Stability of AQM algorithms in low congestion scenarios

Pawel Mrozowski and Andrzej Chydzinski

Abstract— It is well known that maintaining a stable queue size and high throughput in routers operating in high bandwidth-delay product networks is a difficult task. Fortunately, some newly proposed active queue management solutions (by Sun et al. and Ren et al.) seem to work quite well in such environments. In this paper we demonstrate that two additional factors make the task of achieving a stable queue size and high throughput very difficult. Namely, when the congestion level is low or the target queue size is short, none of the known AQMs performs reasonably well - the queue is very unstable and the throughput often goes far below fifty percent of the link capacity. Therefore, new AQM algorithms, able to work well in such scenarios, are needed.

Keywords— active queue management, Internet routers, packet queueing, performance evaluation

I. Introduction

The active queue management (AQM) algorithms for Internet routers (see, for example, [2]-[24]) are designed with several objectives in mind (see [15] for a comprehensive list). However, the stability of the queue size, in terms of low queue size variance, plays a central role in this list as it is strictly connected with other requirements. For instance, an AQM that is able to maintain a very low variance of the queue size guarantees also that the achieved throughput is close to maximal possible one. This is due to the fact that the empty queue probability is very low in a system with such AQM. Therefore, a stable queue size automatically provides a good throughput.

The other important objective of the active queue management is a low queue size. Obtaining a stable but long queue is not a very demanding task, and even a simple drop-tail algorithm can do that to some extent. Therefore, these two requirements, stable and short queue size, have to be always presented together.

What are the factors that make it difficult to preserve a stable and short queue size in the router's buffer? In [16] a large propagation delay was identified as the reason of unstability of several popular AQM algorithms. However, the newest algorithms, like AN-AQM [23], can keep a stable queue even if the RTPT is as long as 500ms. Therefore the question arises whether the problem was definitely solved.

In this paper we show that it is not so. In particular, we demonstrate that two other factors make the AQMs to be prone to high queue size variations. These factors are: low congestion level and low average queue size. The best algorithms can stabilize the queue size very well, but only when the congestion is heavy or moderate and when the buffer and the target queue size are not very small. For instance, in [23] the superior performance of AN-AQM is shown using at least 300 flows on 45Mb/s link, which is equivalent to a rather heavy congestion. Moreover, a large buffer (900 packets) was used and the target queue size was set to 300 packets (50 in one of the experiments).

In this paper we study the stability of the queue size using a small buffer (90 packets), a low congestion level and a new AQM testbed specification. The results indicate that the variation of the queue size grows when the congestion level diminishes or when the queue gets shorter. In fact, none of the algorithms used in our experiments performed very well when these two unfavourable factors were involved. Therefore, new algorithms that can stabilize the queue size under such network conditions are needed.

The remaining part of the report is organized in the following way. In Section II, we present a detailed description of the network model and simulation environment used in our experiments. In Section III, we describe briefly six active queue management algorithms used in the paper. Then, in Section IV, the simulation results are presented and discussed. In particular, the average queue size, the standard deviation and the variation coefficient are presented for different queue management algorithms and congestion scenarios. The final remarks and conclusions are given in Section V.

II. Network model

In simulations we follow the *trans-oceanic link* scenario described in [25]. This is a long-delay scenario, suitable for our purposes. In particular, the standard dumb-bell topology with two routers, A and B, the bottleneck link A-B, and six network nodes, N1-N6 are used (see Fig. 1). The link propagation delays are the following:

N1-RA:	0ms,
N2-RA:	12ms,
N3-RA:	25ms,
N4-RB:	2ms,
N5-RB:	37ms,
N6-RB:	75ms,
RA-RB:	65ms.

Therefore, the longest RTPT (N3-N6-N3) is equal to 330s. All the links have the capacity of 100Mb/s.

The TCP senders are located in nodes N1-N3 and they transmit data to nodes N4-N6 over all nine possible transmission

This is extended version of the paper [1] presented during ACC'09 conference, Athens, September 27-29, 2009.

The authors are with the Silesian University of Technology, Institute of Computer Sciences. Address: Akademicka 16, 44-100 Gliwice, Poland. (corresponding author's e-mail: Andrzej.Chydzinski@polsl.pl)

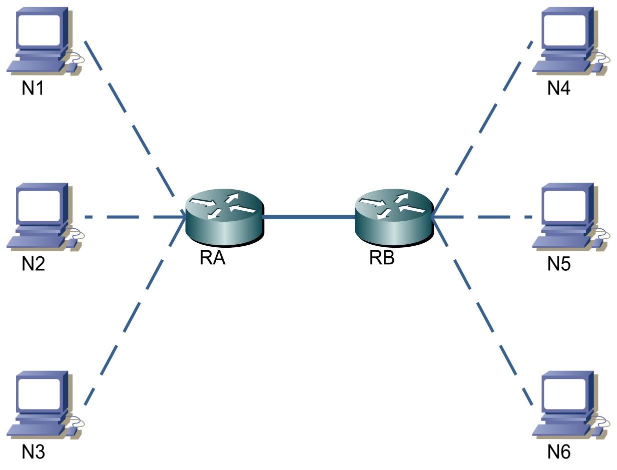

Fig. 1. The dumb-bell network topology.

paths: N1-N4, N1-N5, N1-N6, N2-N4, N2-N5, N2-N6, N3-N4, N3-N5, N3-N6.

