value of old_{avr} reaches the *min-threshold* in the Central router, i.e.

$$C: Q_{avr} = \left(C: min\_threshold * (1 - \frac{1}{2^n})\right) + ((C: curr\_queue\_size + N: curr\_queue\_size)/2)) * \frac{1}{2^n}$$

We denote here the parameter ***New Average Queue(NAQ)*** as:

$$\boldsymbol{NAQ} = (C: curr\_queue\_size + N: curr\_queue\_size)/2. \quad (2)$$

As the right choice of *n* is the exponential weight factor and is user-configurable value, we will run the experiment with different *n* values to determine the better option.

VI. EXPERIMENTAL RESULTS

The initially selected MLP network structure is 6-4-1 and is trained to MSE (Mean-Square-Error) = 0.1. The train data are given in Fig. 5. They have 12 input samples as combinations between DSCPs, *min-threshold* and *max-threshold*, defined in Remote 1 and 2. After conducting the test phase with the 'New' data, the obtained *MD* approximation is shown in Fig. 6. The approximation error E_{APPROX} is calculated according to (3), where MD_{RSi} is the initial real system value for the Central router, for i-th input

$$E_{APPROX} = \sqrt{\sum_{i=1}^{n} \frac{(MD_{RS_i} - MD_{NN_i})^2}{n}} \quad (3)$$

combination, MD_{NNi} is the NN response, and *n* is the number of input combinations. The obtained results using this NN topology are given in Fig. 7. In this case, E_{APROX} is 2.56. Obviously, it is necessary to improve the MLP parameters by training a network with an improved structure of 6-6-4-1 and with more iterations, aiming to reach a smaller MSE. In this case, we apply MSE of 0.01. Better obtained results are given in Fig. 8. In this case, E_{APROX} is 0.91. Thus, based on the training of the optimized neural network with the defined AF classes and their initial matching random-detection parameters, we obtain a relatively good MD approximation. Further work is foreseen to test the NN with more combinations of input parameters. For testing the WRED

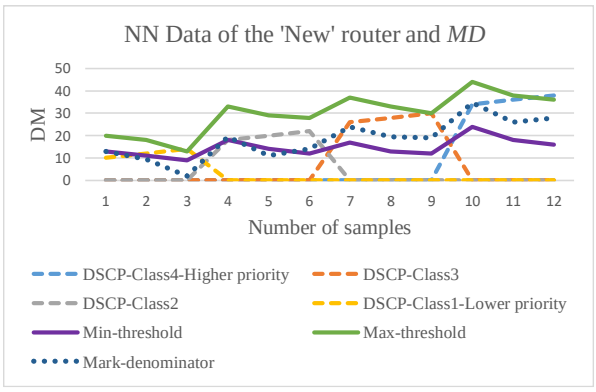


Figure 6. NN 'New' test data with *MD* approximation. The ordinate represents the number of packets in the queue and DM

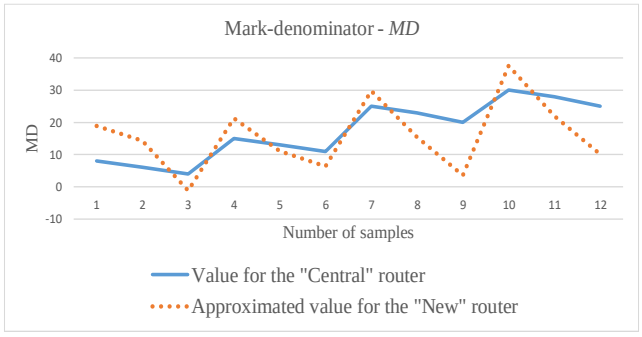


Figure 7. *MD* approximation with MLP – 6-4-1. The ordinate represents the number of packets in the queue and DM

