Theoretical and Applied Informatics
ISSN 1896–5334
Vol.23 (2011), no. 1
pp. 3–20
DOI: 10.2478/v10179-011-0001-0

On the search of efficient AQM for large delay networks

AGNIESZKA BRACHMAN, ŁUKASZ CHROST

Institute of Informatics
Silesian University of Technology
ul. Akadmicka 16, Gliwice, Poland

Received 2 October 2010, Revised 13 December 2010, Accepted 2 December 2010

Abstract: The main idea of all Active Queue Management algorithms, is to notify the TCP sender about incoming congestion by dropping packets, to prevent from the buffer overflow, and its negative consequences. However, most AQM algorithms proposed so far, neglect the impact of the high speed and long delay links. As a result, the algorithms' efficiency, in terms of throughput and/or queue stability, is usually significantly decreased. The contribution of this paper is twofold. First of all, the performance of the well known AQM algorithms in high speed and long delay scenarios is evaluated and compared. Secondly, a new AQM algorithm is proposed, to improve the throughput in the large delay scenarios and to exclude the usage of random number generator.

Keywords: active queue management, large delay network, throughput

1. Introduction

In the current Internet, congestion control is accomplished by TCP sender, who reacts to packets losses by reducing the transmission rate. When packet loss is caused due to the buffer overflow, it is also an indication of a network congestion. The Internet congestion occurrence has many disadvantages such as wasting network resources and long delays in packet delivery. Proactive congestion management applied at routers could successfully alleviate these problems. Therefore, the main idea of all Active Queue Management algorithms, is to notify the TCP sender about incoming congestion by dropping packets, to prevent from the buffer overflow. Indication of incoming congestion protects from multiple packets losses, and allows decreasing queuing delays.

AQM is a very active research area in the Internet community. The most commonly referred AQM scheme is RED (Random Early Detection) [14], which is also the only algorithm recommended for AQM in RFC 2309 [4]. Since the introduction of RED, many

enhancements have been proposed to improve the link utilization, fairness, dropping rate and stability. Nevertheless, two main problems remain with the family of RED algorithms. First of all, it is difficult to configure their parameters, to provide an universal configuration for all traffic scenarios. Secondly, they relay on the long term queue size averages, which makes them unstable in the face of varying arrival and departure rates. The alternative family of algorithms is called load-based AQMs. This group of algorithms aims at handling with varying intensity of network traffic. The family embraces algorithms such as PI, REM, AVQ, ANAQM.

Queue management is a major element of the network infrastructure, and the high link utilization is one of the most important requirements, concerning Active Queue Management (AQM) algorithms, applied at routers. High link utilization, along with short and low varying queuing delays are provided with low and stable queue in the first place. Satisfying the stringent delay and jitter requirements, is very important for the Quality of Service-sensitive, multimedia applications and other real-time interactive applications. There is also more capacity to accommodate sudden traffic bursts without dropping packets. Avoiding oscillations is equivalent to proper response to perturbations, caused by emerging and retiring flows as well as to varying transmission rates of TCP sources. Achieving high link utilization is beneficial for network providers and end users.

Many proposed AQM algorithms neglect the impact of the high speed and large delay links. Large delays, i.e. large round-trip times (RTT), cause unstable operation, which results in huge oscillations, that cause the low link utilization and introduce the delay jitter. The contribution of this paper is twofold. First of all, the well known AQM algorithms are evaluated and compared under large delay scenarios. The second objective is to propose a new queue management algorithm, to achieve a very good performance. A new method called LINDROP is proposed. LINDROP uses a non decreasing, linear function f applied at the average queue size, to estimate the dropping rate of incoming packets. The algorithm is intended to improve the bottleneck link throughput in long delay scenarios. The proposed solution is validated, and its performance is compared with other AQM schemes through simulation. The metrics used to assess the performance include: a) the bottleneck link utilization, b) the average queue length.