The total numbers of TCP connections on the bottleneck link A-B depends on the congestion scenario. Three congestion scenarios are considered:

- 10 TCP connections - the uncongested network scenario,
- 100 TCP connections - the mild congestion scenario,
- 1000 TCP connections - the heavy congestion scenario.

What is important, all the TCP connections are uniformly distributed among nine transmission paths listed above.

The detailed description of the TCP traffic is the following. 90 percent of the TCP flows on each transmission path are using 1500 bytes long packets and the remaining 10 percent – 536 bytes long packets. All 536-bytes-long connections and 75 percent of the 1500-bytes-long connections are FTP flows transmitting large files. The remaining 25 percent of the 1500-bytes-long TCP connections transmit data in a way that is supposed to mimic HTTP traffic. Namely, through each of these connections a large number of small files is sent. Transmissions of these small files begin according to the Poisson process with rate 12.5 in the uncongested network, 1.25 in the mild congestion scenario and 0.125 in the heavy congestion scenario. The file sizes are Pareto-distributed with the average size of 50 kbytes and the shape parameter set to 1.3.

Additionally, reverse traffic is simulated. The total number of reverse flows is 1, 10 and 100 in the uncongested, mildly congested and heavily congested network, respectively. They are uniformly distributed among all nine transmission paths. The reverse flows are UDP CBRs with 1000-bytes-long datagrams and the sending rate chosen so that the total reverse traffic uses 10 percent of the bandwidth of the bottleneck links in the reverse direction.

If not stated differently, the buffer size of 90 packets was used, which is approximately 10 percent of the bandwidth-delay product (with the average delay equal to 100ms). This buffer size was chosen to achieve short queue sizes. In some experiment a larger buffer (900 packets) was used, but all such experiment are clearly indicated.

The following metrics on the bottleneck link are collected from 30th to 100th second of the simulation time: the average queue size, the average throughput, the standard deviation of the queue size and the coefficient of variation of the queue size. Additionally, the queue size is recorded every 0.1 second in order to present its time-dependent behavior.

The Ns2 simulator ver. 2.33 is used [26]. In addition to classic drop-tail queue (DT), six other AQM algorithms are used: RED, BLUE, PI, REM, AVQ and AN-AQM. Default Ns2 implementations and parameterizations of RED, PI, REM, AVQ are used, BLUE was implemented and parameterized following [8], AN-AQM is implemented and parameterized according to [23] with the target queue size equal to 30% of the buffer size.

We will describe briefly these algorithms in the next section.

III. AQM ALGORITHMS

RED (Random Early Detection, [2], [4]) was the first widely studied AQM algorithm. Its functionality begins when the queue gets longer than the minimum threshold value. When this happens, the packets are marked and dropped according to the linear marking probability function. Passing through the maximum threshold the dropping probability is set to 1 and all the packets are dropped. An important feature of the RED algorithm is the exponentially weighed moving average used for low-pass filtering of the queue size.

The BLUE algorithm [8] uses packet loss and link utilization history to manage congestion. The main difference between BLUE and RED (and many other AQMs) is that BLUE does not use instantaneous or average queue length in the process of dropping/marking packets. The value of the dropping probability is increased when the queue starts to drop packets due to the buffer overflow. As a result, the rate of congestion notification is increased. When the queue becomes empty, the dropping probability is decreased, which enables managing effectively the link utilization.

REM’s (Random Exponential Marking, [5]) idea is to decouple congestion measure from performance measure. Two main features of REM are the adjustment of the amount of user’s data rate to the link capacity and calculating value called ”the price”. This value is then used in packets dropping process. The probability marking function is exponential.

The fourth algorithm used herein is PI [7]. It is based on a classic proportional integrator controller. In opposite to RED, the loss probability is calculated from instantaneous queue length, not the averaged queue length. PI tries to regulate the steady-state of the queue at desired value.

The fifth AQM used is AVQ [13]. The idea behind this acronym states for maintaining an adaptive virtual queue (VQ). It is used to detect overflow events caused by virtual representative of the real packet. Virtual queue capacity is set equal or smaller to the capacity of the link. Its buffer size is also equal to the buffer size of the real queue. After the occurrence of the overflow event, the real packet is marked or dropped. This approach enables the system to control utilization of the link, not the queue size.

Finally, the AN-AQM (Adaptive Neuron AQM, [23]) is a novel, neuron-based scheme that is designed especially for the purpose of the queue size stabilization.

We decided not to use the DC-AQM algorithm (see [16]) in our experiments, although it is designed especially for long delay networks. This is due to the fact that it has some serious practical limitations (RTPT has to be constant and known).

