Differentiating Services with Dynamic, QoS-Aware Queuing

Mahsa Pourvali
Department of Computer Engineering, Ferdowsi University of Mashhad
Mashhad, Iran
Mahsa.pourvali@stu-mail.um.ac.ir

Mohammad Hossein Yaghmaee
Department of Computer Engineering, Ferdowsi University of Mashhad
Mashhad, Iran
Yaghmaee@ieee.org

Seyed Amin Hosseini Seno
Department of Computer Engineering, Ferdowsi University of Mashhad
Mashhad, Iran
hosseini@um.ac.ir

Amir Hossein Mohajerzadeh
Department of Computer Engineering, Ferdowsi University Of Mashhad
Mashhad, Iran
Ah.mohajerzadeh@stu-mail.um.ac.ir

Abstract— **The growth of different types of applications in the Internet arises the need for supporting Quality of Service (QoS) for them. Differentiating services aims to differentiate among applications in order to provide their requirements based on the level of QoS they need and their characteristics. In this paper we introduce a dynamic QoS-Aware queuing algorithm to differentiate services in IP based networks, which consists of a packet-by-packet dynamic classification algorithm and a weighted round robin scheduling algorithm. The proposed dynamic QoS-aware queuing provides the QoS required for delay sensitive applications and improves the performance of the system, meaning satisfying more users. Plus, the algorithm is capable of providing a satisfactory fairness among flows. The main advantage of the algorithm is its ability to dynamically adapt its parameters to the changes in the network.**

Keywords-component; differentiating services, packet classification; packet scheduling; Quality of Service;

I. INTRODUCTION

In the last years, With the increases in various types of applications in the Internet and as a result, the wide range of service expectations, it seems crucial for the Internet to provide some levels of QoS for applications. Moreover, this continuous growth in the number of users, expecting high quality services makes it vital for the Internet to consider servicing more users while providing some satisfactory fairness among them. Differentiating services is categorizing applications based on their needs, which can be done by scheduling, classification or dropping algorithms. Among the QoS parameters requested by the applications, delay is the most important one, as it is critical for applications such as some controlling, sensor and multimedia or interactive ones. In some applications there is no difference if packets are receiving late at the destination or not being received at all. So, in our proposed scheme we consider delay sensitive (DS) packets requirements while providing QoS in the Internet. Contrary to most queuing algorithms, we do not only focus on the situation where, the number of demands exceeds the number of resources and the network is to service some applications better than others. Our proposed scheme aims to service more users, moreover it provides a satisfactory degree of fairness among applications. Although there is no agreement in defining Fairness in network, every algorithm follows some definition based on their aims.

In this paper, we propose a new queuing paradigm; Dynamic QoS-Aware algorithm for differentiating services in IP based networks, which consists of an adoptive packet-by-packet classifier and a simple weighted round robin scheduler. It has got the ability to address the DS applications' requirements and provide fairness among flows and also improve the performance of the system, meaning satisfying more users. Moreover the proposed algorithm is capable of adopting itself to the changing nature of the Internet.

The rest of this paper is organized as follows: in section 2 we discuss the related work, in section 3 we provide details on the proposed algorithm, in section 4 we present the simulation results and in section 5 we conclude our paper.

II. RELATED WORK

Differentiating services in the Internet seems necessary in order to provide different requirements of various types of applications. Today's Internet is coupled with wide variety of applications, expecting their requested services. So it is vital for the Internet to provide their requirements up to a level. Besides, the Internet is to provide some degree of fairness among these applications. Much has been done in order to address these criteria. Proposals that have been suggested try to provide solutions by introducing active queue management schemes, packet classification, packet scheduling algorithms and so on. For example algorithms, which are proposed in [2], [3] and [4] are dropping approaches,

while papers such as [5], [6], [7] and [8] has worked on scheduling paradigms.

RED [9], is a widely deployed and the most referenced queuing algorithm. Much has been done in order to improve RED's performance in different aspects. For instance, [10] has worked in order to improve RED's fairness. [11] and [12] has tried to solve the problem of RED's dependency on parameters.

