Implementation of modified AQM mechanisms in IP routers

Joanna Domańska, Adam Domański, and Tadeusz Czachórski

Abstract— **The article is an attempt to answer the question if it makes sense to modify the way of choosing packets to reject in AQM mechanisms. Simulation and analytical research of RED and DSRED mechanisms shows that their efficiency grows when packet is received from the front of the queue. It is especially conspicuous when taking into account the self-similarity of traffic. However implementation of the above mentioned mechanisms in real router does not corroborate such a clear advantage over Drop-From-Front strategy. In this article the results of analytical, simulation and real router research based on the Linux operating system have been presented.**

I. Introduction

The standard IP router working according to the best effort rule rejects incoming packets only when it is necessary because of lack of space in input buffers. The active queue management, recommended by IETF, assumes that earlier rejection of packets is made with the growing possibility of router overload. RED algorithms enhances the efficiency of transfer and cooperate with TCP congestion windows mechanisms in adapting the flows intensity to the congestions at a network [1]. The inventor of RED mechanism - S. Floyd has assumed that in the normal work of AQM router the rule of dropping packets (end of the queue, head of the queue) should not have any influence on the data transfer delay [2]. Currently more and more applications use UDP protocol to transfer data (VoIP, VOD). As a result the characteristics of data streams in the network has significantly changed. The authors of article have shown that rejecting packets from head of the queue for RED and DSRED mechanisms greatly affects the decrease of packets average time traversal through router [3]. For the purpose of this article authors have implemented chosen algorithms in the router environment based on Linux operating system. The article presents the results acquired in the process of research and confronts them with the results acquired using analytical and simulation methods.

Sections II gives basic notions on active queue management, Section III presents briefly a self-similar model used in the article. Section IV contains analytical and simulation analysis of the problem of choosing either tail or front packets to drop in RED and DSRED mechanisms. Section V presents the results of the implementation of the above mentioned mechanisms in real router, some conclusions are given in Section VI.

Manuscript received December, 2007 and revised February, 2008.

This research was financed by Polish Ministry of Science and Higher Education project no. N517 025 31/2997 and supported by European Network of Excellence EuroFGI (Future Generation Internet).

This Paper was presented as part at the Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN) 2007

Joanna Domańska, Tadeusz Czachórski are with Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Baltycka 5, 44-100 Gliwice, Poland (Email: joanna,tadek@iitis.gliwice.pl)

Adam Domański is with Institute of Informatics, Silesian Technical University, Akademicka 16, 44-100 Gliwice, Poland (Email: adam.domanski@polsl.pl)

II. Active queue management

In *passive* queue management, packets coming to a buffer are rejected only if there is no space in the buffer to store them, hence the senders have no earlier warning on the danger of growing congestion. In this case all packets coming during saturation of the buffer are lost. The existing schemes may differ on the choice of packet to be deleted (end of the tail, head of the tail, random). During a saturation period all connections are affected and all react in the same way, hence they become synchronised. To enhance the throughput and fairness of the link sharing, also to eliminate the synchronisation, the Internet Engineering Task Force (IETF) recommends *active* algorithms of buffer management. They incorporate mechanisms of preventive packet dropping when there is still place to store some packets, to advertise that the queue is growing and the danger of congestion is ahead. The probability of packet rejection is growing together with the level of congestion. The packets are dropped randomly, hence only chosen users are notified and the global synchronisation of connections is avoided. A detailed discussion of the active queue management goals may be found in [1].

The RED (Random Early Detection) algorithm was proposed by IETF to enhance the transmission via IP routers. It was primarily described by Sally Floyd and Van Jacobson in [4]. Its performance is based on a drop function giving probability that a packet is rejected. The argument *avg* of this function is a weighted moving average queue length, acting as a low-pass filter and calculated at the arrival of each packet as

$$avg = (1 - w)avg' + wq$$

where avg' is the previous value of avg, q is the current queue length and w is a weight determining the importance of the instantaneous queue length, typically $w \ll 1$. If w is too small, the reaction on arising congestion is too slow, if w is too large, the algorithm is too sensitive on ephemeral changes of the queue (noise). Articles [4], [5] recommend $w = 0.001$ or $w = 0.002$, and [6] shows the efficiency of $w = 0.05$ and $w = 0.07$. Article [7] analyses the influence of w on queueing time fluctuations, obviously the larger w, the higher fluctuations. In RED drop function there are two thresholds Min_{th} and Max_{th}. If $avg < Min_{th}$ all packets are admitted, if $Min_{th} < avg < Max_{th}$ then dropping probability p is growing linearly from 0 to p_{max} :

$$p = p_{max}\frac{avg - Min_{th}}{Max_{th} - Min_{th}}$$

and if $avg > Max_{th}$ then all packets are dropped. The value of p_{max} has also a strong influence on the RED performance: if it is too large, the overall throughput is unnecessarily choked and if it's too small the danger of synchronisation arises; [5] recommends $p_{max} = 0.1$. The problem of the choice of parameters is still discussed, see e.g. [8], [9]. The mean avg may be also determined in other way, see [10] for discussion. Despite of evident highlights, RED has also such drawbacks as low throughput, unfair bandwidth sharing, introduction of variable latency, deterioration of network stability. Therefore numerous propositions of basic algorithms improvements appear, their comparison may be found e.g. in [11].

