Control of Traffic Congestion with Weighted Random Early Detection and 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 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. A neural network is trained to automaticlly adapt new end users to the quality of service policy, already set by other end-users and accepted by the 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, and the positive consequences of applying such a method are mentioned.**

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

I. INTRODUCTION

The aim of Quality of Service (QoS) in communication networks is to guarantee the quality of message delivering by congestion management and congestion avoidance. This is achieved by dividing the traffic in queues and manage any queue 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 each new device chooses its configuration parameters for queue management, based on the configuration parameters of the neighboring/end 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 automatically adapting to the already selected class based QoS policy, of new users added to the backbone of the network. A Neural Network (NN), defined among many other types of neural networks NNs by Graupe [1] is trained to adapt new end users to the QoS policy, already set by other end-users and accepted by the intermediate routers. The Weighted Random Early Detection (WRED) method, described in Cisco guide [2], was applied to manage and to define the train and test NN parameters. The automatic adaptation of additional networking devices to existing infrastructure with an already-defined QoS policy would lead to the release of human resources and acceleration of the adaptation of traffic parameters in communication 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 proposed method for weighted random early detection parameter adjustment is presented. Section V gives the experimental results. The conclusion closes the article.

II. RELATED WORKS

The authors Sahu and Sar [3] 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 method, it is left to the administrator. The results are not clear represented and discussed, moreover the authors claim that their developed system missed some points of congestion. Within the model proposed by Calderón, et al. [4], 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,

 ISBN: 978-1-61208-634-7

obtaining an average error of 4%. The authors Kumar, et al. [5] 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 [2]. They do not interpret the task we offer - to automaticlly 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 [6] 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

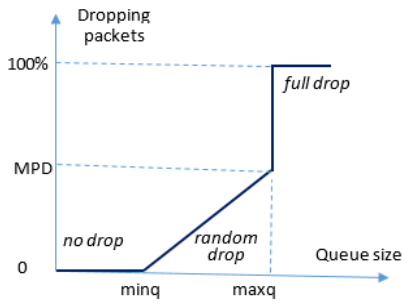

Figure 1. Random Early Detection phases.

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 [6]. 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. Firstly, this mechanism does not affect non-TCP protocols. There are risky by insensitive protocols to embezzle the queue. Secondary, 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 by Abbas et. al. [7]. 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 on Fig.2. AF1 and AF2 have the same ***max***q and MPD parameters. The AF1 ***min***q parameter has the les 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 Vukadinović and Trajković [8] 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 Differentiated Services Code

 ISBN: 978-1-61208-634-7

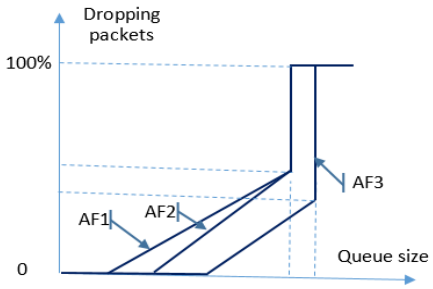

Figure 2. WRED phases.

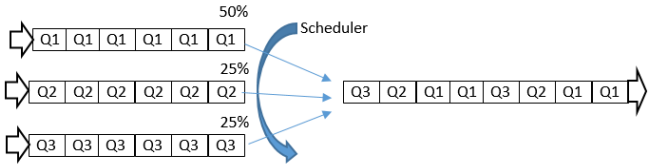

Figure 3. Weighted Fair Queue

Point (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 dropt 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.

C.DiffServ model configuration steps

1) Network traffic classification

Performs predominantly on edge for QoS domain router - Cisco Guide [9]. 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 is defined guaranteed outbound bandwidth. The rest of outbound bandwidth is distributing between all other queues. For every class based queue is 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. PROPOSED METHOD FOR WRED PARAMETER ADJUSTMENT

In this study, we apply the WRED method for QoS in simple network and use NN to adjust parameters in new added 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. 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

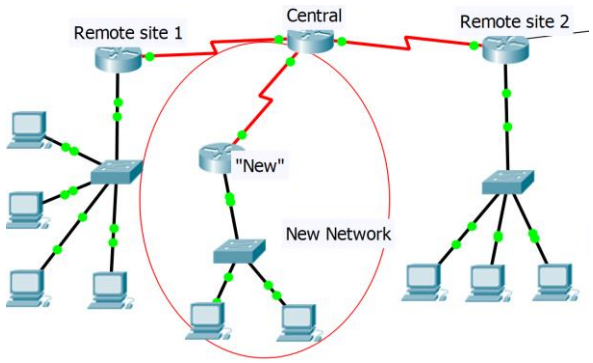

Figure 4. Investigated topology with edge routers (Remote site 1 and 2), intermedite (Central) router and the ‘New’ added router

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 MLP type, training it with a BP (Backpropagation) algorithm. It was trained with the DSCP 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 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 result the 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.