Noncongestive queuing algorithm (NCQ) is proposed in [1]. NCQ differentiates packets based on their size into two categories, congestive and non-congestive and prioritizes small packets by using a priority scheduling. The authors discuss that small packets do not impose much delay on the network and it is not fair if they receive significant delay.

III. THE PROPOSED ALGORITHM

Based on what is discussed above, we consider three queues in each node: q_1, q_2 and q_3. The lower number shows the higher priority. The classifier classifies packets into one of the three queues. And the scheduler selects the queue from which the packet is chosen for getting service.

A. THE CLASSIFICATION ALGORITHM

When a packet comes to the classifier, with a probability, it is queued into one of the queues. The probability that the packet enters q_1, q_2 and q_3 is P_1, P_2 and P_3 respectively. As a result:

$$P_1+P_2+P_3=1 \quad (1)$$

Each packet is classified according to the packet length (PL), delay sensitivity (DS) and priority given (PG). The system as a whole is shown in Fig. 1.

Delay Sensitivity Parameter (DS)

When a packet comes to a node, first it is recognized whether the packet is delay sensitive or not. The application, which is sensitive to delay, marks its packets with the tag DS. Consider that our proposed scheme is not only application based, as DS packets receive priority up to a threshold. The reason is that, the application is not the best place to decide whether or not packets must be given high priority, as it is not aware of other application's requirements, the network available resources, and the routers traffic load. The best place to decide is the intermediate nodes, which are aware of the network current situation. We consider our approach not completely service based nor completely application based.

After recognizing the delay sensitivity of the packet, if the packet is delay sensitive, it will not be classified into q_2 for sure. And if the packet is not sensitive to delay, it will not be classified into q_0. So for each packet there are two possible queues into which the packet can be classified:

$$\text{If DS} = 1 \Rightarrow P_3 = 0, \text{ if DS} = 0 \Rightarrow P_1 = 0$$

Packet Length Parameter (PL)

As it is discussed in [1], when small packets are serviced first, the average goodput of the system is improved. Plus, the average delay of the prioritized

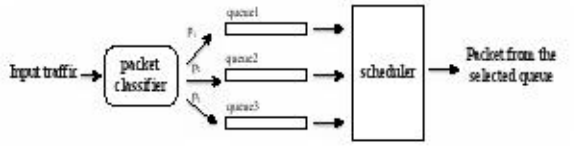


Figure 1. The model of the system.

packets decreases while it does not impose any noticeable delay on large packets. [1] determines a certain and predefined threshold, up to which small packets get high priority. They have proved analytically that, in this way, the imposed delay on low priority traffic in not significant. We follow this idea, but we consider that having a certain and predefined threshold, does not work well under the changing nature of the Internet, as different types of applications tend to transmit various types of traffic with various characteristics. As a result, it seems inefficient and even impossible to predetermine a fixed threshold based on the packet length. Additionally, it is unfair and too strict to divide packets into the two categories congestive and noncongestive, based on a fixed threshold and prioritize noncongestive packets. This means that, when the threshold is 130 bytes as in the NCQ algorithm, for example, there is no difference among a 140 byte packet and a 1400 byte packet, as none of them get priority, but there is a big difference among a 130 byte packet and a 140 byte packet, since the former gets high priority while the latter gets low priority. As it was mentioned earlier, fairness is one of our algorithm's major goals. With considering what is discussed above and in order to omit the fixed threshold in our proposed paradigm, we introduce a probability function according to which packets get high priority regarding their length (Fig. 2). The larger a packet is, the less it is probable for it to get high priority. Notice that high priority for DS packets means entering q_0, and for non-delay sensitive (NDS) packets means entering q_1. The variable β introduces the average length of the packet. We calculate the probability for the packet to get priority regarding its length, in compare with the average length of packets come to the node. This sounds completely fair, as the packet is compared with the history of the packets, which had come to the node.