IV. Simulation results

In Figures 2-8 the bottleneck queue size process for seven queue management algorithms in the mild congestion scenario is shown. Although the congestion level was not very low in these simulations, it becomes obvious at first glance that none of the algorithms were able to keep a stable queue size during the experiment (for comparison, see very flat plots obtained in [23]). Detailed queueing characteristics for the mild congestion scenario are presented in the third column of Tabs. I-IV. The standard deviation of the queue size is high for every algorithm, resulting in a relatively low throughput (varying from 71 to 92 percent of the bottleneck link capacity).

	uncongested network	mild congestion	heavy congestion
DT	1,8	22,0	73,5
RED	0,4	4,3	22,2
BLUE	1,7	17,5	48,8
REM	1,7	21,8	70,6
PI	1,6	21,8	70,9
AVQ	1,0	13,1	21,5
AN-AQM	0,6	4,1	18,9

TABLE I
THE AVERAGE QUEUE SIZE (IN PACKETS).

	uncongested network	mild congestion	heavy congestion
DT	40,5%	92,0%	100,0%
RED	24,2%	71,6%	94,5%
BLUE	39,3%	91,9%	100,0%
REM	39,3%	92,3%	100,0%
PI	35,2%	91,9%	99,8%
AVQ	34,5%	87,6%	98,0%
AN-AQM	24,6%	72,0%	99,3%

TABLE II
THE AVERAGE THROUGHPUT.

The situation gets even worse when we further decrease the network congestion. In the second column of Tabs. I-IV we can see the queueing performance in the uncongested network. The queue is very unstable in this case – the coefficient of variation varies from 471 to 692 percent. This unstability of the queue size yields a very poor throughput (see Tab. II), which is only 40 percent in the best case. Sample queue size processes in the uncongested network are presented in Figs. 9-11 (for DT, RED and AN-AQM, respectively).

	uncongested network	mild congestion	heavy congestion
DT	8,6	25,6	21,4
RED	2,4	7,9	9,7
BLUE	8,1	24,4	38,9
REM	8,1	25,2	22,0
PI	8,0	24,9	22,1
AVQ	5,5	18,0	17,9
AN-AQM	3,9	7,9	9,8

TABLE III
THE STANDARD DEVIATION OF THE QUEUE SIZE (IN PACKETS).

	uncongested network	mild congestion	heavy congestion
DT	473,5%	116,2%	29,1%
RED	674,2%	184,5%	43,6%
BLUE	472,7%	145,1%	79,7%
REM	471,1%	115,8%	31,1%
PI	496,3%	114,2%	31,2%
AVQ	530,1%	137,6%	83,3%
AN-AQM	692,9%	191,8%	52,2%

TABLE IV
THE COEFFICIENT OF VARIATION OF THE QUEUE SIZE.

	queue size	throughput	std. dev. queue size
DT	227,0	94,0%	272,0
RED	4,7	60,2%	17,4
BLUE	213,6	97,8%	287,2
REM	25,6	93,3%	47,6
PI	3,6	61,5%	13,6
AVQ	38,0	79,4%	74,6
AN-AQM	38,1	83,6%	63,8

TABLE V
QUEUEING PERFORMANCE FOR A LARGE BUFFER (900PKTS) IN THE UNCONGESTED SCENARIO.

In the previously discussed results, both short buffer and low congestion level could influence the network performance. Now, let us check an impact of a short queue size only. For that purpose, we should analyze the fourth column in Tabs. I-IV containing results for the heavy congestion scenario. The throughput is much better now, but the standard deviation of the queue size is still high and the queue is still not very stable (see also sample queue size processes in the heavy congestion scenario presented in Figs. 12-14 for DT, RED and AN-AQM, respectively).

Finally, we should check a bare impact of the low congestion level, allowing the queue size to be high. For that purpose

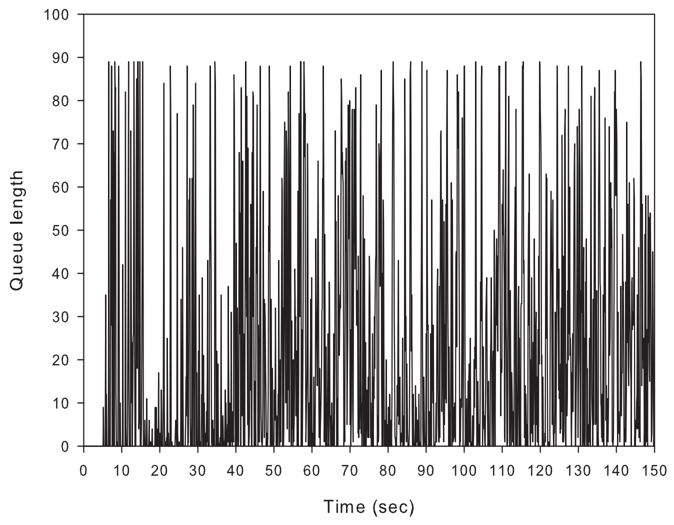

Fig. 2. The queue size process in the mild congestion scenario with the DT algorithm.

