Int. J. Appl. Math. Comput. Sci., 2011, Vol. 21, No. 3, 567–577
DOI: 10.2478/v10006-011-0045-7

amcs

ANALYSIS OF AQM QUEUES WITH QUEUE SIZE BASED PACKET DROPPING

ANDRZEJ CHYDZIŃSKI, ŁUKASZ CHRÓST

Institute of Informatics
Silesian University of Technology, Akademicka 16, 44–100 Gliwice, Poland
e-mail: andrzej.chydzinski@polsl.pl

Queueing systems in which an arriving job is blocked and lost with a probability that depends on the queue size are studied. The study is motivated by the popularity of Active Queue Management (AQM) algorithms proposed for packet queueing in Internet routers. AQM algorithms often exploit the idea of queue-size based packet dropping. The main results include analytical solutions for queue size distribution, loss ratio and throughput. The analytical results are illustrated via numerical examples that include some commonly used blocking probabilities (dropping functions).

Keywords: single-server queue, packet dropping, queue size distribution, active queue management.

1. Introduction

Active Queue Management (AQM) algorithms designed for Internet routers have been studied widely since the paper by Floyd and Jacobson (1993) was published (see, e.g., Athuraliya *et al.*, 2001; Fatta *et al.*, 2003; Feng *et al.*, 2002; Floyd *et al.*, 2001; Hollot *et al.*, 2002; Kunniyur and Srikant, 2004; Lakshmikantha *et al.*, 2005; Sun and Zukerman, 2007, and the references given therein). The idea of AQM is that the router can reject an incoming packet even if it is capable of storing it in its buffer. In other words, the classic drop-tail rejection discipline, where packets are dropped only when the buffer is overflown, is replaced by some advanced rejection discipline. Typically, an arriving packet is blocked randomly with a probability that depends on the current or past system state. In practice, this blocking probability often depends on the queue size. It may also depend on other characteristics, like packet loss history, the arrival process parameters, the frequency of empty buffer and full buffer events, and others. The main goal of AQM in Internet routers is preventing queues from growing too long, while at the same time maintaining high link utilizations. There are also other objectives of AQM, like stable, predictable work (low variance of the queue size), desynchronization of TCP sources, fair bandwidth division, and others (for more details, see, for instance, Bohacek *et al.*, 2004).

In this paper we carry out an analysis of the queueing system in which an arriving packet is dropped with a probability that is a function of the queue size observed upon arrival (this function will be called a *dropping function*). In the AQM literature, several types of dropping functions are used, for instance, the linear dropping function (RED algorithm (Floyd and Jacobson, 1993)), the exponential dropping function (REM algorithm (Athuraliya *et al.*, 2001)), the doubly linear dropping function (GRED (Rosolen *et al.*, 1999)), etc. We do not assume here a particular dropping function—it can be in any form.

The main contribution of this paper is to provide formulas for the queue size distribution (at an arbitrary time) and the overall loss ratio in the system with the dropping function in a general form. In addition to that, the queue size distribution at departure epochs and transition probabilities for the imbedded Markov chain are obtained in intermediate steps. The analysis is conducted assuming a Poisson arrival process and the general type of the service time distribution.

To the best of the authors' knowledge, the results presented here are new. Although the literature devoted to AQM is wide (for surveys of the most popular AQM algorithms, see, for instance, Bohacek *et al.*, 2004; Chatranon *et al.*, 2004), there are very few papers in which a classic approach, based on queueing theory, is used to analyze AQM mechanisms. In most of the papers, the simulation results obtained by means of discrete-event simulators (e.g., *Network Simulator ns-2*, 2011; *Opnet*, 2011) are used for performance evaluation purposes. This is perfectly understandable, as complex behaviour of the TCP protocol has to be simulated accurately if a real networking application is considered. However, simulation-based eva-

luation lacks some fundamental insight into the AQM mechanism.

Only two papers, that by Bonald *et al*. (2000) as well as the one by Hao and Wei (2005), come close to our approach. In the former, a model of the RED queue with batch Poisson arrivals is proposed and analyzed assuming the exponential service time distribution. It is also assumed that the RED router uses the same dropping probability, d_n, for every packet in an arriving batch, where n is the queue size observed at the time the first packet in the batch arrives at the router. The latter assumption does not introduce a large error only if the dropping function is very smooth (small Δd_n). Therefore, many interesting classes of dropping functions are in fact excluded from the analysis. It is also not explained how the stationary distribution, π, for the Markov chain can be derived—only a note that it differs from the stationary distribution for the drop-tail algorithm can be found.

The analysis presented in this paper is more comprehensive. We use a general form of the service time distribution (a more realistic model of packet processing time in an Internet router), a general form of the dropping function, and no approximations are used—all the results are strict. As the service time is not memoryless herein, the queue size process is not Markovian any more. This forced application of different, more advanced analytical techniques.

Hao and Wei (2005) study a queue-size based packet dropping mechanism using an extension of the $GI^X/M/1$ queue. A technique based on the thinning of the arrival process is applied[1]. In the beginning, the whole space of possible queue sizes is divided into $m+1$ subintervals: $[0, L_1), [L_1, L_2), \ldots, [L_m, b]$. In each interval $[L_i, L_{i+1})$, the dropping probability is constant and equal to d_i. Then, the thinned arrival process is introduced via interarrival time densities $A_i(t)$, where the index i means that the queue size is in the range $[L_i, L_{i+1})$. Finally, the classic methodology for analysis of the $GI^X/M/1$ queue is applied (however, the details are not presented). There are two issues with applicability of the results presented by Hao and Wei (2005). In order to obtain numerical results, we have to use the distributions $A_i(t)$, which are computed approximately and for subexponential interarrival times only. Moreover, the classic methodology for analysis of the $GI^X/M/1$ queue exploits an assumption that the interarrival times are independent. In the thinned arrival process, the interarrival times can be strongly dependent. Herein, we use a different queueing model and different analytical approach than those presented by Hao and Wei (2005).

We believe that the results presented in this paper are of practical importance not only due to their AQM and networking origins, but also because a queue-size based dropping policy can be used in many other applications. This is due to the fact that manipulating the shape of the dropping function allows us to control the average queue size and the loss ratio in the system. These possibilities will be demonstrated in numerical examples.