DSRED (double-slope RED) introduced in [12] and developed in [13] is one of these modifications. Three thresholds K_l, K_m and K_h (usually $K_m = (K_l + K_h)/2$) and parameter γ determine two slopes of the DSRED drop function:

$$p(avg) = \begin{cases} 0 & \text{if} \quad avg < K_l \\ \alpha(avg - K_l) & \text{if} \quad K_l \le avg < K_m \\ 1 - \gamma + \beta(avg - K_m) & \text{if} \quad K_m \le avg < K_h \\ 1 & \text{if} \quad K_h \le avg \le N \end{cases}$$

where

$$\alpha = \frac{2(1-\gamma)}{K_h - K_l}, \qquad \beta = \frac{2\gamma}{K_h - K_l}$$

The double slope function makes the algorithm more elastic (more parameters to fix); gentle at the beginning (for low congestion) drop function enhances throughput and reduces queue waiting times.

In section IV, we present analytical (based on Markov chain) and simulation models of RED and DSRED. We assume either Poisson or self-similar traffic. Because of the difficulty in analyzing RED mathematically [14], RED and DSRED are studied in an open-loop scenario.

III. Self-similarity of network traffic

Measurements and statistical analysis of network traffic, e.g. [15], [16] show that it displays a self-similar character. It is observed on various protocol layers and in different network structures. Self-similarity of a process means that the change of time scales does not affect the statistical characteristics of the process. It results in long-range dependence and makes possible the occurrence of very long periods of high (or low) traffic intensity. These features have a great impact on a network performance. They enlarge the mean queue lengths at buffers and increase the probability of packet losses, reducing this way the quality of services provided by a network. Also TCP/IP traffic is characterised by burstiness and long-term correlation, [17], its features are additionally influenced by the performance of congestion avoidance and congestion management mechanisms, [18], [19].

Let a process X_k represent the traffic intensity measured in fixed time intervals and let the aggregated process $X_k^{(m)}$ be the average of the basic process over a group of m consecutive samples: $X_k^{(m)} = \frac{1}{m}(X_{k\cdot m-m+1} + ... + X_{k\cdot m})$, where $k \ge 1$. There are several methods used to check if a process is self-similar. The easiest one is a visual test: one can observe the behaviour of the basic process X_t and the aggregated process $X_k^{(m)}$. If these processes have the same character - the increase of m does not smooth the process, the process is self-similar. More formally, the difference between short-range dependent and long-range dependent (self-similar) process is as follows [16]: for the first process the sum of covariance $\sum_{k=0}^{k=\infty} cov(k)$ is convergent, the spectrum of the process $S(\omega) = \sum_{k=-\infty}^{k=\infty} R(k)e^{-j\omega k}$, where $R(k)$ is the autocorrelation function of the process, is finite at $\omega = 0$, and the variance $var(X_k^{(m)})$ tends asymptotically for large m to the function $\frac{var(X)}{m}$. In the case of long-range dependent process, the sum of covariance $\sum_{k=0}^{k=\infty} cov(k)$ is divergent, $S(0)$ is singular, and $var(X_k^{(m)})$ tends asymptotically to $\frac{var(X)}{m^{\beta}}$, where $0 < \beta < 1$. The parameter β is related to the Hurst parameter H (often used to characterise the self-similarity of a process): $H = 1 - \frac{\beta}{2}$ [16]. For $0.5 < H \le 1$ process is self-similar; the closer H is to 1, the greater is the degree of persistence of long-range dependence.

To represent the self-similar traffic we use here a model introduced by S. Robert [20], [21]. The time of the model is discrete and divided into unit length slots. Only one packet can arrive during each time-slot. In the case of memoryless, geometrical source, the packet comes into system with fixed probability α_1. In the case of self-similar traffic, packet arrivals are determined by a n-state discrete time Markov chain called modulator. It was assumed that modulator has $n = 5$ states ($i = 0, 1, \ldots 4$) and packets arrive only when the modulator is in state $i = 0$. The elements of the modulator transition probability matrix depend only on two parameters: q and a – therefore only two parameters should be fitted to match the mean value and Hurst parameter of the process. If p_{ij} denotes the modulator transition probability from state i to state j, then it was assumed that $p_{0j} = 1/a^j$, $p_{j0} = (q/a)^j$, $p_{jj} = 1 - (q/a)^j$ where $j = 1, \ldots, 4$, $p_{00} = 1 - 1/a - \ldots - 1/a^4$, and remaining probabilities are equal to zero. The passages from the state 0 to one of other states determine the process behaviour on one time scale, hence the number of these states corresponds to the number of time-scales where the process may by considered as self-similar.