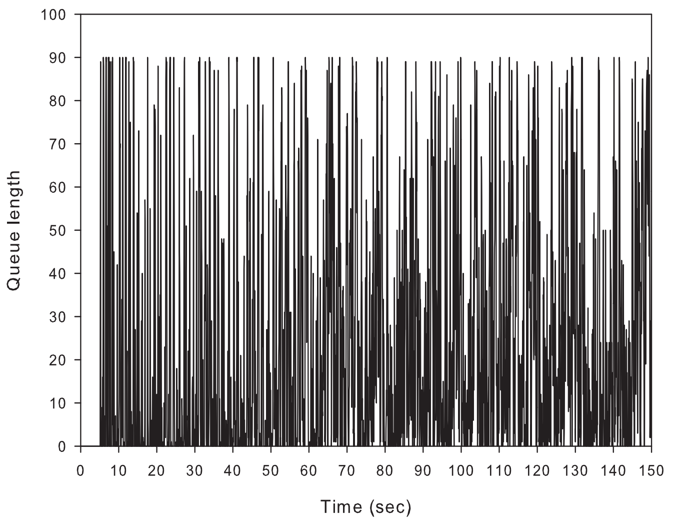

Fig. 4. The queue size process in the mild congestion scenario with the BLUE algorithm.

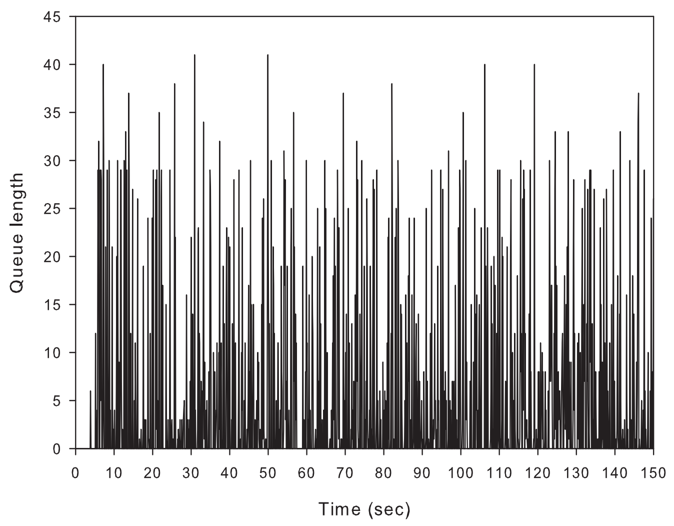

Fig. 3. The queue size process in the mild congestion scenario with the RED algorithm.

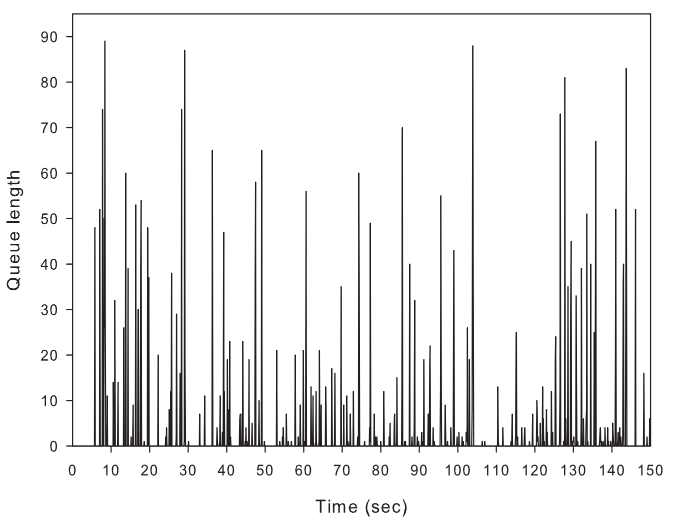

Fig. 5. The queue size process in the mild congestion scenario with the REM algorithm.

we performed a number of experiments in uncongested network using a large buffer (900 packets). The result are shown in Tab. V. Again, none of the algorithms were able to maintain a stable queue size and high throughput. We can observe either a low throughput (RED, PI AVQ, AN-AQM) or a very high deviation of the queue size. Sample queue size processes for a large buffer in the uncongested network are presented in Figs. 15-17. As we can see, especially unstable behaviour is observed for DT and AN-AQM queues.

V. Conclusions

In this paper, using the newest AQM testbed, we have shown the network conditions that are likely to make the queue size unstable. Namely, we demonstrated that, in addition to large propagation delays, low congestion levels and low target queue sizes are especially unfavourable for both the classic and the newest algorithms. These two factors influence the stability independently, and when they are both present, the performance of the network deteriorates drastically due to the queue size instability. In our experiments, in the worst cases, about 75 percent of the network resources was lost, due to the very low throughput of the queueing algorithm (e. g. RED, AN-AQM in Tab. II).

Therefore, new algorithms that can stabilize the queue size under these conditions are needed.

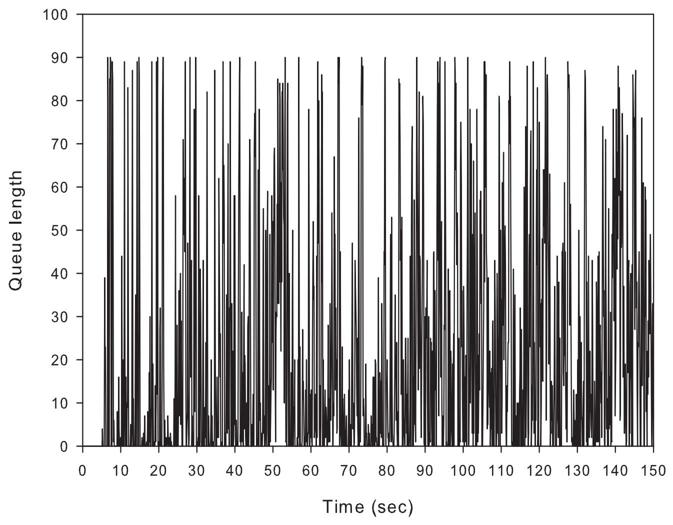

Fig. 6. The queue size process in the mild congestion scenario with the PI algorithm.