The paper is structured in the following way. First, the model of the queueing system is formally described (Section 2). Then, in Section 3, the queue length at departure epochs is studied using the embedded Markov chain technique. The main result of this section is Theorem 1, which gives the transition probabilities of the embedded Markov chain. Furthermore, detailed calculations for the RED dropping function are given. The major part of the paper then follows, presenting a formula for queue size distribution at an arbitrary time (Section 4) and a formula for the overall loss ratio in the system (Section 5). In Section 6, numerical examples are presented. In particular, four different dropping functions are used to obtain the queue sizes and loss ratios in three congestion scenarios. In addition to the numbers obtained by means of the analytical formulas, some simulation results are shown. Finally, remarks concluding the paper are gathered in Section 7.

2. Queueing model

In this article we deal with a single server queue fed by a Poisson process with rate λ. The service time is distributed according to a distribution function $F(\cdot)$, which is not further specified, and the standard independence assumptions are made. A job arriving at the system is blocked and lost with probability d_n, where n is the queue size (including service position) at the arrival time of this job. It is assumed that the buffer size is finite and the maximum number of jobs in the system is b. Equivalently, this assumption can be rewritten as

$$d_n = 1 \quad \text{for} \quad n \geq b. \tag{1}$$

We assume also that the time origin corresponds to a departure epoch. The dropping function d_n is not further specified and can have any form.

We propose denoting this queueing system by $M/G/1/b(AQM)$, which comes from Kendall's notation for a classic finite-buffer queue, i.e., $M/G/1/b$.

The finite buffer is assumed for two reasons. Firstly, this assumption reflects the fact that in network devices the buffering space for packet storage is also limited. Secondly, a limited queue size guarantees the existence of the steady-state distribution of the queue size.

Here and subsequently, $\mathbf{P}$ denotes the probability and $X(t)$ denotes the queue size at the moment t, including the service position.

[1] By "thinning" of the arrival process we mean extending the actual interarrival times due to the drop events.

3. Queue size at departure epochs

Let X_n, $n = 1, 2, \ldots$, be the queue size left behind by the n-th departing job. We always have $X_n < b$.

In this section, we are interested in finding the stationary distribution of X_n, namely,

$$\pi_k = \lim_{n\to\infty} \mathbf{P}(X_n = k), \quad 0 \le k \le b-1. \tag{2}$$

As the sequence X_n is an ergodic Markov chain, we can find π_k by solving the following system of equations:

$$\begin{cases} \pi_k = \displaystyle\sum_{j=0}^{b-1} \pi_j p_{j,k}, & 0 \le k \le b-1, \\ \displaystyle\sum_{j=0}^{b-1} \pi_j = 1, \end{cases} \tag{3}$$

where

$$p_{j,k} = \mathbf{P}(X_{n+1} = k | X_n = j), \quad 0 \le j, k \le b-1, \tag{4}$$

are the transition probabilities for chain X_n. Therefore, the task is completed by finding probabilities $p_{j,k}$. To accomplish that, we introduce the following convention: $Q_{n,k}(u)$ denotes the probability that in time interval $(0, u]$ exactly k jobs are let into the system, assuming that $X(0) = n$ and the first departure time is after u.

Note that this is not a trivial thinning of the Poisson process. The first arriving job is blocked with probability d_n, but the second one is blocked either with probability d_{n+1} (if the previous one was accepted) or with probability d_n (if the previous one was blocked). This gets more complicated with subsequent arrivals, according to the acceptance and rejection history.

If we denote by $a_{n,k}$ the probability that k jobs are let into the system during the service time, we have

$$a_{n,k} = \int_0^\infty Q_{n,k}(u)\,\mathrm{d}F(u), \tag{5}$$

and the following representation for $p_{j,k}$ can be obtained:

$$p_{j,k} = \begin{cases} a_{1,k} & \text{if} \quad j = 0, 0 \le k \le b-1, \\ a_{j,k-j+1} & \text{if} \quad 1 \le j \le b-1, \\ & \quad\ j-1 \le k \le b-1, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

We can see now how the analysis of the $M/G/1/b(AQM)$ queue differs from that of the classic $M/G/1/b$ queue (see, for instance, Takagi, 1993, p. 200). According to classic theory, the coefficients (5) depend only on k and do not depend on the initial queue size. What is more, here they have a much more complex form, depending on $Q_{n,k}(u)$. In classic theory, $Q_{n,k}(u)$ reduces to a simple Poisson formula.

If we can compute $Q_{n,k}(u)$, we shall be able to compute probabilities $p_{j,k}$. Therefore, the remaining part of this section is devoted to finding $Q_{n,k}(u)$. Denoting by $q_{n,k}(s)$ the Laplace transform of $Q_{n,k}(u)$, namely,

$$q_{n,k}(s) = \int_0^\infty e^{-su} Q_{n,k}(u)\,\mathrm{d}u, \tag{7}$$

we will now prove the following theorem.

Theorem 1. *In the M/G/1/b(AQM) system the following holds true:*

$$q_{n,k}(s) = \frac{\displaystyle\prod_{i=0}^{k-1} c_{n+i}}{\displaystyle\prod_{i=0}^{k}(s + c_{n+i})}, \qquad n \ge 0, \ \ k \ge 0, \tag{8}$$

where

$$c_n = \lambda(1 - d_n). \tag{9}$$

(In order to simplify the notation, we use a convention stating that $\prod_{i=0}^{-1} = 1$*.)*

Proof. Using the total probability formula with respect to the first arrival time, we obtain the following system of equations for $n \ge 0$, $k > 0$:

$$Q_{n,k}(u) = \int_0^u \lambda e^{-\lambda v} \big(d_n Q_{n,k}(u - v) + (1 - d_n) Q_{n+1,k-1}(u - v) \big)\,\mathrm{d}v. \tag{10}$$

The first summand under the integral corresponds to the case where the first arriving job is dropped and the queue size remains n. The second summand corresponds to the case where the first arriving job is accepted and the new queue size is $n + 1$.

Similarly, using the total probability formula with respect to the first arrival time, for $n \ge 0$, $k = 0$ we get

$$Q_{n,0}(u) = \int_0^u \lambda e^{-\lambda v} d_n Q_{n,0}(u - v)\,\mathrm{d}v + e^{-\lambda u}. \tag{11}$$

The first summand corresponds to the situation where the first arriving job is dropped and the queue size remains empty, while the second summand corresponds to the situation where there are no arrivals by the time u.