In order to separate small and large packets dynamically and in compare to the packets' history, the average packet length is calculated and it is updated with the arrival of each new packet. And as a result, the center of the probability function moves to the right and left. So the aggressiveness of the algorithm on prioritizing packets regarding their length dynamically changes according to the history of the packets. This adopting feature of the algorithm makes it capable of classifying packets dynamically based on the current situation of the network and the history of the packets and adopting itself to the changing nature of the Internet. According to what is discussed, the probability function P based on packet length PL, is calculated as follows:

$$P = \begin{cases} 0 & PL \geq \alpha_2 \\ 0.318 \cot^{-1}(PL - \beta) - 1 & \alpha_1 < PL < \alpha_2 \\ 1 & PL \leq \alpha_1 \end{cases} \quad (2)$$

$$\beta = ave\ (PL) \quad (3)$$

Priority Given Parameter (PG)

The priority given (PG) we calculate here is the same as threshold given in [1], which represents the number of prioritized packets to the number of total packets. But, we have got two different thresholds for DS and NDS traffic. The reason is that we do not want prioritizing small NDS packets impose delays on small DS packets. *PG* is calculated as follows:

$$PG = priority\ given\ packets\ /\ total\ packets \quad (4)$$

In order to omit the fixed and predetermined threshold for both delay and NDS packets, we have proposed the probability function, which is shown in Fig. 3. It is reasonable and fair, as the higher is the priority given, the less is probable for the incoming packets to get high priority. We consider the initial value 0.05 for the *k*, the same as fixed threshold given in [1], as it is proved that it leads us to our discussed aims.

But we consider that it will not be efficient to determine a fixed threshold for priority given, as the network's condition tends to change during the time. So, in order to adapt to the changing nature of the Internet, we do not determine *k* to be fixed. But, its value is calculated as a function of changes in probability function P regarding packet length (PL). That is to say, when the packets tend to be smaller in the network, and the algorithm becomes stricter in giving high priority to the packets, *k* is increased in value, which means the probability function moves to the right. The reason is that, by considering this PG threshold, we aim to control the delay imposed to lower priority packets. And the delay caused by each packet, is relevant to the packet's length. So when for example there are relatively small packets in the network, each prioritized packet imposes lower delay on lower priority ones due to its small size. So the number of prioritized packets increases due to the fact that what we want is achieving the maximum performance, while not imposing significant delay on others and as the length of the packets are relatively small, we still can service more small packets. This way, the PG threshold dynamically can adopt itself to the network's changes.

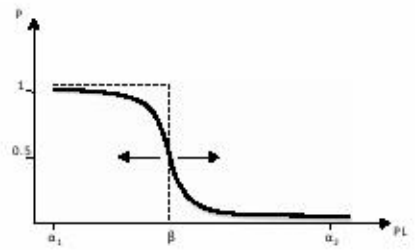


Figure 2. The probability function P_i based on the parameter PL.

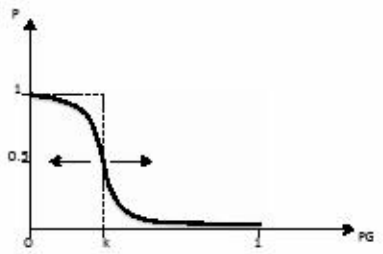


Figure 3. The probability function P_i based on the parameter PG.

Regarding what we have discussed above, the probability function p according to PG is as follows:

$$P = \begin{cases} 0 & PG = 1 \\ 0.318 \cot^{-1}(PG - k) - 1 & 0 < PG < 1 \\ 1 & PG = 0 \end{cases} \quad (5)$$

$$K = \frac{1}{1 + \frac{(\alpha_2 - \beta)}{\beta - \alpha_1}} \quad (6)$$

We calculate the final probability with which the packets get high priority the average of P based on PL and PG because both parameters are important the same.