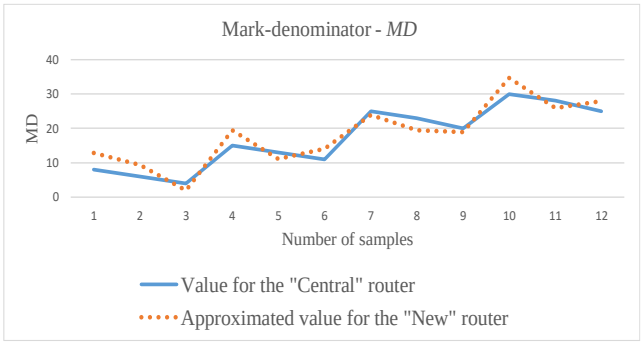


Figure 8. *MD* approximation with MLP – 6-6-4-1 The ordinate represents. the number of packets in the queue and DM

functionality, extended with ECN, corresponding to cases 2)/ a) and 2)/ c), we choose the critical moment when the previous value of old_{avr} reaches the *min-threshold* in the Central router. The goal is to determine the value of C: Q_{avr}, taking into account also the current queue size *N: curr_queue_size* of the New router. We tested how *C:* Q_{var} tracks *NAQ*, depending on its different peak changes, according to (2), and how the exponential weight factor *n* influences the adaptation. Fig. 9, 10, 11 and 12 show the adaptation of *C:* Q_{var} when n=4,3,2,1 correspondingly. The obtained results show that a large factor of n=3 represented in Fig. 10, smooths out the peaks and lowers the queue length. The average queue size will not

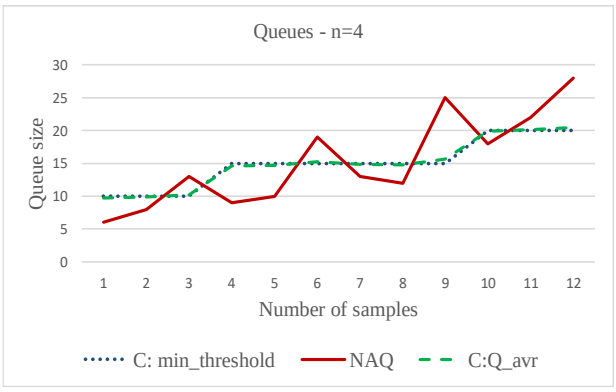


Figure 9. Adaptation of $C: Q_{avr}$ with n=4. The ordinate represents the number of packets in the queues and DM

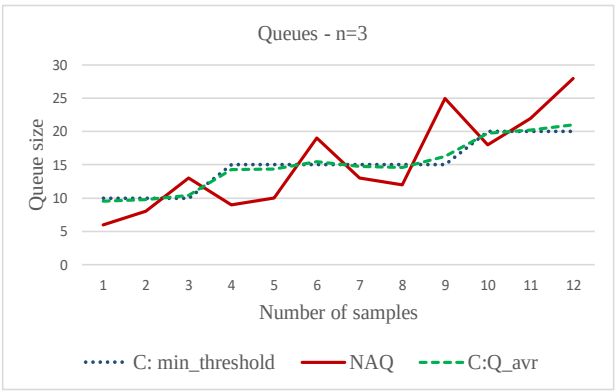


Figure 10. Adaptation of $C: Q_{avr}$ with n=3. The ordinate represents the number of packets in the queues and DM

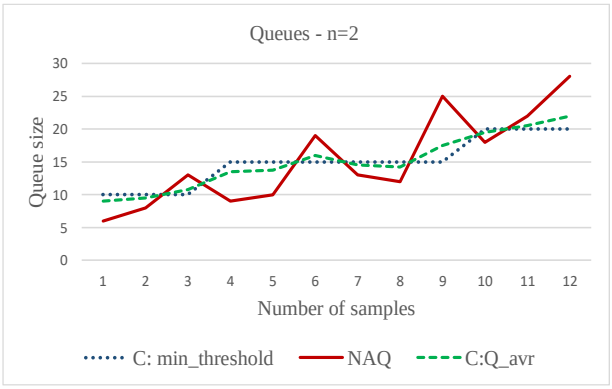


Figure 11. Adaptation of $C: Q_{avr}$ with n=2. The ordinate represents the number of packets in the queues and DM

probably change very quickly, avoiding dramatic fluctuations in size. The WRED process will be slow to start dropping packets and the slow-changing average C: Q_{var} will accommodate temporary peaks in traffic. But if the value of n gets too high (n=4, Fig. 9), WRED will not react to congestion. Packets will be transmitted or dropped as if WRED does not work. For low values of n (n=2, Fig. 11), the C: Qvar tracks closely the current queue size. The resulting average value may fluctuate adequately with the changes in