Applying Laplace transforms to (10) and (11), we obtain

$$q_{n,k}(s) = d_n q_{n,k}(s) \frac{\lambda}{s + \lambda} + (1 - d_n) q_{n+1,k-1}(s) \frac{\lambda}{s + \lambda}, n \ge 0, k > 0, \tag{12}$$

and

$$q_{n,0}(s) = d_n q_{n,0}(s)\frac{\lambda}{s+\lambda} + \frac{1}{s+\lambda}, \quad n \geq 0. \tag{13}$$

From (12) we obtain then

$$q_{n,k}(s) = \frac{c_n}{s+c_n} q_{n+1,k-1}(s), \quad n \geq 0, k > 0, \tag{14}$$

and from (13)

$$q_{n,0}(s) = \frac{1}{s+c_n}, \quad n \geq 0. \tag{15}$$

Proceeding by induction, it is easy to check that (14) and (15) lead to (8), which completes the proof. ■

Now we have to find $Q_{n,k}(u)$ by inverting transforms $q_{n,k}(s)$. Due to the assumption (1), for $n \geq b$ the formula (8) reduces to a simple form:

$$q_{n,k}(s) = 0, \quad k \geq 1, \tag{16}$$

and

$$q_{n,0}(s) = \frac{1}{s}, \tag{17}$$

which is equivalent to

$$Q_{n,k}(s) = 0, \quad k \geq 1, \tag{18}$$

and

$$Q_{n,0}(s) = 1. \tag{19}$$

Inverting $q_{n,k}(s)$ for $n < b$ is not very difficult either since (8) has a well-known inverse transform. Let us denote by $D_{n,k}(s)$ the denominator in (8), namely,

$$D_{n,k}(s) = \prod_{i=0}^{k}(s + c_{n+i}). \tag{20}$$

First, we have to rewrite (8) in the following form:

$$D_{n,k}(s) = \prod_{j=1}^{L}(s + c_j^*)^{\alpha_j}, \tag{21}$$

where L is the number of different roots of $D_{n,k}(s)$ and $-c_j^*$ is a root of $D_{n,k}(s)$ with multiplicity α_j. Using this notation we have

$$Q_{n,k}(u) = \sum_{j=1}^{L} \mathrm{res}_{s=-c_j^*}\left[\frac{N_{n,k}}{D_{n,k}(s)} e^{su}\right], \tag{22}$$

where $\mathrm{res}_{s=-c_j^*}$ denotes the residue at the point $-c_j^*$ and

$$N_{n,k} = \prod_{i=0}^{k-1} c_{n+i}. \tag{23}$$

The residue in (22) can be easily computed using the following limit:

$$\mathrm{res}_{s=-c_j^*} \frac{N_{n,k}}{D_{n,k}(s)} e^{su} = \frac{1}{(\alpha_j - 1)!} \lim_{s \to -c_j^*} \frac{\mathrm{d}^{\alpha_j - 1}}{\mathrm{d}s^{\alpha_j - 1}} \left[\frac{(s + c_j^*)^{\alpha_j} N_{n,k}}{D_{n,k}(s)} e^{su}\right]. \tag{24}$$

If every $-c_j^*$ has multiplicity 1 (i.e., $\alpha_j = 1$ for every j), then

$$\mathrm{res}_{s=-c_j^*} \frac{N_{n,k}}{D_{n,k}(s)} e^{su} = \frac{N_{n,k}}{D'_{n,k}(-c_j^*)} e^{-c_j^* u} \tag{25}$$

and

$$Q_{n,k}(u) = \sum_{j=1}^{L} \frac{N_{n,k}}{D'_{n,k}(-c_j^*)} e^{-c_j^* u} = \sum_{i=0}^{k} \frac{N_{n,k}}{D'_{n,k}(-c_{n+i})} e^{-c_{n+i} u}. \tag{26}$$

When applying the formulas (21)–(26), the main task is to separate different roots of $D_{n,k}(s)$ and find their multiplicities. The tediousness of this work depends on the form of the dropping function. Injective functions are easiest while functions with several flat parts (steps) make the work very tedious. The following example shows results for a dropping function that has flat parts and an injective part.

Example 1. (RED dropping function) The RED dropping function (for its shape, see Fig. 1) is parameterized by two numbers: a threshold b_0, $0 \leq b_0 < b$, and a probability p_0, $0 < p_0 \leq 1$, such that

$$d_{b_0} = 0, \quad d_{b-1} = p_0. \tag{27}$$

Using these two parameters, the RED dropping function can be presented in the following form:

$$d_n = \begin{cases} 0 & \text{if} \quad 0 \leq n \leq b_0, \\ vn + w & \text{if} \quad b_0 < n < b, \\ 1 & \text{if} \quad n \geq b, \end{cases} \tag{28}$$

with

$$v = \frac{p_0}{b - b_0 - 1}, \quad w = -\frac{p_0 b_0}{b - b_0 - 1}. \tag{29}$$

We want to compute $D_{n,k}(s)$ for the dropping function (28), in the form of (21). To this end, we can first use the formula (20), then count all the obtained distinct roots of $D_{n,k}(s)$ and their multiplicities.

As we can see, the dropping function (28) has two flat parts: for $n \leq b_0$ and for $n \geq b$. Both of these flat parts cause root multiplications in (20). Moreover, we have a shifted index in formula (20). Namely, instead of c_n,

we have c_{n+i} in every factor of (20), with i depending on k. Therefore, the number of distinct roots and their multiplicities depend on n, k and their position with relation to b_0 and b. All we have to do is consider all possibilities. Having done this, we obtain

$$D_{n,k}(s) = \begin{cases} (s+c_0)^{b_0-n+1}(s+c_b)^{k-b+n+1}\displaystyle\prod_{i=b_0+1}^{b-1}(s+c_i) \\ \qquad \text{if} \quad k > b-n, n < b_0, \\ (s+c_b)^{k-b+n+1}\displaystyle\prod_{i=n}^{b-1}(s+c_i) \\ \qquad \text{if} \quad k > b-n, n \geq b_0, \\ (s+c_0)^{b_0-n+1}\displaystyle\prod_{i=b_0+1}^{n+k}(s+c_i) \\ \qquad \text{if} \quad b_0-n \leq k \leq b-n, n < b_0, \\ \displaystyle\prod_{i=n}^{n+k}(s+c_i) \\ \qquad \text{if} \quad b_0-n \leq k \leq b-n, n \geq b_0, \\ (s+c_0)^{k+1} \\ \qquad \text{if} \quad k \leq b_0-n. \end{cases} \tag{30}$$

All roots that occur in this representation are different. For instance, in the first line we have the following different roots:

$$-c_0,\ -c_b,\ -c_{b_0+1},\ldots,-c_{b-1}, \tag{31}$$

with the multiplicities

$$b_0-n+1,\ k-b+n+1,\ 1,\ldots,1, \tag{32}$$

respectively.