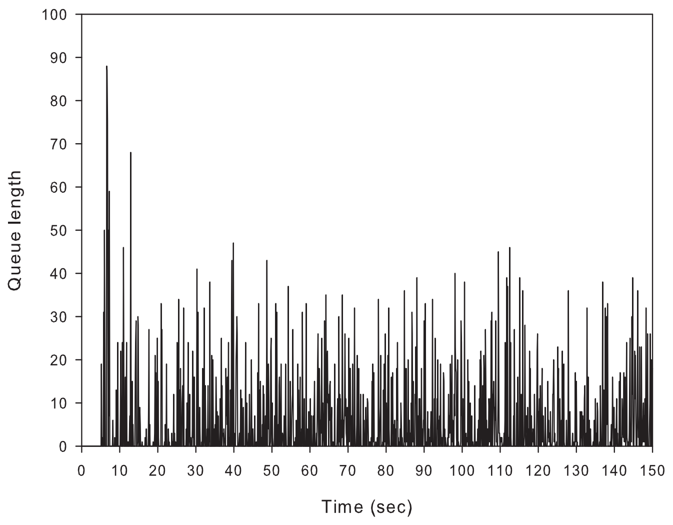

Fig. 8. The queue size process in the mild congestion scenario with the AN-AQM algorithm.

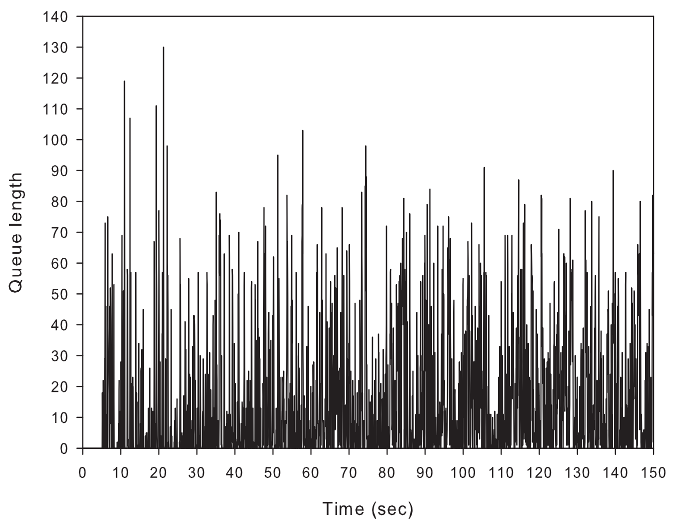

Fig. 7. The queue size process in the mild congestion scenario with the AVQ algorithm.

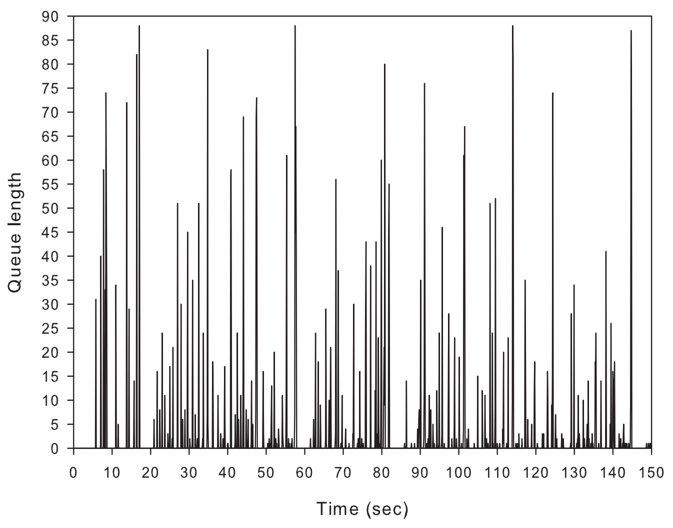

Fig. 9. The queue size process in the uncongested network with the DT algorithm.

Acknowledgements: This is extended version of the paper [1] presented during Applied Computing Conference, Athens, September 27-29, 2009. The work was supported by MNiSW under grant N N516 381134.