B. THE SCHEDULING ALGORITHM

As the classification algorithm has prioritized small packets up to a threshold, in order to improve the goodput of the system, it is reasonable to consider the first queue as a priority queue. The reason is that, by servicing these small packets we will improve goodput while there is no significant impact on others. For q_1 and q_2 we consider a simple weighted round robin scheduler. But in calculating weights for packets belonging to q_1 and q_2 we only consider the large packets in q_1 and large packets in q_2. The reason is that. The q_1 consists of the non-prioritized, DS packets and prioritized, NDS ones. The prioritized, NDS packets are those we gave high priority in order to improve the performance of the system. Moreover, they do not impose significant delay on large packets in the same queue nor on the large packets in the queue with the lowest priority.

IV. EVALUATION RESULTS

We have implemented our evaluation plan on the OPNET simulator. We assume two different classes, which are DS and NDS. Flows belonging to each class send packets in different lengths. We use the dumbbell topology, as shown in Fig. 4. We measure average goodput for DS flows, NDS flows and for the system, as they are defined in [1]. So we have:

$$Goodput = \frac{original\ data}{time} \quad (7)$$

$$Average\ goodput = \frac{\sum_{i=1}^{n} Goodput_i}{n} \quad (8)$$

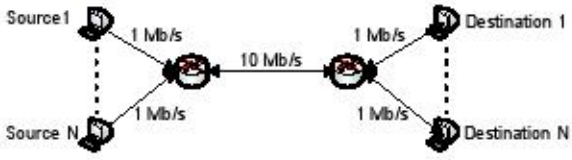


Figure 4. Simulation topology.

Where Original Data is the number of bytes received at the destination and Time is the amount of time needed for such data delivery. $Goodput_i$ is the goodput of the ith flow. We also measure application efficiency and user satisfaction the same as they are defined in [1], which is based on the worst and average task completion time. Additionally we measure DS and NDS packets' delays while transmitting. We also measure fairness again based on what is introduced in [1] as the Application Satisfaction index (ASI). As it is mentioned before we follow the delay-wise definition given in [1] for fairness. So for ASI we again have the same definition as [1]:

$$ASI = 1 - \frac{\left|\sum_{i=1}^{n} Delay_i - \frac{Data_i}{TotalData} Delay_{max}\right|}{n\,Delay_{max}} \qquad (9)$$

Where $Delay_i$ is the queuing delay of the ith flow, $delay_{max}$ is the maximum queuing delay among the flows. $Data_i$ is the total data of the ith node, which is transmitted to the destination and TotalData is the total amount of data of all nodes received at the destinations. By network monitoring we defined α_1 and α_2 to be 100 and 1500 bytes respectively.

In the first place, we considered the number of delay-sensitive flows to the 10 percent of the total flows, and we increased the number of flows in the simulation to examine the network's behavior. As can be seen from Fig. 5, delay-sensitive flows achieve significant performance gain in terms of goodput. With the increase in the number of flows, the goodput of DS flows decreases (Fig. 5a). This is not symptomatic since we do not want to increase the effect on non-delay sensitive flows and the probability of getting priority for DS packets decreases as the PG increases. Besides, in our proposed scheme the impact on NDS flows is not noticeable at all (Fig. 5b). There is also a satisfactory improvement in the average goodput of the system as it is shown in Fig. 5c. The interesting point is that our proposed algorithm also improves the system's average task completion time in compare with both NCQ and Droptail, as can be seen in Fig. 6a while there is no impact on the system's worst time (Fig. 6b). We consider this as a result of the algorithm's capability to adopt itself to the changing situation of the network. Additionally, Fig. 7 shows that our proposed scheme provides fairness in a higher degree in compare with both NCQ and Droptail. This is the consequence of providing fairness among different flows based on their characteristics and not only among classes of flows but the good point is that we do not need to store any information of each flow so the algorithm remains stable. The other point is that, our proposed paradigm reduces DS packets delay considerabley (Fig. 8a), while the delay imposed to NDS packets is not significant in comparison to the improve we made (Fig. 8b).