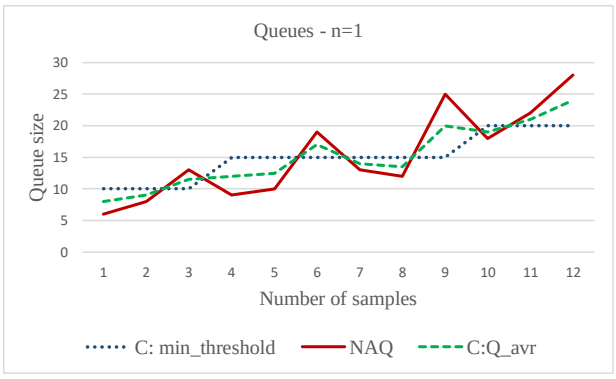


Figure 12. Adaptation of $C: Q_{avr}$ with $n=1$. The ordinate represents the number of packets in the queues

the traffic levels of both Central and New routers (i.e. of NAQ). Once the resulting queue falls below the minimum threshold, the process will stop dropping packets. If the value of n gets too low (n=1, Fig. 12), WRED will overreact to temporary resulting traffic bursts and will drop traffic unnecessarily. Thus, the proposition of n = 2 seems to be the most appropriate in terms of queue efficiency.

VII. REALTIME REMOTE ROUTERS RECONFIGURATION

The purpose of our research is to get better QoS management by synchronizing the queue management parameters on the routers in one network segment without the manual reconfiguration of any new router. Moreover, we try to synchronize the queue management parameters on all routers in network segment after the reconfiguration of only the Central router.

A. Processes and management

A data-flow diagram is shown on Fig. 13. The NN block and The Manager are software blocks that work on an external machine (PC or a laptop). The Manager is responsible for the process navigation. Each router can perform the role of a master (Central) or a subordinate. Router1, Router2 and New on Fig. 13 are subordinate routers.

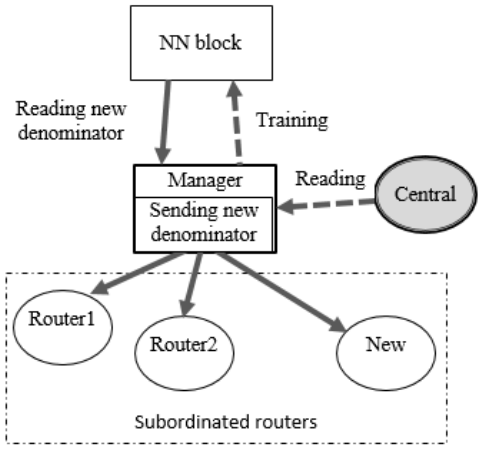


Figure 13. Real-time configuration process

There are two types of processes: training and reconfiguring. The training process includes: reading QoS parameters from the Central router, preparing and sending the training matrix to the NN. The reconfiguring includes: reading the QoS configuration from the subordinate router, preparing and sending a query to the NN and, based on the NN response, prepares the synchronized configuration parameters and sends them to the subordinate router.

There are two possible situations: (1) inclusion of a new router; (2) reconfiguring. In the first situation, the NN is already trained and all routers' configurations are synchronized. A new router with autonomous QoS configuration is included in the network segment. The Manager makes connection to the new router, extracts proper denominator from the NN and reconfigures the new router. The new router starts work in synchronization with all routers in the segment.

In the second situation, all routers' configurations are synchronized but the QoS configuration on the Central router is changed. The Manager makes connection to the Central router, extracts the new queue management parameters and trains the NN. Then it makes connections to all other routers consequently: reads their current QoS parameters, sets them to the NN, gets the new proper denominator and reconfigures the routers. All routers QoS configurations are synchronized again.

The communication between the Manager and the NN block performs in off-line mode being based on computer operating system mechanisms. The communication between the Manager and the routers is accomplished via SSH protocol. Therefore, any router in the management network segment must be registered in the Manager.