REFERENCES

[1] Mrozowski, P.; Chydzinski, A. On the Stability of AQM Algorithms. Proc. of Applied Computing Conference, pp. 276-281, Athens, Sept. 2009.
[2] Floyd, S.; Jacobson, V. Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on Networking, Volume 1, Issue 4, Page(s): 397 – 413, 1993.
[3] Rosolen, V.; Bonaventure, O., and Leduc, G. A RED discard strategy for ATM networks and its performance evaluation with TCP/IP traffic. SIGCOMM Comput. Commun. Rev. 29, 3, Jul. 1999.
[4] Floyd, S.; Gummadi, R.; Shenker, S. Adaptive RED: An algorithm for increasing the robustness of RED. Tech. Rep.: ACIRI, 2001.
[5] Athuraliya, S.; Low, S. H.; Li, V. H.; Qinghe Yin. REM: active queue management, IEEE Network. On page(s): 48-53, Volume: 15, Issue: 3, May 2001.
[6] Wu, W.; Ren, Y.; Shan, X. A self-configuring PI controller for active queue management. In Asia-Pacific Conference on Communications (APCC), Japan, 2001.
[7] Hollot, C. V.; Misra, V.; Towsley, D.; Weibo Gong. Analysis and design of controllers for AQM routers supporting TCP flows, IEEE Transactions on Automatic Control. On page(s): 945–959, Volume: 47, Issue: 6, Jun 2002.
[8] Feng, W.; Shin, K. G.; Kandlur, D. D.; Saha, D. The BLUE active queue management algorithms. IEEE/ACM Transactions on Networking. Page(s): 513 – 528, Volume: 10, Issue: 4, Aug 2002.
[9] Wydrowski, B. and Zukerman, M. GREEN: an active queue management algorithm for a self managed Internet, in: Proceeding of IEEE International Conference on Communications ICC2002, vol. 4, pp.

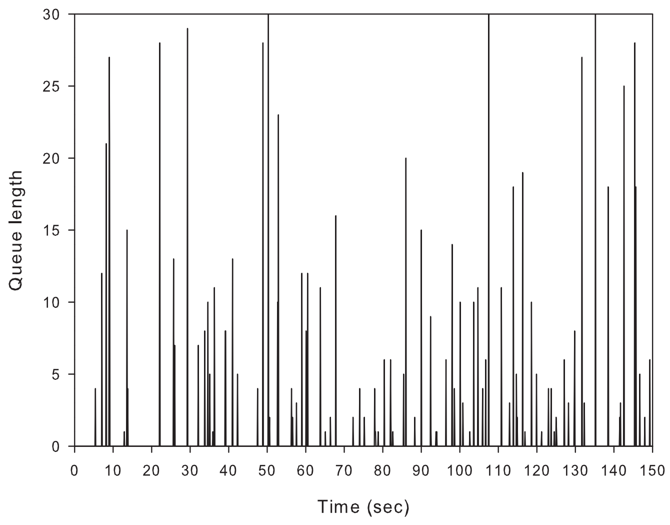

Fig. 10. The queue size process in the uncongested network with the RED algorithm.

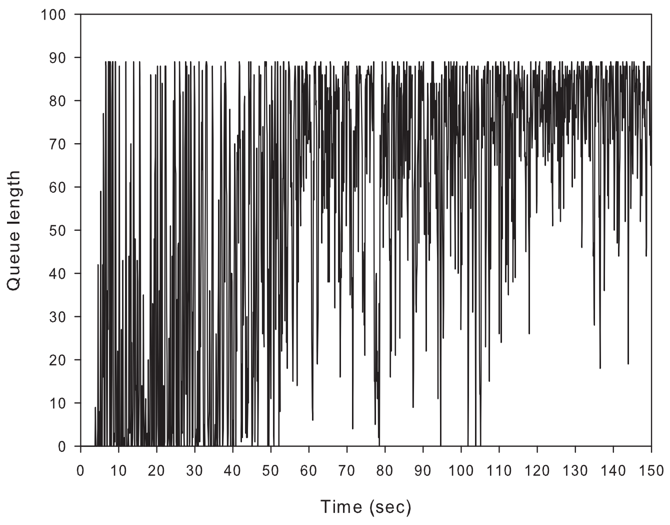

Fig. 12. The queue size process in the heavy congestion scenario with the DT algorithm.

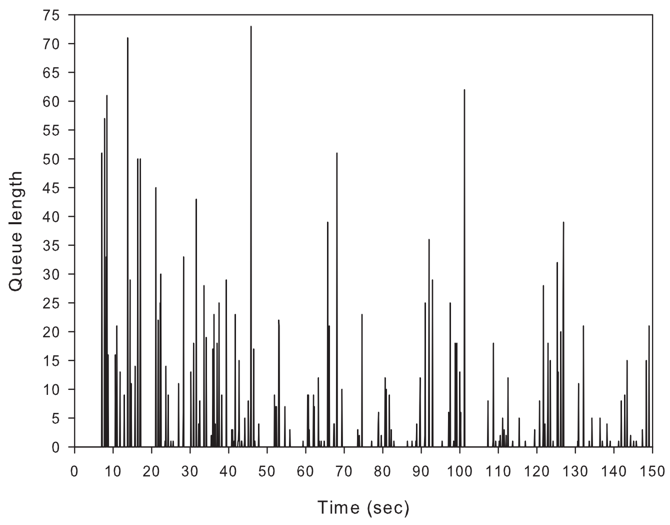

Fig. 11. The queue size process in the uncongested network with the AN-AQM algorithm.