V. 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_{APROX} is calculated according to (1), where MD_{RSi} is the initial real system value for the Central router, for i-th input

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

combination, and MD_{NNi} is the NN response, and n is the number of input combinations. In this case EAPROX is 2.56. Obviously, it is necessary to improve the MPL parameters, training a network with 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. The obtained better results using this NN topology are given in Fig. 7. In this case EAPROX is 0.91. Fig.8 represents the NN 'New' test data with MD approximation. 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.

VI. 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

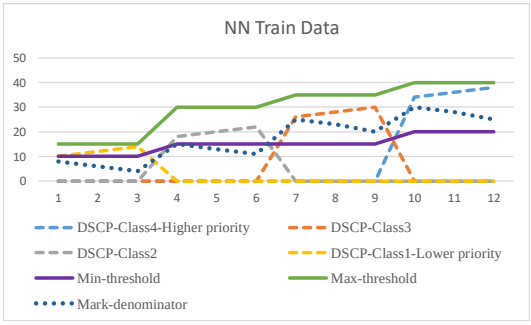

Figure 5. NN train data with initial QoS parameters

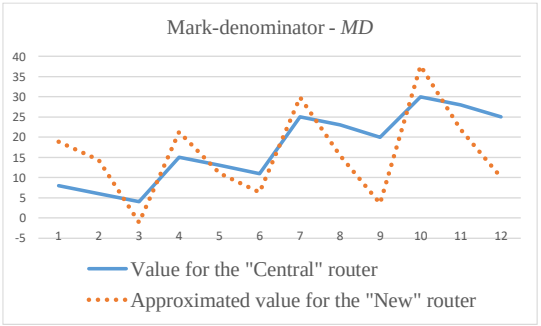

Figure 6. *MD* approximation with MLP – 6-4-1

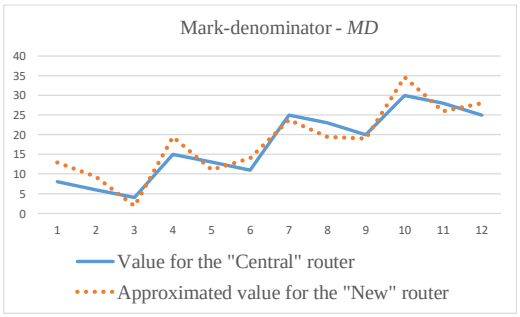

Figure 7. *MD* approximation with MLP – 6-6-4-1

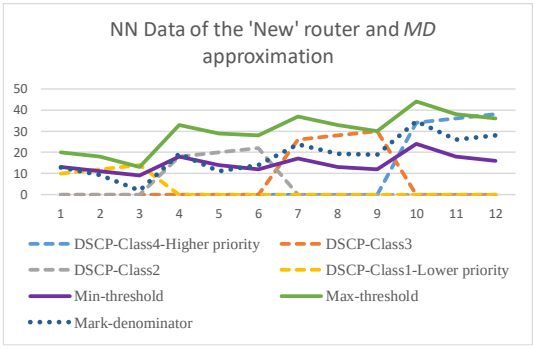

Figure 8. NN 'New' test data with *MD* approximation

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. 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. As hardware devices we intend to use Cisco routers, platforms 2800/2900 with IOS 15.0.

REFERENCES

[1] 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.

[2]https://www.cisco.com/c/en/us/td/docs/ios/120s/feature/guide/fswfq26.html, last accessed 21.04.2018.

[3] 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.

[4] A. F. Luque Calderón, E. J. Vela Porras, 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.

[5] 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.

[6] 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

[7] 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, Firstquarter 2016. doi: 10.1109/COMST.2015.2463121, 2016.

[8] 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.

[9] QoS: DiffServ for Quality of Service Overview Configuration Guide, Cisco IOS Release 15M&T, January16, 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