The model was fitted to real data [22]. This model enables us to represent, with the use of few parameters, a network traffic which is self-similar over several time-scales.

IV. Influence of choice which packet to drop on RED/DSRED mechanism - analytic and simulation results

The RED or DSRED queue mechanisms are represented by a single-server model based either on discrete-time Markov chain or simulation. The service time represents the time of a packet treatment and dispatching. Its distribution is geometric. The model of incoming traffic was presented above. For both considered in comparisons cases, i.e. for geometric interarrival time distribution (which corresponds to Poisson traffic in case of continuous time models) and self-similar traffic, the considered traffic intensities are the same. A detailed discussion of the choice of model parameters is also presented in [23].

In Markov model, the Markov chain state is defined by the number of packets in the queue, the integer part of the *avg* value and by four flags u_1, u_2, u_3, u_4 approximating the rest of this value (as *avg* is a real number, it is impossible to attribute a state to each of the infinite number of its possible values) in the following way:

$$\frac{[(i-1)*0.25]+(i*0.25)}{2}$$

where i is the number of non-zero flag. If all flags are null, we assume the integer value of *avg*. In case of self-similar traffic this state definition is supplemented by a variable denoting the state of the modulator.

The vector p of state probabilities is given by a system of linear equations

$$p = p * P$$

where P is the transition probability matrix which is generally large (the number of states, hence the order of the matrix P may be hundreds of thousands or millions), sparse and ill conditioned, and the use of well known and broadly used numerical algorithms for algebraic and differential equation systems gives poor results. That is why a projection method using Krylov subspaces, as recommended in [24] was chosen.

The method of Arnoldi is an orthogonal projection process onto the Krylov subspace. It may be used to compute approximations to the unit eigenvalue and the corresponding eigenvector of the matrix P. The matrix H_m (upper Hessenberg matrix) represents the restriction of the linear transformation P to the subspace K_m. Approximation of the eigenvalue of P can be obtained from the eigenvalue of H_m. We often use the so-called Rayleigh-Ritz procedure for extracting eigenvalue and eigenvector approximations from a given subspace. If λ_i is an eigenvalue of H_m and p_i the corresponding eigenvector, i.e.,

$$H_m p_i = \lambda_i p_i,$$

then λ_i is taken as an approximation to an eigenvalue of P, and $V_m p_i$ as an approximation to the corresponding eigenvector of P.

To simplify the notation, we denote: $v = p(t_i)$, and $w = p(t_{i+1})$. The solution should have the form $w = e^P v$ (for simplicity, we omit here the constant $\tau_i = t_{i+1} - t_i$). Following the observation that a truncated series of order $m-1$ (or, more generally, that approximating $e^P v$ by a polynomial of degree $m-1$, notice that this polynomial is a linear combination of the vectors $v, Pv, ..., P^{m-1}v$), will yield an element of the Krylov subspace:

$$K_m(P, v) \equiv Span\{v, Pv, ..., P^{m-1}v\}$$

The method is reduced to find such as element $K_m(P, v)$ of this space, that best approximates $w = e^P v$. The set of base vectors of these subspace is denoted by $V_m = [v_1, v_2, ..., v_m]$, $v_1 = v/\beta$ where $\beta = \| v \|_2$, $v = \beta V_m e_1$ and:

$$w \approx V_m \left[(V_m^T V_m)^{-1} V_m^T e^P V_m\right] \beta e_1 \tag{1}$$

In vector e_i the *i*-th element is equal 1 and the others are null. The set of base vectors V_m is obtained via Arnoldi's procedure [24]:

```
1. $v_1 = v / \| v \|_2$
2. For j=1,2,...,m do
        $z = Pv_j$
        For $i = 1, 2, ..., j$ do
            $h_{ij} = v_i^T z$
            $z = z - h_{ij} v_i$
        $h_{j+1,j} = \| z \|_2$
        $v_{j+1} = z / h_{j+1,j}$
```

The above algorithm is the modified Gram-Schmidt orthogonalization procedure, the obtained vectors v_i are orthonormal, and the upper Hessenberg matrix H_m (its dimension is $m \times m$) which is composed of coefficients h_{ij} holds the equation:

$$P V_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T \tag{2}$$

The set of vectors V_m is orthonormal, hence we can simplify the eq. (1):

$$w \approx \beta V_m (V_m^T e^P V_m) e_1$$

If we approximate $V_m^T e^P V_m$ by $e^{V_m^T P V_m}$ the sought vector w will become:

$$w \approx \beta V_m e^{V_m^T P V_m} e_1$$

Also, because the set of vectors is orthonormal V_m we may rewrite (1) as:

$$H_m = V_m^T P V_m \tag{3}$$

and the solution may be expressed as:

$$w \approx \beta V_m e^{H_m} e_1$$

This solution still needs the calculation of matrix exponential but the size of the matrix is considerably smaller (m - the dimension of Krylov subspace is significantly smaller than n - the number of states of the considered system). Hence we can use any method advised for small systems. e.g. Padé approximation.