In the second place, we arranged the number of DS

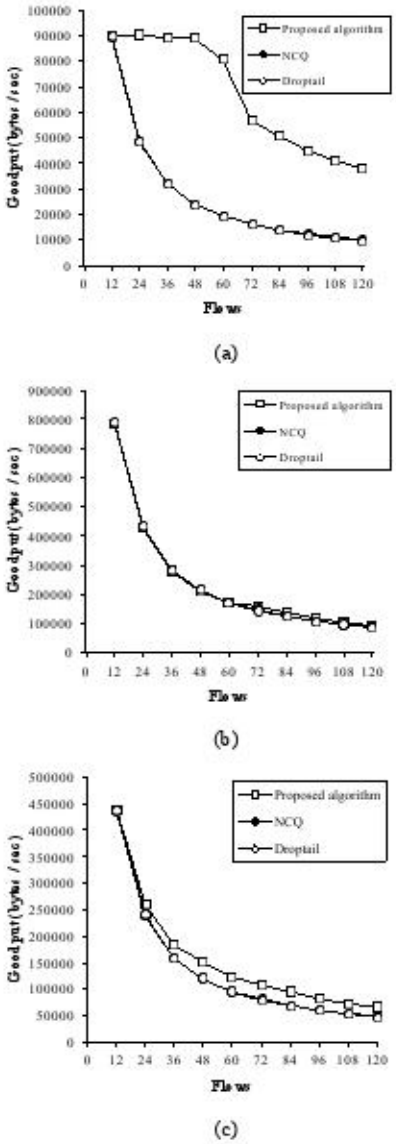


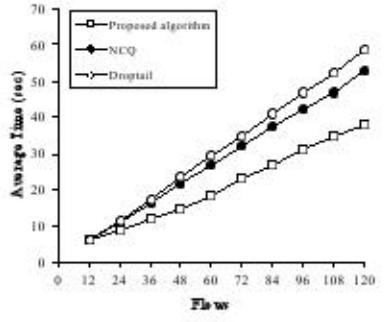
Figure 5. Goodput. (a) Average goodput of non-delay sensitive flows. (b) Average goodput of delay-sensitive flows. (c) Average system goodput.

traffic to the 20 percent of the total traffic. As Fig. 9 illustrates, there is again significant improve in goodput for DS flows and the system in comparison with NCQ and Droptail but the cost imposed on NDS is not noticeable (Fig. 9b). Fig. 10 portrays that there is a satisfying decrease in average time in compare to NCQ and Droptail, while the worst time remains the same. The fairness and delay are also improved. (Figs. 11 and 12)

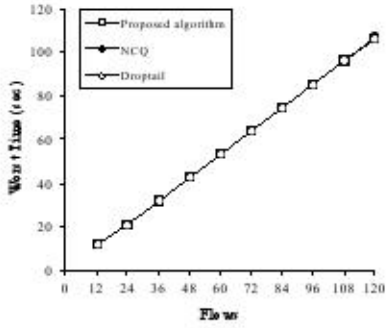
V. CONCLUSION

In this paper, a new dynamic QoS-aware queuing algorithm is proposed. It provides three priority queues in order to differentiate services among the traffic. It classifies packets based on three parameters: delay sensitivity, packet length and priority given. By evaluating the algorithm, provided results show significant performance gain in terms of goodput and delay for delay-sensitive flows and the system, while does not impose any noticeable cost for NDS ones. Moreover, it reduces the average time required to complete flows while not increase the worst time. And as a result leads to satisfy

more users in the Internet in a specific time. Also, it improves the fairness among flows, which we measured through the ASI factor.



(a)



(b)

Figure 6. (a) Average task completion time in the system. (b) Worst task completion time in the system.