B. Manager block implementation

This Manager is written as multithreading Windows application by C# programming language. As hardware devices are used Cisco routers, platforms 2800/2900 with IOS 15.0. The Manager interface has two tabs: Registration and Processes given in Fig. 14 and Fig. 15 respectively. The Central router and the subordinated routers are separated in different blocks for their different roles. The IP address, username and password for SSH connection are saved for each registered router, shown in Fig. 16. The IP address is used as a router identificator.

The training process starts after the button "Train (off-line)" is selected. The result of the training process is displayed in the textbox on the right as shown in Fig. 15. There are two buttons for reconfiguration, according to the two situations mentioned above. Only the selected router is reconfigured after the selection of the button "Reconfigure". All routers, included in the list "Subordinated routers" are reconfigured after the selection of the button "Reconfigure All". The result of this process is displayed in the textbox "Reconfiguration Results" as shown in Fig. 17. The

administrator must troubleshoot the problem in case of appearance of router reconfiguration problems.

All communications in the dataflow diagram, shown in Fig. 13, are of a machine-machine type. The training and the reconfiguration are made automatically but all processes must be started by a person. This approach is appropriate for the first situation, described above – inclusion of a new router. The router registration has to be made manually and the manual start of the reconfiguration should not lead to a significant processing delay. The second situation would be more flexible if the Central router sends a signal to the Manager for the configuration change automatically, thus forcing the training and reconfiguration processes. Solving this problem is a matter of our future research. We need to find a mechanism to alert the Manager about the changes of the Central router configuration. The Manager also must work as a server to listen permanently to that signal.