Instead of implementing the formulas (21)–(26), we can also use software for symbolic computations that has a built-in function to compute the inversion of (8) automatically (e.g., *Mathematica*, 2011). ♦

4. Queue size distribution at an arbitrary time

In this section we will find the queue size distribution at an arbitrary time, namely:

$$P_k = \lim_{t\to\infty} \mathbf{P}(X(t)=k), \quad 0 \leq k \leq b, \tag{33}$$

where $X(t)$ denotes the queue size at the time t.

Denoting by

$$\rho = \lambda m_F, \tag{34}$$

the offered load of the system, here

$$m_F = \text{the average service time} = \int_0^\infty x\,\mathrm{d}F(x), \tag{35}$$

we can prove the following theorem.

Theorem 2. *In the M/G/1/b(AQM) system the queue length distribution at an arbitrary time has the form*

$$P_k = \frac{\pi_k/(1-d_k)}{\pi_0/(1-d_0)+\rho}, \quad 0 \leq k \leq b-1, \tag{36}$$

$$P_b = 1 - \frac{\sum_{i=0}^{b-1}\pi_i/(1-d_i)}{\pi_0/(1-d_0)+\rho}. \tag{37}$$

Note that the distribution π_k was computed in the previous section. Therefore, the formulas (36), (37) can be used immediately to obtain numerical results.

Proof. Let X'_n denote the queue size observed upon arrival of the n-th arriving job (accepted or not) and let P'_k denote its limiting distribution, namely,

$$P'_k = \lim_{n\to\infty} \mathbf{P}(X'_n = k), \quad 0 \leq k \leq b. \tag{38}$$

According to the PASTA property (Heyman and Sobel, 1982, p. 391), we know that

$$P_k = P'_k, \quad 0 \leq k \leq b. \tag{39}$$

Therefore, our task is reduced to computing P'_k for $0 \leq k \leq b$.

Let $\hat{X}_n$ denote the queue size observed upon arrival of the n-th accepted job and let $\hat{\pi}_k$ denote its limiting distribution:

$$\hat{\pi}_k = \lim_{n\to\infty} \mathbf{P}(\hat{X}_n = k), \quad 0 \leq k \leq b-1. \tag{40}$$

(Note that we always have $\hat{X}_n < b$.) According to Burke's theorem (Takagi, 1991, p. 7), we have

$$\hat{\pi}_k = \pi_k, \quad 0 \leq k \leq b-1. \tag{41}$$

Now, let P^*_k denote the limiting distribution of the queue size observed upon arrival of the n-th arriving job (accepted or not), excluding queue sizes equal to b. Therefore, distributions P'_k and P^*_k are proportional in the range $0 \leq k \leq b-1$, namely,

$$P'_k = c\,P^*_k, \quad 0 \leq k \leq b-1, \tag{42}$$

for some constant c.

On the other hand, we have

$$P^*_k = \frac{\hat{\pi}_k}{(1-d_k)h}, \quad 0 \leq k \leq b-1, \tag{43}$$

where

$$h = \sum_{i=0}^{b-1}\frac{\hat{\pi}_i}{1-d_i}. \tag{44}$$

Combining

$$\sum_{k=0}^{b} P'_k = 1 \tag{45}$$

with

$$\sum_{k=0}^{b-1} P_k^* = 1, \tag{46}$$

and (42) yields

$$c + P_b' = 1, \tag{47}$$

and we have to find c now.

The probability that the system is empty is

$$P_0 = P_0' = 1 - \rho_c, \tag{48}$$

where ρ_c is the carried load of the system

$$\rho_c = \rho(1 - P_B), \tag{49}$$

and P_B is the blocking probability

$$P_B = \sum_{k=0}^{b} d_k P_k'. \tag{50}$$

(Note that in our system the blocking probability is not equal to the full-buffer probability, i.e., $P_B \neq P_b$.)

Rewriting (48), we obtain

$$P_B = 1 - \frac{1 - P_0'}{\rho}. \tag{51}$$

Combining (51), (50), (42) and (47) yields

$$\sum_{k=0}^{b-1} d_k c P_k^* + 1 - c = 1 - \frac{1 - cP_0^*}{\rho}, \tag{52}$$

and, as a consequence,

$$c = \frac{1}{P_0^* + \rho - \rho \sum_{k=0}^{b-1} d_k P_k^*}. \tag{53}$$

Taking into account (42) we get

$$P_k' = \frac{P_k^*}{P_0^* + \rho - \rho \sum_{i=0}^{b-1} d_i P_i^*}, \quad 0 \leq k \leq b-1, \tag{54}$$

while using (47) we obtain

$$P_b' = 1 - \frac{1}{P_0^* + \rho - \rho \sum_{i=0}^{b-1} d_i P_i^*}. \tag{55}$$

Then, using (43) we obtain

$$P_k' = \frac{\hat{\pi}_k/(1-d_k)}{\hat{\pi}_0/(1-d_0) + \rho h - \rho \sum_{i=0}^{b-1} d_i \hat{\pi}_i/(1-d_i)}, \quad 0 \leq k \leq b-1, \tag{56}$$

$$P_b' = 1 - \frac{h}{\hat{\pi}_0/(1-d_0) + \rho h - \rho \sum_{i=0}^{b-1} d_i \hat{\pi}_i/(1-d_i)}, \tag{57}$$

which, combined with (39) and (41), completes the proof. ■

5. Loss ratio and throughput

Besides the queue size distribution, there are two other characteristics that are important from a practical point of view (especially when dealing with AQM in Internet routers), namely, the loss ratio and system throughput.

The loss ratio is the long-run fraction of jobs (packets) that are dropped. As it is equivalent to the probability that an arriving job is dropped, P_B, it can be easily calculated using the formula (51) with $P_0' = P_0$ obtained by means of Theorem 2.

The throughput of the system is the average rate of the output traffic and is equal to

$$\gamma = \lambda(1 - P_B). \tag{58}$$

Therefore, it can also be computed by means of (51) and Theorem 2.

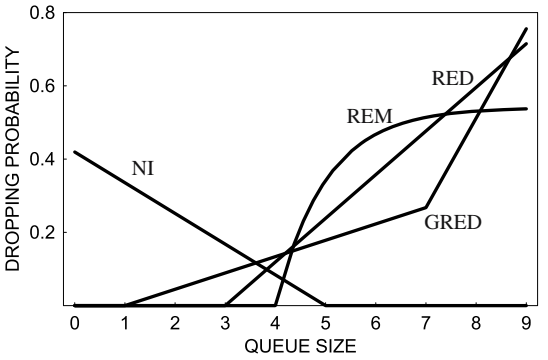

Fig. 1. Dropping functions used in numerical examples.