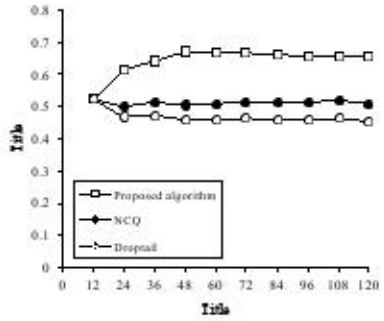
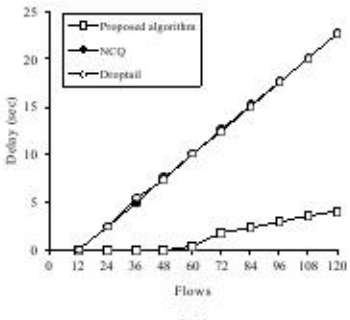
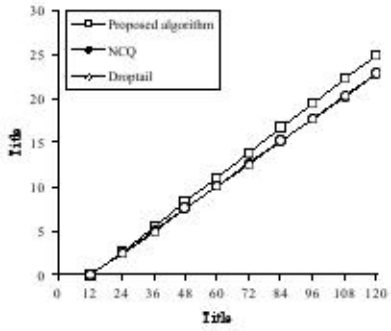


Figure 7. Application Satisfaction Index.

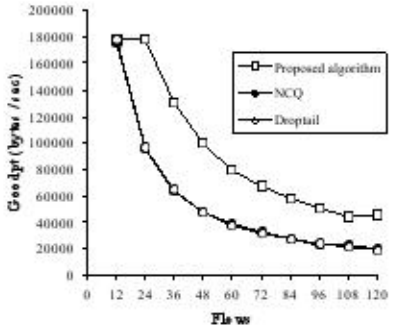


(a)

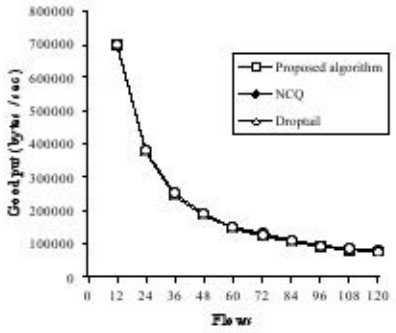


(b)

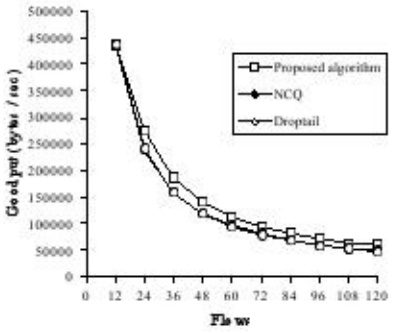
Figure 8. (a) Average delay sensitive packets' delay. (b) Average non-delay sensitive packets' delay.



(a)

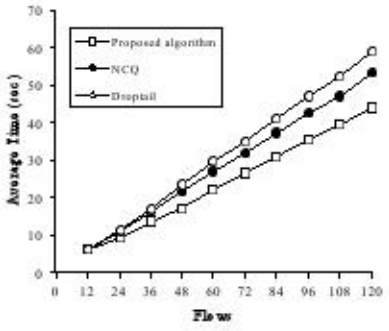


(b)

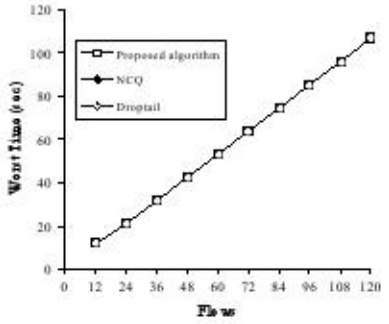


(c)

Figure 9. Goodput. (a) Average goodput of non-delay sensitive flows. (b) Average goodput of delay-sensitive flows. (c) Average system goodput.



(a)



(b)

Figure 10. (a) Average task completion time in the system. (b) Worst task completion time in the system.