In the above description the constant τ was omitted. It may be easily put to the obtained solution because $V_m^T(P\tau)V_m = H_m$, and Krylov subspaces related to P and $P\tau$ are indentical.

Hence, the use of the Krylov subspaces for transient states consists in:

- the use of Arnoldi procedure to obtain the orthonormal set of base vectors V_m and Hessenberg matrix H_m.
- the use of Padé approximation to obtain $e^{H_m \tau}$.
- calculation of the state probability vector approximated by $\beta V_m e^{H_m \tau} e_1$.

To validate the Markovian results, we used a simulation packet OMNET++, written in C++ by A. Varga [http://www.omnetpp.org/]. Below we present some numerical results.

Our goal is to capture the influence of the way a packet is chosen to be deleted (end of the tail, head of the tail) on the RED and DSRED queueing times. Input traffic intensity (for geometric and self-similar traffic) was chosen as $\alpha = 0.5$, and due to the modulator characteristics, the Hurst parameter of self-similar traffic was fixed to $H = 0.78$.

The RED parameters had the following values: buffer size 250 packets, threshold values $Min_{th} = 100$ and $Max_{th} =$

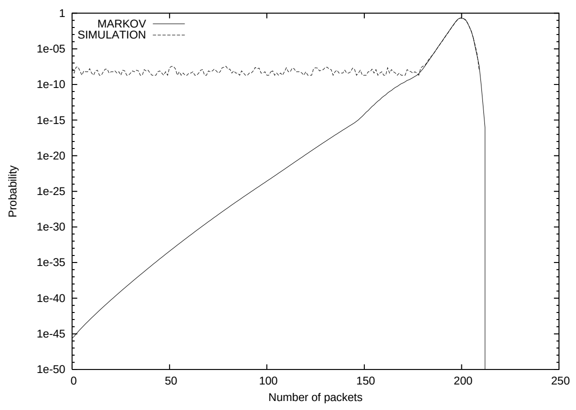

Fig. 1. Queue distribution for RED queue: geometric source, $\alpha = 0.5$, $\mu = 0.25$, $w = 0.07$, analytic and simulation results.

200, $p_{max} = 0.1$, $w = 0.002$ or $w = 0.07$. Parameter μ of geometric distribution of service times (probability of the end of service within a current time-slot) was $\mu = 0.25$ or $\mu = 0.5$. Due to the changes of μ, two different traffic loads (low and high) were considered.

In case of DSRED policy, the traffic pattern and the buffer size are the same, parameters $K_l = Min_{th} = 100$ and $K_h = Max_{th} = 200$, intermediate threshold $K_m = 150$. The shaping parameter γ had three values $\gamma = 0.15, 0.5, 0.85$.

Fig. 1 displays a comparison of analytical and simulation results. They are almost identical if probabilities are greater then 10^{-10}, for smaller values the simulation results are not significant (the simulation run involved 250 millions of packets) while Markov model is able to give probabilities of very rare events.

If the mean queue length is relatively low, the influence of dropping scheme on queueing time is negligible: the introduction of drop-from-front strategy gives 0.7% shorter mean queueing time in case of RED and 0.8% shorter mean queueing time in case of DSRED, see Fig. 2.

Naturally, the introduction of DSRED gives shorter mean queue length and shorter mean queueing time compared to RED. However, when the Poisson traffic is replaced by self-similar one with the same intensity and preserving the same parameters of RED, the length of the queue grows and the influence of the dropping scheme is more visible: drop-from-front strategy reduces mean queueing time by 16.4%. A comparison of response time distributions for RED queue, for both strategies is presented in Fig. 3 (left). The same comparison in case of DSRED queue is presented in Fig. 3 (right). In this case the response time with drop-from-front strategy is 18.1% shorter then for tail-drop mechanism.

The change of w_q value (from 0.07 to 0.002) in computation of moving average results in longer response time and mean queue – see the Tables – but the introduction of drop-from-front in place of tail-drop gives about 1% of changes. A comparison of queueing time distributions in these cases is given in Fig. 4 (left - RED) and (right - DSRED).