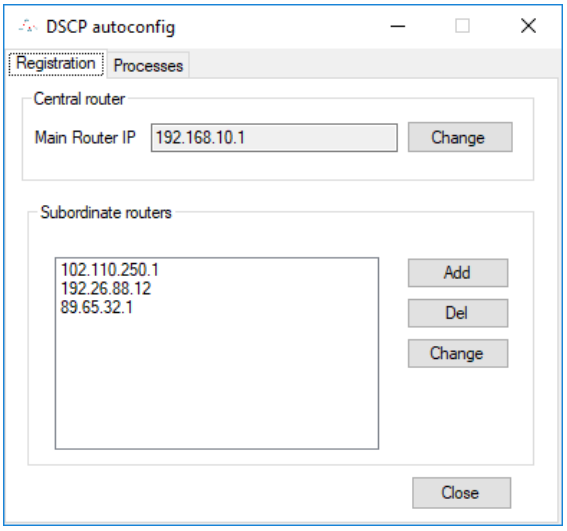


Figure 14. Manager software – Registration tab

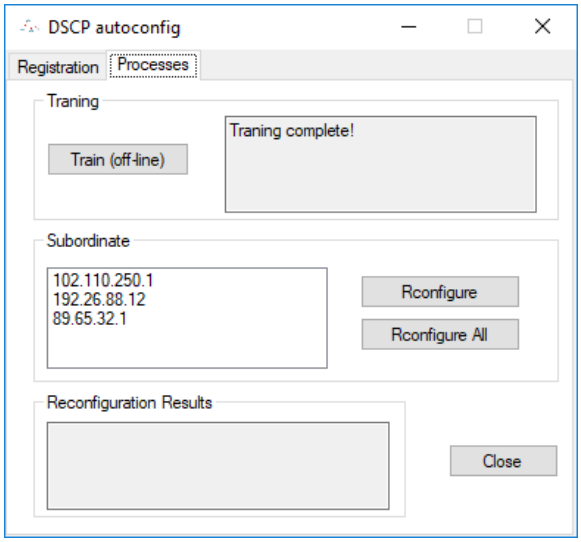


Figure 15. Manager software – Processes tab

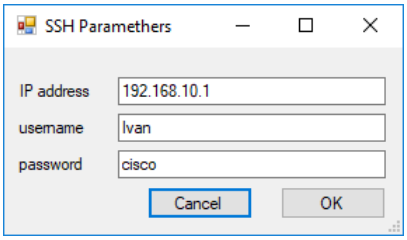


Figure 16. Managing the parameters for SSH connection

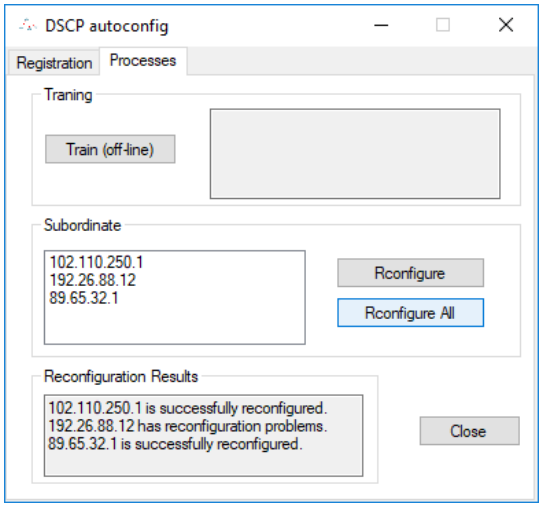


Figure 17. Manager software – Registration tab width reconfiguration results

VIII. CONCLUSION

In this research, a MLP neural network was trained, aiming to automatically adapt new end users to the quality of service policy, already set by other end-users and accepted by the intermediate routers. The WRED method was applied to manage and to define the train and test NN parameters. The proposed method shows good MD approximation results for the tested input set. The main benefit of the automatic adaptation of additional networking devices to existing infrastructure with an already-defined QoS policy would lead to the release of human qualified resources, needed for manual QoS parameter pre-settings. It also would accelerate the traffic parameters adaptation in communication management and in real-time communication. The proposed accommodation of C:Q_{avr} in the Central router, taking into account also the current queue size N: $curr\_queue\_size$ of the New router, choosing the critical moment when the previous value of old_{avr} reaches the *min-threshold* in the Central router, shows good tracking especially when n=2. If the value of n gets too low (n=1, Fig. 12), WRED will overreact to temporary resulting traffic bursts and will drop traffic unnecessarily. Thus, the proposition of n = 2 seems to be the most appropriate in terms of queue efficiency.

A software application was developed to verify the proposed method. It is installed on the external computer system and works as a manager for all processes: reading the initial configuration, preparing the training matrix, starting

the NN training, getting the new proper denominator from already trained NN, reconfiguring the subordinated routers. The verification indicates that the method is applicable.

As further work, the input training and test sets may be increased to generalize the method. The idea is to train the NN with the same standard AF classes but with much more possible/ reasonable combinations of min-max thresholds, together with a proper proposal for the required link bandwidth at the outputs of the NN. The investigated topology given in Fig. 4, may be tested with more Remote routers and many "New" routers, to test the behavior of the Central router. In this case, different NNs could be trained with QoS parameters defined in the different Remotes, and the NN outputs may be combined in input train data for a generalized neural network, to give the final MD proposal. Also, software modules will be developed to integrate the neural network into a module of the Central router operating system, for direct data exchange between the routers. Aiming to achieve/solve this task, we envisage the use of Python programming language, suitable for implementation in networking operating systems.

REFERENCES

[1] I. Topalova, P. Radoyska, "Control of Traffic Congestion with Weighted Random Early Detection and Neural Network Implementation", ICAS 2018, The Fourteenth International Conference on Autonomic and Autonomous Systems, pp. 8-12, Nice, France, 20-24 May 2018

[2] D. Graupe, 'Deep Learning Neural Networks: Design and Case Studies', World Scientific Publishing Co Inc. pp. 57–110, ISBN 978-981-314-647-1, July, 2016.

[3] Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2, Chapter: Congestion Avoidance Overview https://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfconav.html#wpxref11086, last accessed 16.11.2018.