6. Numerical examples

In this section we demonstrate queue size distributions and loss ratios for different system loads and dropping functions. We assume that the service time, d, is constant and equal to 1. Manipulating λ we can change the load $\rho = \lambda d$ offered to the system. We consider three cases:

(a) $\rho = 1$ (critically loaded system),

(b) $\rho = 2$ (overloaded system),

(c) $\rho = 0.5$ (underloaded system),

obtained for $\lambda = 1$, $\lambda = 2$ and $\lambda = 0.5$, respectively. We assume that $b = 10$ and use the following four dropping functions:

(A) the RED dropping function:

$$d_n = \begin{cases} 0 & \text{if } n \leq 3, \\ 0.11917n - 0.35752 & \text{if } 3 < n < 10, \\ 1 & \text{if } n \geq 10, \end{cases} \tag{59}$$

(B) the gentle RED (GRED) dropping function:

$$d_n = \begin{cases} 0 & \text{if } n \le 1, \\ 0.04460n - 0.04460 & \text{if } 1 < n < 7, \\ 0.24414n - 1.44140 & \text{if } 7 \le n < 10, \\ 1 & \text{if } n \ge 10, \end{cases} \tag{60}$$

(C) the REM dropping function:

$$d_n = \begin{cases} 0 & \text{if } n \le 4, \\ -0.54085e^{-n+4} + 0.54085 & \text{if } 4 < n < 10, \\ 1 & \text{if } n \ge 10, \end{cases} \tag{61}$$

(D) the non-increasing (NI) dropping function:

$$d_n = \begin{cases} -0.08387n + 0.41936 & \text{if } n \le 5, \\ 0 & \text{if } 5 < n < 10, \\ 1 & \text{if } n \ge 10. \end{cases} \tag{62}$$

All these dropping functions are depicted in Fig. 1.

All the dropping functions (A)–(D) were carefully parameterized in order to force the average queue size equal to 3 when $\rho = 1$ (this will be further commented on at the end of this section).

Now we can present the numerical results. In Table 1 and Fig. 2, results for $\rho = 1$ are shown. In particular, in Table 1 the average queue size at an arbitrary time, the standard deviation of the queue size, the loss ratio, the full and empty buffer probability are presented, while in Fig. 2, detailed distributions of the queue size at an arbitrary time are depicted. We see that the loss ratio and the queue size deviation may differ significantly, depending on the dropping function, even though the average queue size is common. For instance, the loss ratio for the NI function is about 25% while for the RED function it is about 9%. Generally speaking, the more the dropping function is concentrated around low queue sizes, the higher the loss ratio and the queue size deviation. Note also that in this example the empty system probability and the loss ratio are equal. This is connected with the fact that $\rho = 1$ (see (51)).

In Table 3 and Fig. 3, results for $\rho = 2$ are presented. The loss ratio is now almost the same for all dropping functions. This is connected with the fact that the system is strongly overloaded, the idle periods are very rare (see P_0), and the loss ratio is close to the minimal possible value of $1/2$. Therefore, in this example we obtain four systems with almost the same loss ratio but different average queue sizes and different queue size deviations. Especially a low deviation in the queue size can be observed for the NI dropping function. This observation indicates that the dropping function can be used to design a system with a predictable queue size (in terms of a low queue size variance).

In Table 4 and Fig. 4, results for $\rho = 0.5$ are depicted. From the practical point of view, this is the least interesting case, as for $\rho \ll 1$ the queue is stable and has a low average length even without the dropping function. However, even in this case, application of a specific dropping function can be beneficial, depending on what we are aiming at. For instance, the NI function gives a very low queue size (at the cost of a high loss ratio).

To check the correctness of the numerical results obtained by means of Theorems 1 and 2, we performed also a number of simulations using the Omnet++ network simulator (*Omnet*, 2011). Several different dropping functions and system parameterizations were used, and simulation results always agreed very well with analytical results. A sample output for the RED dropping function and $\rho = 1$ is presented in Table 2. The queue size distribution was observed empirically upon arrivals of 10^8 jobs (equivalent to 10 mins of work by the simulator).

Table 2. Analytical vs. simulation results for the RED dropping function and $\rho = 1$.

Characteristic of interest	Analytical results	Simulation results
Average queue size	3.0000	2.9991
Std. dev.	1.8877	1.8873
Loss ratio	9.1281×10^{-2}	9.1227×10^{-2}
P_0	9.1281×10^{-2}	9.1312×10^{-2}
P_1	1.5684×10^{-1}	1.5691×10^{-1}
P_2	1.7822×10^{-1}	1.7832×10^{-1}
P_3	1.8217×10^{-1}	1.8215×10^{-1}
P_4	1.6764×10^{-1}	1.6760×10^{-1}
P_5	1.2088×10^{-1}	1.2090×10^{-1}
P_6	6.6513×10^{-2}	6.6455×10^{-2}
P_7	2.7031×10^{-2}	2.6974×10^{-2}
P_8	7.7614×10^{-3}	7.7407×10^{-3}
P_9	1.4733×10^{-3}	1.4650×10^{-3}
P_{10}	1.4668×10^{-4}	1.4577×10^{-4}

As mentioned above, all the dropping functions in this section were chosen so that the average queue size is equal to 3 for $\rho = 1$. This is a good example that the dropping function is a powerful tool. Instead of the average queue size, we could achieve a common loss ratio. The question arises what target queue sizes or loss ratios can be achieved. The natural bounds are given by the finite-buffer queue without the dropping function. For instance, for $\rho = 1$ and $b = 10$, in the simple finite-buffer model the average queue size is 5.0650 and the loss ratio is 0.05085. Therefore, using dropping functions we can obtain a system with any average queue size in the interval $[0, 5.0650]$ or a system with any loss ratio in the interval $[0.05085, 1]$.

The potential of the presented results for improvement of known algorithms (e.g., RED, GRED, REM) and development of new algorithms is twofold.

Table 1. Basic queueing characteristics for the dropping functions (a)–(d) and $\rho = 1$.