In case of heavy traffic, for both mechanisms RED/DSRED, irrespective of the w_q value and of the traffic self-similarity, drop-from-front strategy gives two times shorter mean queue-

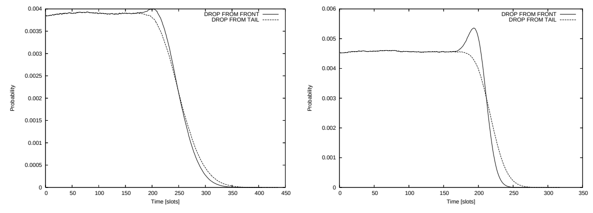

Fig. 2. Waiting times for RED (left) DSRED (right) queues: drop-from-front and drop-from-tail strategies, geometric source, $\alpha = 0.5$, $\mu = 0.5$, $w = 0.07$, $\gamma = 0.5$.

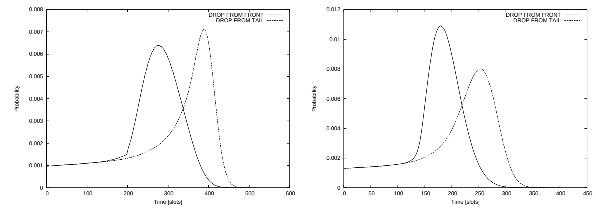

Fig. 3. Waiting times for RED (left) and DSRED (right) queues: drop-from-front and drop-from-tail strategies, self-similar source, $\alpha = 0.5$, $\mu = 0.5$, $w = 0.07$, $\gamma = 0.5$.

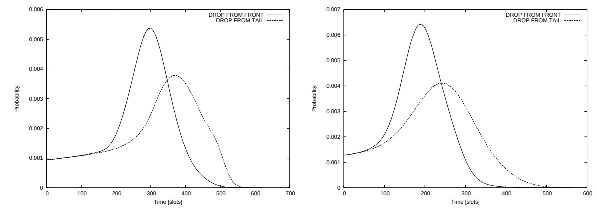

Fig. 4. Waiting times for RED (left) and DSRED (right) queues: drop-from-front and drop-from-tail strategies, self-similar source, $\alpha = 0.5$, $\mu = 0.5$, $w = 0.002$, $\gamma = 0.5$.

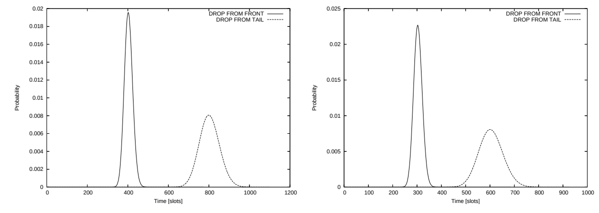

Fig. 5. Waiting times for RED (left) and DSRED (right) queues: drop-from-front and drop-from-tail strategies, geometric source, $\alpha = 0.5$, $\mu = 0.25$, $w = 0.07$, $\gamma = 0.5$.

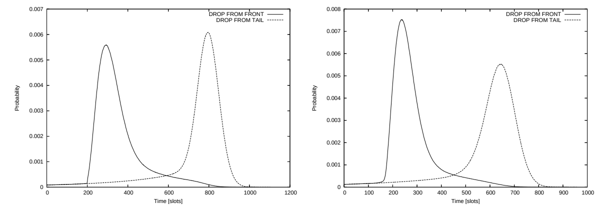

Fig. 6. Waiting times for RED (left) and DSRED (right) queues: drop-from-front and drop-from-tail strategies, self-similar source, $\alpha = 0.5$, $\mu = 0.25$, $w = 0.07$, $\gamma = 0.5$.

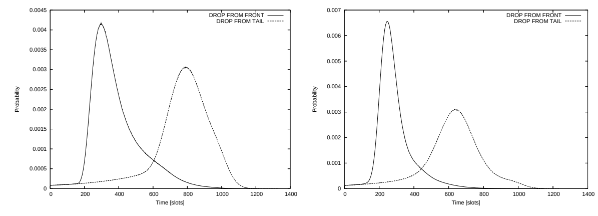

Fig. 7. Waiting times for RED (left) and DSRED (right) queues: drop-from-front and drop-from-tail strategies, self-similar source, $\alpha = 0.5$, $\mu = 0.25$, $w = 0.002$, $\gamma = 0.5$.