[4] A. Khater and M. R. Hashemi, "Dynamic Flow Management Based on DiffServ in SDN Networks," Electrical Engineering (ICEE), Iranian Conference on, Mashhad, 2018, pp. 1505-1510.
doi: 10.1109/ICEE.2018.8472638

[5] J. Li, L. Yang, X. Fu, F. Chao and Y. Qu, "Dynamic QoS solution for enterprise networks using TSK fuzzy interpolation," 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Naples, 2017, pp. 1-6. doi: 10.1109/FUZZ-IEEE.2017.8015711

[6] Y. Sahu and S. K. Sar, 'Congestion analysis in wireless network using predictive techniques', Research Journal of Computer and Information Technology Sciences, ISSN 2320 – 6527 vol. 5(7), pp. 1-4, September, 2017.

[7] A. F. Luque Calderón, E. J. Vela Porras and O. J. Salcedo Parra, 'Predicting Traffic through Artificial Neural Networks', Contemporary Engineering Sciences, vol. 10, no. 24, pp. 1195 - 1209 HIKARI Ltd, www.m-hikari.com https://doi.org/10.12988/ces. 2017.710146, 2017.

[8] S. S. Kumar, K. Dhaneshwar, K. Garima, G. Neha and S. Ayush, 'Congestion Control in Wired Network for Heterogeneous resources using Neural Network', International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 5, May 2013, ISSN: 2277 128X, pp.533-537, 2013.

[9] S. Floyd and V. Jacobson, 'Random Early Detection Gateways for Congestion Avoidance', IEEE/ACM Transactions on Networking, Networking, vol. 1 No. 4, pp. 397-413, August, 1993,Available:http://www.icir.org/floyd/papers/early.twocolumn.pdf: accessed August, 2018.

[10] G. Abbas, Z. Halim and Z. H. Abbas, 'Fairness-Driven Queue Management: A Survey and Taxonomy', IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 324-367, First quarter 2016. doi: 10.1109/COMST.2015.2463121, 2016.

[11] V. Vukadinović and L. Trajković, 'RED with Dynamic Thresholds for improved fairness', Proceedings of the 2004 ACM symposium on Applied computing (SAC '04). ACM, New York, NY, USA, 371-372. DOI: https://doi.org/10.1145/967900.967980, 2004.

[12] QoS: DiffServ for Quality of Service Overview Configuration Guide, Cisco IOS Release 15M&T, January 16, 2013 Available:https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_dfsrv/configuration/15-mt/qos-dfsrv-15-mt-book.html: accessed August, 2018.

[13] B. Trammell, M. Kühlewind, D. Boppart, I. Learmonth, G. Fairhurst, and R. Scheffenegger, "Enabling Internet-Wide Deployment of Explicit Congestion Notification", Passive and Active Measurement. PAM 2015, pp 193-205, March, 2015. DOI: https://doi.org/10.1007/978-3-319-15509-8_15

[14] M. Kühlewind, S. Neuner and B. Trammell, "On the state of ECN and TCP options in the internet", Proceedings of the Passive and Active Measurement, 2013, Hong Kong SAR, China, 2013. DOI: https://doi.org/10.1007/978-3-642-36516-4_14

[15] S. McQuistin and C. Perkins, "Is Explicit Congestion Notification usable with UDP?", IMC '15 Proceedings of the 2015 Internet Measurement Conference, Pages 63-69, Tokyo, Japan — October 28 - 30, 2015 doi: 10.1145/2815675.2815716

[16] E. Stergiou, D. Liarokapis, C. Angelis and F. Vartziotis, "Vigorous Distance Learning Applications Using the Stream Control Transmission Protocol", Science Journal of Education. Vol. 5, No. 6, pp. 262-267, 2017.

[17] S. Saini and A. Fehnker, "Evaluating the Stream Control Transmission Protocol Using Uppaal", EPTCS 244, 2017, pp. 1-13, 2017. DOI: 10.4204/EPTCS.244.1

[18] J. Wang, J. Chen, S. Zhang and W. Wang, "An Explicit Congestion Control Protocol Based on Bandwidth Estimation", IEEE Global Telecommunications Conference - GLOBECOM 2011, Kathmandu, 2011, pp. 1-5. .,doi:10.1109/GLOCOM.2011.6134086, 2011.

[19] QoS: Congestion Avoidance Configuration Guide, Cisco IOS XE Release 3S, Cisco Systems, Inc. 170 West Tasman Drive San Jose, CA 95134-1706 USA