	Average queue size	Standard deviation	Loss ratio or P_B	Empty system probab., P_0	Full buffer probab., P_b
RED	3.0000	1.8877	9.1281×10^{-2}	9.1281×10^{-2}	1.4669×10^{-4}
GRED	3.0000	2.0517	9.9384×10^{-2}	9.9384×10^{-2}	3.8145×10^{-4}
REM	3.0000	1.8329	8.9339×10^{-2}	8.9339×10^{-2}	2.3479×10^{-4}
NI	3.0000	3.0283	2.4802×10^{-1}	2.4802×10^{-1}	2.5923×10^{-2}

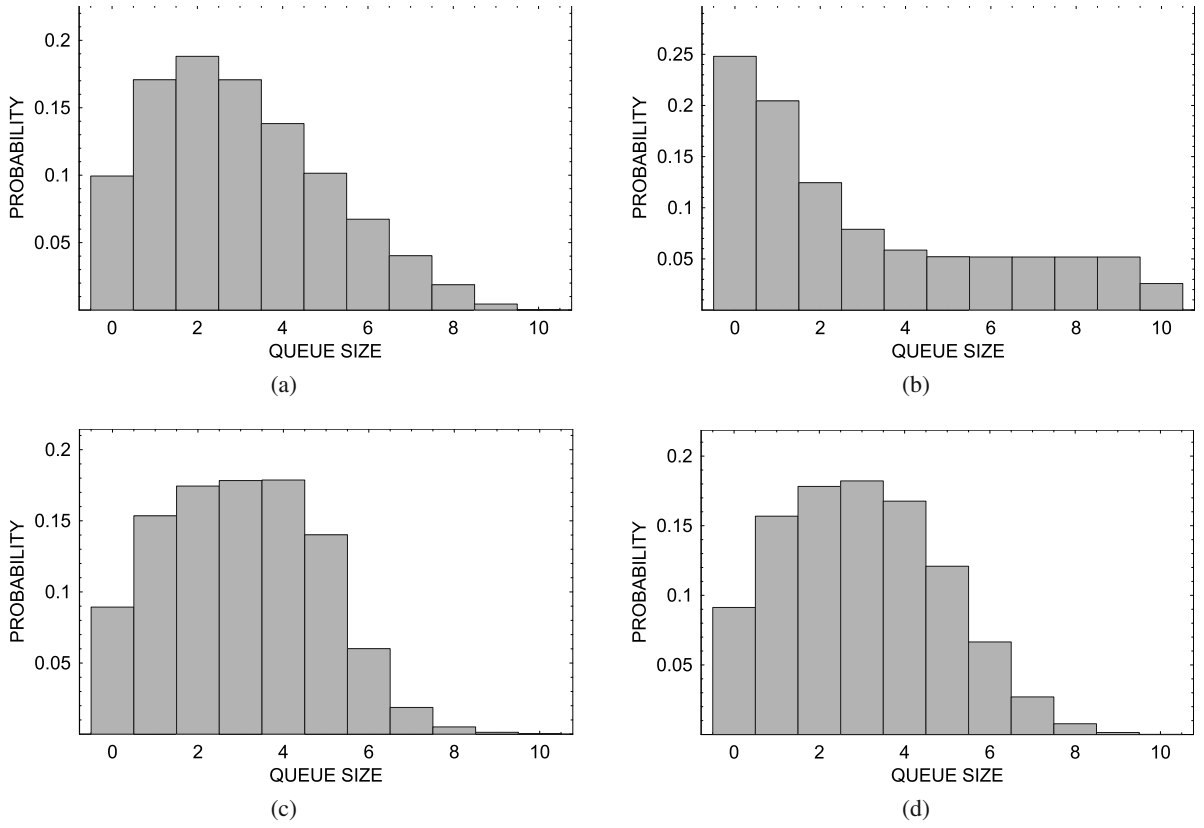

Fig. 2. Queue size probability distribution at an arbitrary time for $\rho = 1$ and different dropping functions: RED (a), GRED (b), REM (c), NI (d).

First of all, we can use the analytical results to compare several dropping functions with respect to the performance characteristics of interest. For instance, for $\rho = 1$, the best dropping function, among the four considered in this section, is REM. As can be observed in Table 1, REM gives the smallest loss ratio (and, as a consequence, the highest throughput) and the most stable queue (the smallest standard deviation of the queue size), while keeping the same, as the other dropping functions, average queue size. Naturally, this conclusion may not be valid for different values of ρ. Also, it is likely that there are dropping functions that produce better performance results than REM defined in (61). Theorem 2 does not point out such dropping functions directly. However, using Theorem 2 implemented in some programming language, we can perform quickly a number of experiments with shapes of dropping functions and choose the most suitable one for our purposes.

Secondly, we have already demonstrated that we can find several dropping functions that guarantee a required average queue size. The same is true regarding other characteristics—we can find dropping functions that will keep the throughput, the loss ratio and the variance of the queue size on a given level. In fact, dropping functions are even more powerful than that. Using their variable shapes, we may try to control all of the mentioned characteristics at the same time (though in some limited ranges). Moreover, we can find dropping functions that guarantee different performance characteristics for different system loads. For instance, it is possible to find a dropping func-

Table 3. Basic queueing characteristics for the dropping functions (a)–(d) and $\rho = 2$.

	Average queue size	Standard deviation	Loss ratio, P_B	Empty system probab., P_0	Full buffer probab., P_b
RED	7.1458	1.4367	0.50002	4.4488×10^{-5}	3.4385×10^{-2}
GRED	7.7890	1.2363	0.50001	2.2334×10^{-5}	5.0057×10^{-2}
REM	7.1327	1.6627	0.50003	5.2932×10^{-5}	7.4634×10^{-2}
NI	9.3721	0.7457	0.50000	3.0466×10^{-6}	4.9997×10^{-1}