TABLE I
COMPARISON OF RED AND DSRED

		Mean queue length	Variation of queue length
RED $\mu = 0.5$ $w = 0.002$	GEO	64.92	1562.07
	SELF-S	130.61	6484.46
DSRED $\gamma = 0.5$ $\mu = 0.5$ $w = 0.002$	GEO	54.65	1077.13
	SELF-S	89.53	3703.73
RED $\mu = 0.25$ $w = 0.002$	GEO	199.84	82.33
	SELF-S	169.86	5056.31
DSRED $\gamma = 0.5$ $\mu = 0.25$ $w = 0.002$	GEO	150.01	131.036
	SELF-S	136.21	3962.95
RED $\mu = 0.5$ $w = 0.07$	GEO	64.43	1504.8
	SELF-S	123.79	5570.37
DSRED $\gamma = 0.15$ $\mu = 0.5$ $w = 0.07$	GEO	53.07	977.58
	SELF-S	76.1	2202.89
DSRED $\gamma = 0.85$ $\mu = 0.5$ $w = 0.07$	GEO	54.87	1018.47
	SELF-S	83.01	2628.08
DSRED $\gamma = 0.85$ $\mu = 0.5$ $w = 0.07$	GEO	57.91	1182.4
	SELF-S	100.03	3731.86
RED $\mu = 0.25$ $w = 0.07$	GEO	199.75	3.47
	SELF-S	163.2	4497.22
DSRED $\gamma = 0.15$ $\mu = 0.25$ $w = 0.07$	GEO	129.41	24.57
	SELF-S	109.52	2313.9
DSRED $\gamma = 0.85$ $\mu = 0.25$ $w = 0.07$	GEO	150	40.05
	SELF-S	130.26	3100.71
DSRED $\gamma = 0.85$ $\mu = 0.25$ $w = 0.07$	GEO	170.59	24.58
	SELF-S	144.43	3642.48

TABLE II
COMPARISON OF RED AND DSRED

		Loss probability
RED $\mu = 0.5$ $w = 0.002$	GEO	0.00389652
	SELF-S	0.150939 (0.135347)
DSRED $\gamma = 0.5$ $\mu = 0.5$ $w = 0.002$	GEO	0.00455001
	SELF-S	0.168627 (0.168509)
RED $\mu = 0.25$ $w = 0.002$	GEO	0.500013
	SELF-S	0.551741 (0.507265)
DSRED $\gamma = 0.5$ $\mu = 0.25$ $w = 0.002$	GEO	0.500066
	SELF-S	0.55552 (0.548233)
RED $\mu = 0.5$ $w = 0.07$	GEO	0.00390818
	SELF-S	0.151645
DSRED $\gamma = 0.15$ $\mu = 0.5$ $w = 0.07$	GEO	0.004675
	SELF-S	0.175138
DSRED $\gamma = 0.85$ $\mu = 0.5$ $w = 0.07$	GEO	0.00457423
	SELF-S	0.170974
DSRED $\gamma = 0.85$ $\mu = 0.5$ $w = 0.07$	GEO	0.00426957
	SELF-S	0.162186
RED $\mu = 0.25$ $w = 0.07$	GEO	0.500006
	SELF-S	0.55187
DSRED $\gamma = 0.15$ $\mu = 0.25$ $w = 0.07$	GEO	0.499999
	SELF-S	0.559453
DSRED $\gamma = 0.85$ $\mu = 0.25$ $w = 0.07$	GEO	0.499999
	SELF-S	0.555498
DSRED $\gamma = 0.85$ $\mu = 0.25$ $w = 0.07$	GEO	0.499999
	SELF-S	0.554022

ing times. Queueing time distributions for all considered cases are presented in Figs. 5, 6, 7.

V. AQM MECHANISMS IN LINUX BASED ROUTERS

We implemented the most popular RED algorithms : RED, FRED, DSRED and ARED in a Linux based router. Our solution was based on user defined queues in Iptables Linux firewall. The behaviour of RED's algorithms were observed in three situations:

- connection oriented TCP streams,
- datagram oriented UDP packets,
- mixed flows containing TCP streams and UDP packets.

In the case of TCP traffic the bahaviour of the both methods (drop-from-front and drop-from-tail) is similar and the choice of packet drop strategy is insignificant, see fig. 8, 9. The amount of packets dropped by RED mechanism is low and the situation is supervised by conguestion avoidance mechanisms built in TCP protocol.

For UDP traffic and mixed traffic (UDP and TCP together) the number of dropped packets is greater than that in the TCP case. In the considered example the RED mechanism drops 75 percent of UDP traffic. In these cases there was also no improvement of results when using Drop-From-Front strategy

The differences between analytical and experimental results may be explained by two factors. First, the variability of generated traffic used in experiments was lower than one may expect in a real network. Second, the implemented drop-from-front strategy program has to make additional operations to find the first packet in the queue. In our router this operation is relatively slow due to its execution in user space (not in kernel space) thus the results are worse than that for drop-from-tail strategy.

Fig. 11 presenting results for FRED algorithm. In the implementation of that AQM algorithm a significant advantage of Drop-From-Front mechanism can be noticed. Computational complexity of FRED algorithm lowers the significance of time overhead caused by searching for the first packet in the queue.

VI. CONCLUSIONS

Drop-from-front strategy, when applied in place of tail-drop one, results in reduction of mean queueing time in RED/DSRED mechanisms of active queue management. In case of light load, the difference is more visible for self-similar traffic. In case of heavy load, the difference is also substantial for short-dependent traffic.