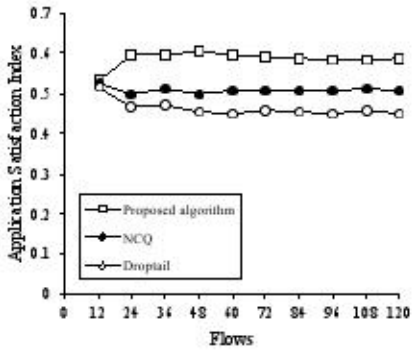
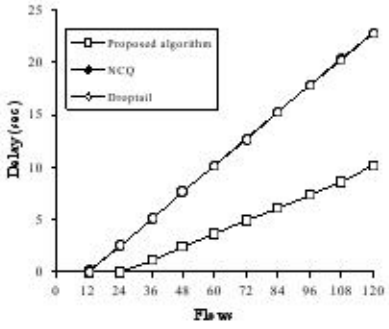
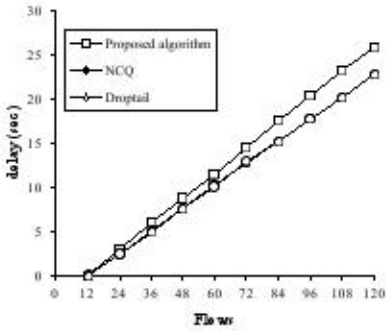


Figure 11. Application Satisfaction Index.



(a)



(b)

Figure 12. (a) Average delay sensitive packets' delay. (b) Average non-delay sensitive packets' delay.

REFERENCES

[1] L. Mamatas, V. Tsaoussidis, "Differentiating services with non congestive queuing (NCQ)," IEEE Transactions on Computers, vol. 58, No. 5, pp. 591-604, 2008.

[2] J. Kim, H. Yoon, I. Yeom, "Active Queue Management for Flow Fairness and Stable Queue Length," IEEE Transactions on Parallel and Distributed Systems, vol. 22, No. 4, pp. 571-579, April 2011.

[3] B. J. Hwang, I. S. Hwang, P. M. Chang, C. Y. Wang, "QOS-aware Active Queue Management for multimedia services over the internet," Eurasip Journal on Wireless Communications and networking, vol. 2011, No. 589863, pp. 1-12, 2010.

[4] P. Erika, J. Juan, "Selective packet dropping for VoIP and TCP flows," Telecommunication Systems, vol. 46, pp. 1-16, 2011.

[5] R. Avudaiammal, P. Seethalakshmi, "Design and implementation of a hybrid packet scheduling algorithm on Network processor based router for enhancing QoS of multimedia applications," European Journal of Scientific research, Vol. 72 No. 2, pp. 245-262, 2012.

[6] Y. Lee, J. Lou, Ju. Luo, and X. Shen, "An Efficient Packet Scheduling Algorithm With Deadline Guarantees for Input-Queued Switches," IEE/ACM transactions on networking, Vol. 15, No. 1, pp. 212-225, 2007.

[7] S. Li, S. Ren, Y. Yu, X. Wang, I. Wang, G. Quan, "Profit and Penalty Aware Scheduling for Real-Time Online Services," IEEE Transactions on industrial informatics, Vol. 8, No.1, pp. 78-89, 2012.

[8] Scheduling Packets with Values and Deadlines in Size-Bounded Buffers(mesle 5)(avazkonambarefrencedige)

[9] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, Vol. 1, No. 4, pp. 397-413, August 1993.

[10] A.H. Altalhi, M.S.B.M. Azmi, A.M. Al-Kharasani and S.A. Ali, "A Modified Packet Marking Algorithm to Improve Bandwidth Fairness in Diffserv Networks," *Research Journal of Information Technology, Vol.* 1, pp. 1-11, 2012.

[11] J. Chen, C. Hu, and Z. Ji, "An Improved ARED Algorithm for Congestion Control of Network Transmission," Mathematical Problems in Engineering, vol. 2010, pp. 1-14, 2010.

[12] M.P. Tahiliani, K.C. Shet, T.G. Basavaraju, "CARED: Cautious Adaptive RED gateways for TCP/IP networks," Journal of Network and Computer Applications, Vol. 35, No. 2, pp. 857-864, 2012.