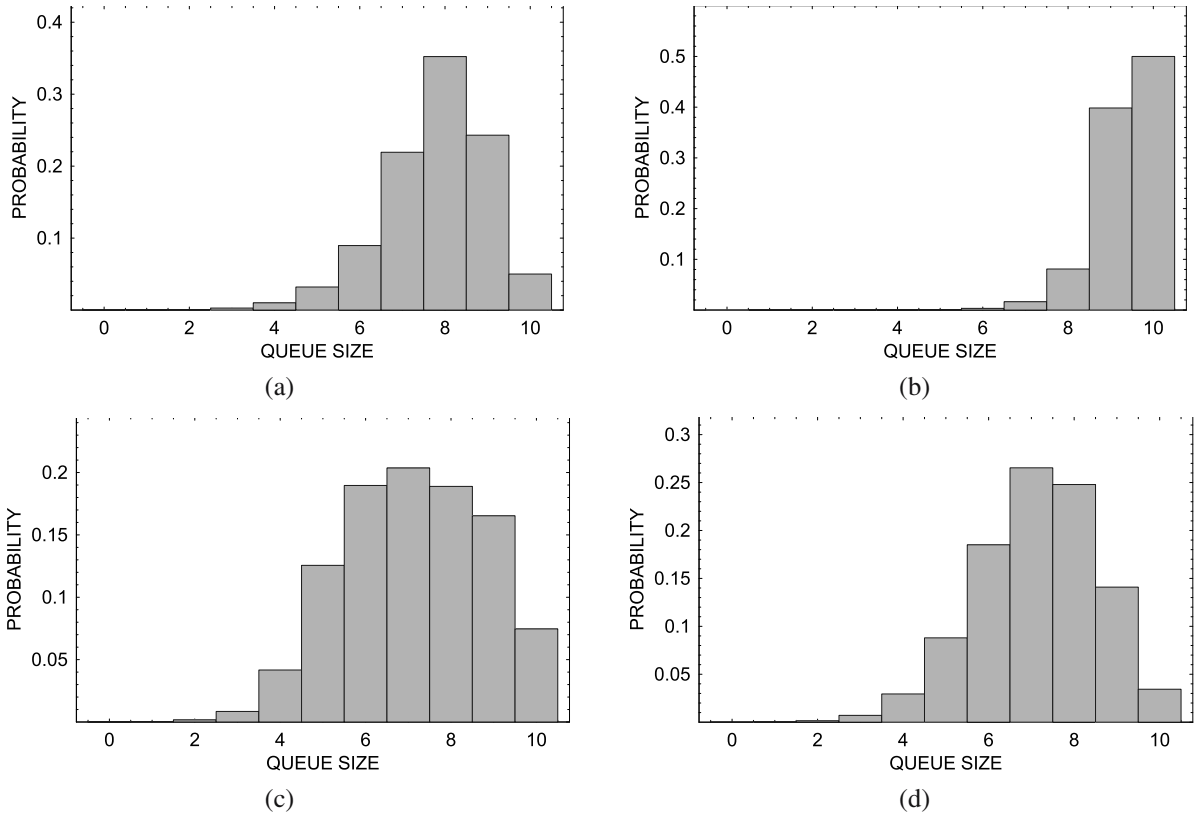

Fig. 3. Queue size probability distribution at an arbitrary time for $\rho = 2$ and different dropping functions: RED (a), GRED (b), REM (c), NI (d).

tion that gives the average queue size equal to 2.0000 for $\rho = 1$ and equal to 3.0000 for $\rho = 1.2$, etc.

The analytical results presented herein have also some limitations when used in performance evaluation of AQM algorithms designed for Internet routers. This is mainly due to the fact that fully functional AQM usually exploits some other mechanisms, besides the pure packet dropping based on the actual queue size. For instance, the RED algorithm uses the EWMA (Exponentially Weighted Moving Average) queue size, which depends on the present and past queue size. Solving analytically the AQM queue with packet dropping depending on the averaged (instead of the actual) queue size seems to be out of reach at the moment.

7. Conclusions and future work

We presented an analysis of a finite-buffer queue in which an arriving job is dropped with a probability that depends on the queue size observed upon arrival. In particular, solutions for the queue size at departure epochs, the queue size at an arbitrary time, the loss ratio and throughput were shown.

It is worth emphasizing that the dropping discipline considered herein is not only of use in active queue management in Internet routers, but also has some deep universal sense. In a system in which the arrival and the service rate cannot be manipulated, application of the dropping function is the simplest way to control the average queue size or loss ratio. Therefore, the potential applicability of such queueing systems is wide and general.

Table 4. Basic queueing characteristics for the dropping functions (a)–(d) and $\rho = 0.5$.

	Average queue size	Standard deviation	Loss ratio, P_B	Empty system probab., P_0	Full buffer probab., P_b
RED	0.74083	0.92341	1.9870×10^{-3}	5.0099×10^{-1}	1.7082×10^{-8}
GRED	0.72134	0.89813	1.0099×10^{-2}	5.0504×10^{-1}	3.7643×10^{-8}
REM	0.74387	0.92902	1.1513×10^{-3}	5.0057×10^{-1}	2.5920×10^{-8}
NI	0.39380	0.68022	3.8635×10^{-1}	6.9317×10^{-1}	7.9364×10^{-7}

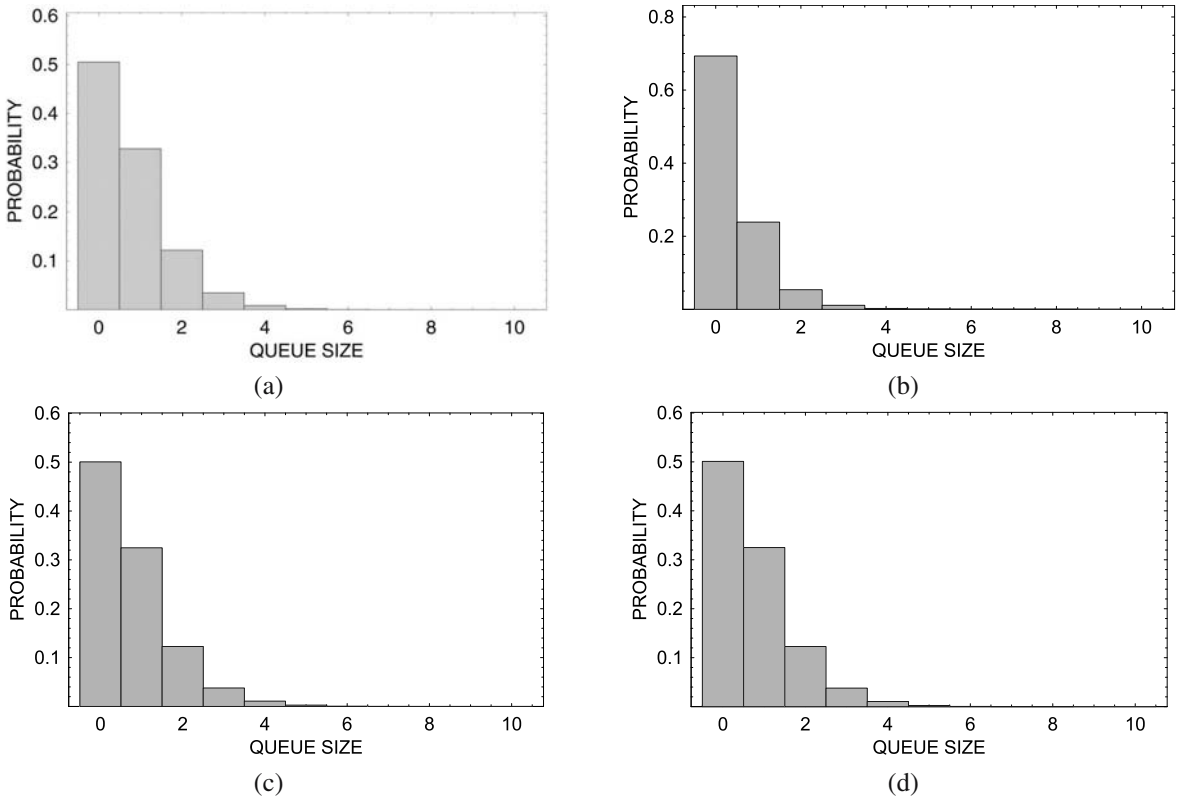

Fig. 4. Queue size probability distribution at an arbitrary time for $\rho = 0.5$ and different dropping functions: RED (a), GRED (b), REM (c), NI (d).