This article has presented the implementation of AQM mechanisms in Linux operating system. Iptables have been used in order to manage streams of data. Such an assumption has facilitated the implementation of algorithms but, on the other hand, it has caused a relatively slower work of the router. When dropping packets from head of queue the efficiency of RED mechanisms has been decreased by the necessity of finding packets to drop from queue.

TABLE III
COMPARISON OF RED AND DSRED

		Mean waiting time	Variance of waiting time
RED $\mu = 0.5$ $w = 0.002$	GEO	132.34	6340.05
	SELF-S	308.49	16866.7
DSRED $\gamma = 0.5$ $\mu = 0.5$ $w = 0.002$	GEO	111.8	4385.54
	SELF-S	218.26	10063.2
RED $\mu = 0.25$ $w = 0.002$	GEO	803.35	3731.58
	SELF-S	760.58	34059.4
DSRED $\gamma = 0.5$ $\mu = 0.25$ $w = 0.002$	GEO	604.09	3910.51
	SELF-S	617.071	31355
RED $\mu = 0.5$ $w = 0.07$	GEO	131.375	6109.14
	SELF-S	293.89	13634.1
DSRED $\gamma = 0.15$ $\mu = 0.5$ $w = 0.07$	GEO	108.63	3985.63
	SELF-S	186.39	4924.46
DSRED $\gamma = 0.85$ $\mu = 0.5$ $w = 0.07$	GEO	110.67	4150.01
	SELF-S	202.25	6025.25
DSRED $\gamma = 0.85$ $\mu = 0.5$ $w = 0.07$	GEO	118.314	4811.09
	SELF-S	240.73	8821.86
RED $\mu = 0.25$ $w = 0.07$	GEO	803	2467.85
	SELF-S	735.69	25134.7
DSRED $\gamma = 0.15$ $\mu = 0.25$ $w = 0.07$	GEO	521.63	1958.88
	SELF-S	501.39	14030.4
DSRED $\gamma = 0.85$ $\mu = 0.25$ $w = 0.07$	GEO	603.98	2554.41
	SELF-S	590.9	18972.3
DSRED $\gamma = 0.85$ $\mu = 0.25$ $w = 0.07$	GEO	686.33	2453.79
	SELF-S	652.57	21408.3

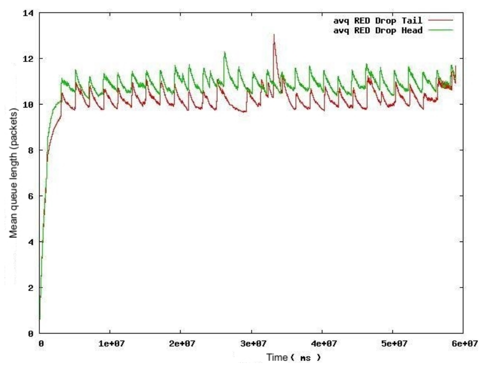

Fig. 8. Mean queue length for RED algorithm for TCP traffic

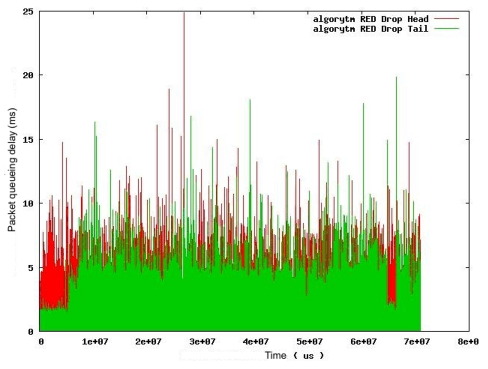

Fig. 9. Packet queueing delay for RED algorihm and TCP traffic

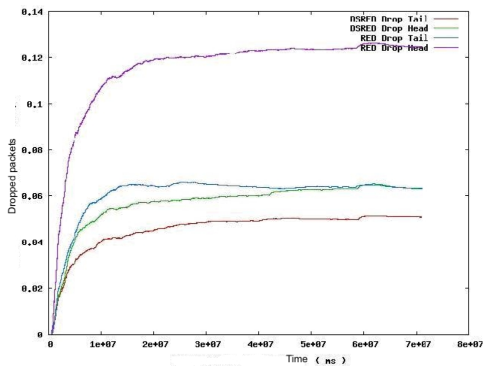

Fig. 10. Amount of dropped packets for RED and DSRED algorithm and TCP traffic

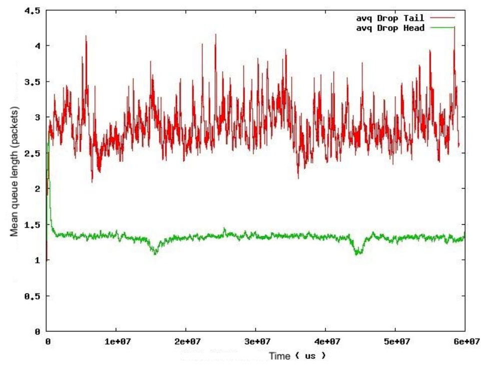

Fig. 11. Mean queue length for FRED algorithm for UDP traffic

Therefore for RED and DSRED mechanisms and strategy of dropping packets from head of queue the decrease of average time of data traversal through router has not been obtained.