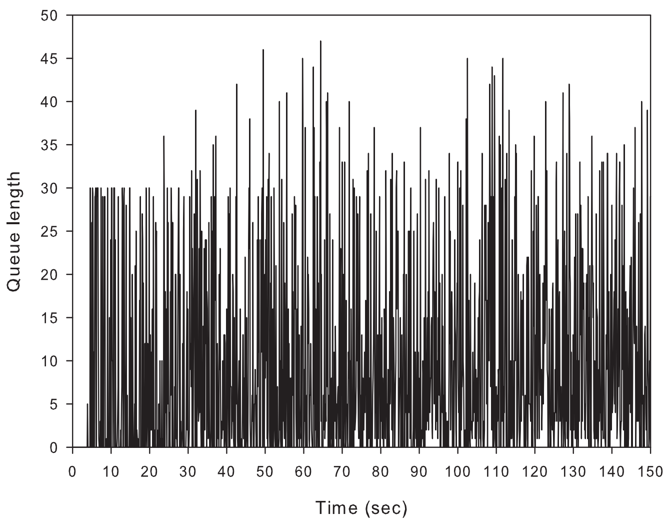

Fig. 13. The queue size process in the heavy congestion scenario with the RED algorithm.

2368-2372, April, 2002.

[10] Heying, Z.; Liu, B. and Wenhua, D. Design of a robust active queue management algorithm based on feedback compensation, in: Proceedings of ACM/SIGCOMM 2003, pp. 277-285. 2002.

[11] Aweya J. I; Ouellette M.; Montuno D.Y. Multi-level active queue management with dynamic thresholds. Computer Communications. Volume 25, Number 8, pp. 756–771, 2002.

[12] Fatta; G.; Hoffmann, F.; Re, G. L.; Urso, A. 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. On page(s): 313-324, Volume: 33, Issue: 3, Aug. 2003.

[13] Kunniyur, S. S.; Srikant, R. An adaptive virtual queue (AVQ) algorithm for active queue management. IEEE/ACM Transactions on Networking. Page(s): 286–299, Volume: 12, Issue: 2, April 2004.

[14] Chatranona, G., Labradorb, M.A. and Banerjee, S., A survey of TCP-friendly router-based AQM schemes. Computer Communications. v27. pp. 1424–1440, 2004.

[15] Bohacek, S.; Shah, K.; Arce, G.R.; Davis, M. Signal processing challenges in active queue management. IEEE Signal Processing Magazine, Volume 21, Issue 5, Pages: 69–79, Sept. 2004.

[16] Ren, F.; Lin, C. and Wei, B. A robust active queue management algorithm in large delay networks. Computer Communications 28(5): Pages: 485–493, 2005.

[17] Lakshmikantha, A.; Beck, C. L.; Srikant, R. Robustness of real and virtual queue-based active queue management schemes, IEEE/ACM Transactions on Networking. On page(s): 81– 93, Volume: 13, Issue: 1, Feb. 2005.

[18] Hong, Y.; Yang, O. W. W. Adaptive AQM controllers for IP routers with a heuristic monitor on TCP flows: Research Articles, International Journal of Communication Systems, v.19 n.1, pp.17–38, 2006.

[19] Chang, X.; Muppala, J. K. A stable queue-based adaptive controller for improving AQM performance, Computer Networks: The International Journal of Computer and Telecommunications Networking, v.50 n.13, pp. 2204–2224, 2006.

[20] Hayes, M. J.; Alavi, S. M. M.; Vandeven, P. Robust active queue management using a quantitative feedback theory based loop-shaping

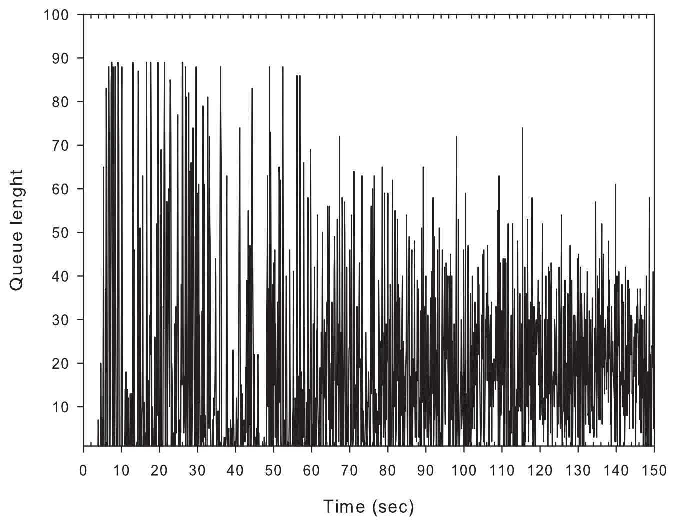

Fig. 14. The queue size process in the heavy congestion scenario with the AN-AQM algorithm.

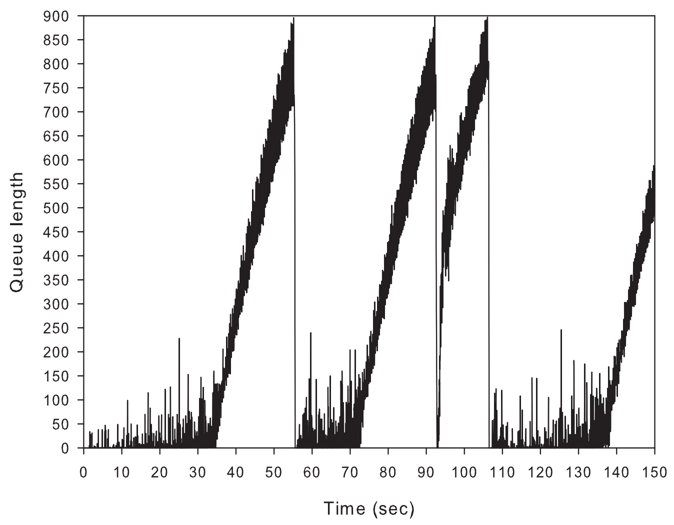

Fig. 15. The queue size process for the large buffer (900 pkts), uncongested network and the DT algorithm.