As for future work, we believe that systems with a dropping function and an infinite buffer (waiting room) should be investigated in the next step. The first problem in the analysis of such systems is which dropping functions guarantee the system stability (i.e., the existence of a limiting distribution of the queue size). We know that an infinite-buffer queue without a dropping function is stable if $\rho < 1$ and unstable if $\rho \geq 1$. However, introduction of the dropping function changes this situation. For instance, a system with $\rho = 2$ and a trivial dropping function $d_n = 0.6$ for every n is obviously stable. But when more complicated dropping functions are involved, the problem gets tougher.

On the other hand, it would be interesting to study the queue with a dropping function and a more complex arrival process. For instance, the Internet traffic is known to be bursty and self-similar. Therefore, batch arrivals (e.g., a batch Poisson process) and processes that are able to mimic self-similar behaviour (e.g., a Markov-modulated Poisson process, a batch Markovian arrival process) should be included in a future study. We believe that the presented methodology can be extended to all aforementioned processes. However, such extensions are not trivial and require a lot of additional work.

Acknowledgment

This material is based upon work supported partially by the Polish Ministry of Science and Higher Education under Grant No. N N516 381134.

References

Athuraliya, S., Low, S.H., Li, V.H. and Yin, Q. (2001). REM: Active queue management, *IEEE Network* **15**(3): 48–53.

Bohacek, S., Shah, K., Arce, G.R. and Davis, M. (2004). Signal processing challenges in active queue management, *IEEE Signal Processing Magazine* **21**(5): 69–79.

Bonald, T., May, M. and Bolot, J.C. (2000). Analytic evaluation of RED performance, *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Tel-Aviv, Israel,* pp. 1415–1424.

Chatranon, G., Labrador, M.A. and Banerjee, S. (2004). A survey of TCP-friendly router-based AQM schemes, *Computer Communications* **27**(15): 1424–1440.

Fatta, G.D., Hoffmann, F., Re, G.L. and Urso, A. (2003). A genetic algorithm for the design of a fuzzy controller for active queue management, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **33**(3): 313–324.

Feng, W., Shin, K.G., Kandlur, D.D. and Saha, D. (2002). The BLUE active queue management algorithms, *IEEE/ACM Transactions on Networking* **10**(4): 513–528.

Floyd, S., Gummadi, R. and Shenker, S. (2001). Adaptive RED: An algorithm for increasing the robustness of REDs active queue management, *Technical Report*, ACIRI, `http://icir.org/floyd/papers/adaptiveRed.pdf`.

Floyd, S. and Jacobson, V. (1993). Random early detection gateways for congestion avoidance, *IEEE/ACM Transactions on Networking* **1**(4): 397–413.

Hao, W. and Wei, Y. (2005). An extended $GI^X/M/1/N$ queueing model for evaluating the performance of AQM algorithms with aggregate traffic, *in* X. Lu and W. Zhao (Eds.), *Networking and Mobile Computing*, Lecture Notes in Computer Science, Vol. 3619, Springer, Berlin/Heidelberg, pp. 395–404.

Heyman, D.P. and Sobel, M.J. (1982). *Stochastic Models of Operations Research, Volume 1: Stochastic Processes and Operating Characteristics*, McGraw-Hill, New York, NY.

Hollot, C.V., Misra, V., Towsley, D. and Gong, W. (2002). Analysis and design of controllers for AQM routers supporting TCP flows, *IEEE Transactions on Automatic Control* **47**(6): 945 –959.

Kunniyur, S.S. and Srikant, R. (2004). An adaptive virtual queue (AVQ) algorithm for active queue management, *IEEE/ACM Transactions on Networking* **12**(2): 286 – 299.

Lakshmikantha, A., Beck, C.L. and Srikant, R. (2005). Robustness of real and virtual queue-based active queue management schemes, *IEEE/ACM Transactions on Networking* **13**(1): 81–93.

Mathematica (2011). `http://www.wolfram.com/`.

Network Simulator ns-2 (2011). `http://www.isi.edu/nsnam/ns/`.

Omnet (2011). `http://www.omnetpp.org/`.

Opnet (2011). `http://www.opnet.com/`.

Rosolen, V., Bonaventure, O. and Leduc, G. (1999). A RED discard strategy for ATM networks and its performance evaluation with TCP/IP traffic, *ACM SIGCOMM Computer Communication Review* **29**(3): 23–43.

Sun, J. and Zukerman, M. (2007). An adaptive neuron AQM for a stable internet, *in* I. Akyildiz, R. Sivakumar, E. Ekici, J. Oliveira and J. McNair (Eds.), *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, Lecture Notes in Computer Science, Vol. 4479, Springer, Berlin/Heidelberg, pp. 844–854.

Takagi, H. (1991). *Queueing Analysis—Vacation and Priority Systems*, North-Holland, Amsterdam.

Takagi, H. (1993). *Queueing Analysis—Finite Systems*, North-Holland, Amsterdam.

Andrzej Chydziński obtained his Ph.D. and D.Sc. in computer science from the Silesian University of Technology in 2002 and 2008, respectively. He is currently a professor in the Institute of Informatics of the same university. His academic interests include performance evaluation of computer networks, future Internet design and queue management in Internet routers. He has authored and co-authored three books as well as more than sixty journal and conference papers.

Łukasz Chróst was born in 1982. He obtained his M.Sc. in computer science in 2006 from the Silesian University of Technology, Poland. Currently he is a research assistant at the Faculty of Automatic Control, Electronics and Computer Science of the same university. His research interests include queueing theory and its applications in virtualization of computer systems as well as the deterministic approach to active queue management.

Received: 26 September 2010
Revised: 15 December 2010