The desired effect has been acquired only for FRED algorithm. Its complexity causes the time of finding packets to become not very significant.

The authors of the article expect that the implementation of AQM algorithms in kernel space will increase the efficiency of the router's work and at the same time will corroborate the results obtained using analytical methods.

REFERENCES

[1] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on queue management and congestion avoidance in the internet," *RFC 2309, IETF*, 1998.

[2] S. Floyd, "Red with drop from front," ftp://ftp.ee.lbl.gov/email/sf.98mar11.txt, 1998.

[3] J. Domańska, A. Domański, and T. Czachórski, "The drop-from-front strategy in aqm," *Lecture Notes in Computer Science*, vol. 4712, pp. 61–72, 2007.

[4] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[5] S. Floyd, "Discussions of setting parameters," http:// www.icir.org/ floyd/ REDparameters.txt, 1997.

[6] B. Zheng and M. Atiquzzaman, "A framework to determine the optimal weight parameter of red in next generation internet routers," The University of Dayton, Department of Electrical and Computer Engineering, Tech. Rep., 2000.

[7] M. May, T. Bonald, and J. Bolot, "Analytic evaluation of red performance," *IEEE Infocom 2000, Tel-Aviv, Izrael*, 2000.

[8] W. Chang Feng, D. Kandlur, and D. Saha, “Adaptive packet marking for maintaining end to end throughput in a differentiated service internet,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, pp. 685–697, 1999.

[9] M. May, C. Diot, B. Lyles, and J. Bolot, “Influence of active queue management parameters on aggregate traffic performance,” Research Report, Institut de Recherche en Informatique et en Automatique, Tech. Rep., 2000.

[10] B. Zheng and M. Atiquzzaman, “Low pass filter/over drop avoidance (lpf/oda): An algorithm to improve the response time of red gateways,” *Int. Journal of Communication Systems*, vol. 15, no. 10, pp. 899–906, 2002.

[11] M. Hassan and R. Jain, *High Performance TCP/IP Networking*. Pearson Education Inc., 2004.

[12] B. Zheng and M. Atiquzzaman, “Dsred: A new queue management scheme for next generation networks,” *The 25th Annual IEEE Conference on Local Computer Networks*, pp. 242–251, 2000.

[13] ——, “Improving performance of active queue management over heterogeneous networks,” *ICC 2001: International Conference on Communications*, pp. 2375–2379, 2001.

[14] C. Liu and R. Jain, “Improving explicit congestion notification with the mark-front strategy,” *Computer Networks (Amsterdam, Netherlands: 1999)*, vol. 35, no. 2–3, pp. 185–201, 2001. [Online]. Available: citeseer.ist.psu.edu/liu00improving.html

[15] W. Stallings, *High Speed Networks, TCP/IP and ATM Design Principles*. Upper Saddle River, N.J: Prentice Hall, 1998.

[16] W. Willinger, W. E. Leland, and M. S. Taqqu, “On the self-similar nature of ethernet traffic,” *IEEE/ACM Transactions on Networking*, February 1994.

[17] P. Abry, P. Flandrin, M. Taqqu, and D. Veitch, *Self-similar Network Traffic Analysis and Performance Evaluation*. K. Park i W. Willinger (eds), 1999, ch. Wavelets for the analysis, estimation and synthesis of scaling data.

[18] V. Paxson and S. Floyd, “Wide area traffic: the failure of poisson modeling,” *IEEE/ACM Transactions on Networking*, vol. 3, 1995.

[19] A. Feldman, A. Gilbert, P. Huang, and W. Willinger, “Dynamics of ip traffic: Study of the role of variability and the impact of control,” *ACM SIGCOMM’99, Cambridge*, 1999.

[20] S. Robert, “Modélisation markovienne du trafic dans les réseaux de communication,” Ph.D. dissertation, Ecole Polytechnique Fédérale de Lausanne, 1996, nr 1479.

[21] S. Robert and J.-Y. L. Boudec, “New models for pseudo self-similar traffic,” *Performance Evaluation*, vol. 30, no. 1-2, pp. 57–68, 1997.

[22] T. Czachórski and J. Domańska, “Markovian models for long-range dependent traffic,” *Archiwum Informatyki Teoretycznej I Stosowanej*, vol. 13, no. 3, pp. 297–308, 2001.

[23] J. Domańska, “Procesy markowa w modelowaniu nateżenia ruchu w sieciach komputerowych,” Ph.D. dissertation, IITiS PAN, Gliwice, 2005.

[24] W. Stewart, *An Introduction to the Numerical Solution of Markov Chains*. Princeton Academic Press, 1994.