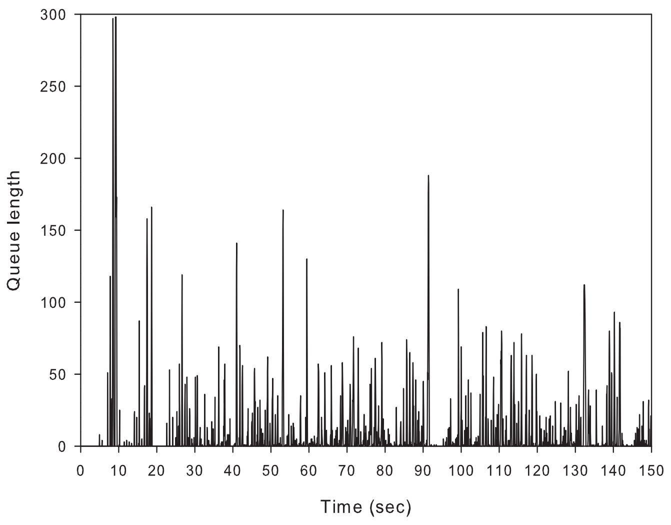

Fig. 16. The queue size process for the large buffer (900 pkts), uncongested network and the RED algorithm.

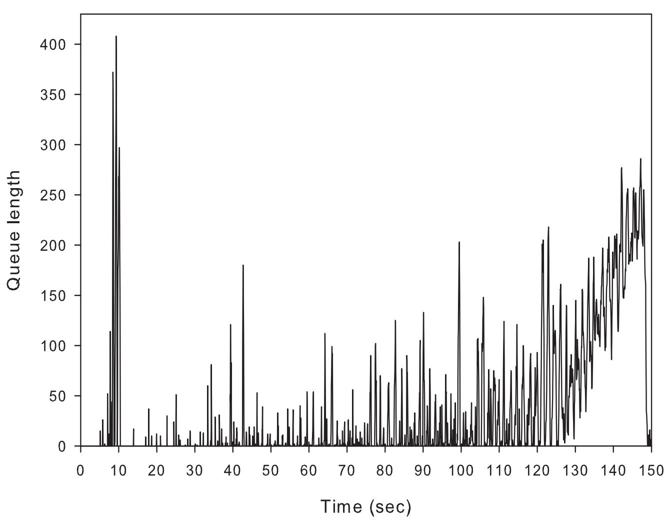

Fig. 17. The queue size process for the large buffer (900 pkts), uncongested network and the AN-AQM algorithm.

framework, in: Proc. American Control Conference, pp. 3077–3082, 2007.
[21] Wang, C.; Liu, J.; Li, B. Sohraby, K; Hou, T. LRED: A Robust and Responsive AQM Algorithm Using Packet Loss Ratio Measurement. IEEE Transactions on Parallel and Distributed Systems, v.18 n.1, pp.29–43, 2007.
[22] Hariri, B.; Sadati, N. NN-RED: An AQM mechanism based on neural networks. Electronics Letters. Volume: 43, Issue: 19, On page(s): 1053–1055, 2007.
[23] Sun, J. and Zukerman, M. An Adaptive Neuron AQM for a Stable Internet, Proc. Networking'07, LNCS 4479, 2007.
[24] Chu, H. Y.; Tsai, K. H.; Chang, W. J. Fuzzy control of active queue management routers for transmission control protocol networks via time-delay affine Takagi-Sugeno fuzzy models. International Journal of Innovative Computing, Information and Control. v4. 291-312, 2008.
[25] Chrost, L. and Chydzinski, A. Proc. of International Conference on Evolving Internet (INTERNET'09), pp. 113-118, Cannes, Aug. 2009.
[26] http://www.isi.edu/nsnam/ns/

Pawel Mrozowski received the bachelor of science degree and MSc from the University in Bielsko-Biala and the Jagiellonian University of Krakow, respectively. Currently he is a PhD student at the Silesian University of Technology. His main research interests include active queue management and network simulations. He is interested also in other areas, e. g. video compression, 3D visualizations and design, implementation and management of the enterprise hardware/software solutions. He is working also as IT specialist in hi-tech manufacturing plant, supporting team of CAD/CAM engineers.

Andrzej Chydzinski received his MS (in applied mathematics), PhD (in computer science) and habilitation (in computer science) degrees from the Silesian University of Technology, Gliwice, Poland, in 1997, 2002 and 2008, respectively. He is currently a professor in the Institute of Computer Sciences of this university. His academic and professional interests include mathematical modeling and simulation of computer networks and queueing systems. Dr Chydzinski has an established record of publications with two books and about sixty journal and conference papers.