The rest of the paper is organized as follows. Section 2. reviews the related research work. In section 3., the LINDROP congestion controller is introduced, along with some guidelines for parameter settings. The performance of LINDROP is compared with that of DT, RED, ARED, REM, PI and ANAQM in Section 4. Through simulation, we evaluate the impact of large delays on the performance of LINDROP and other popular AQMs. Finally, we conclude our research in Section 5.

2. Background and related work

Congestion control heavily depends on end-to-end mechanisms at the transport layer. TCP sender probes the available bandwidth, and adjusts the transmission rate accordingly. The commonly used implementations of TCP (TCP New Reno, TCP SACK) trigger the congestion response algorithm after the receipt of three duplicated ACKs or a timeout of the retransmission timer. If this TCP response is caused by buffer cramming, it results in significant reduction of transmission rate, which results in reducing the link utilization. Since congestion occurs at the routers, it is preferable, that the routers play active role in preventing buffer overflow.

In the present Internet, AQM algorithms are not widely deployed. As a matter of fact, despite many flaws, the FIFO queues are the most commonly encountered solution. FIFO queue with the drop tail strategy (referred as Drop Tail algorithm) are simple to implement, however they are prone to global synchronization of TCP sources, which results in bandwidth underutilization, relatively high transmission delays and high packet drop rates. Nevertheless, the answer why AQMs have not been implemented so far, is simple. Their designers put special emphasis on resolving different problems, but there is lack of a comprehensive solution that can deal with the various conditions. AQMs are very successful in the scenarios they were designed for, however usually suffer performance degradation when the conditions vary.

AQMs are highly supportive for the congestion control and are also essential, when it comes to Quality of Service. The QoS requirements embrace low packet loss rate, short and stable queuing delays from flow perspective as well as the high link utilization and the fair link sharing from more global perspective. AQMs should detect incoming congestion. For this purpose, they can use the following information: per aggregate or per flow arrival rate, the instantaneous or average queue size or combination of these variables. The AQM algorithms can be divided, with regard to control type, into load-based and queue-based.

Almost all existing AQM algorithms neglect the impact of large delay on the performance of AQM. The problem is addressed in [27, 22]. Ren et al. design a robust AQM controller with the ability to compensate for delay, using internal mode control theory. The main limitation of this algorithm is that they assume the RTT as known and constant, which is impossible for real network environment. Solution proposed in [27] has similar shortcoming. For their model, authors adopted the internal model control theory, to restrict the negative impact on algorithm's performance, which is a result of the queue instability caused by large delays. The conclusion is that AQM algorithms designed for queue stabilization perform well only if they know the mean or maximum RTT. Estimating RTT is not a simple issue, the problem is addressed in [19, 20, 17] and the solution is still not satisfying.

The first introduced AQM was RED. It can prevent global synchronization, reduce packet loss rates, and minimize bias against busty sources. Since then, numerous variants of RED and novel schemes have been proposed including: adaptive RED (ARED) [12], stabilized RED (SRED) [19], gentle RED (GRED) [13] and many others [26, 21, 15].

RED uses the Exponentially Weighted Moving Average (EWMA) of the queue size – avg. If avg is smaller than the min_{th} threshold, all the packets are enqueued, if $avg > max_{th}$ all the packets are dropped. When avg is between the two thresholds, packets are dropped with linearly increasing probability. The control parameters include the weight parameter w_q used for average queue size calculations, the two threshold parameters (min_{th}, max_{th}) and the maximum drop probability p_{max} . The guidelines concerning RED tuning, suggest using small weight parameter w_q. It makes RED sensitive to the different congestion characteristics, which results in instantaneous queue oscillations, and ipso facto low bandwidth utilization. Problem of proper RED configuration is widely described in [10, 7, 25], the conclusion is, that there is no universal configuration, that provides good efficiency of the algorithm. The problem of proper RED configuration is described in [28]. Authors of [21] analytically investigate the shape of non-linear RED drop function and compare obtained results with other versions of RED. Similar approach is presented in [15]. Guo et al. believe that the good performance of most of the AQM is obtained, by modifying drop probability function. They introduce the dynamical model of exponential RED (E-RED) and TCP Reno sources. They suggest the condition for E-RED to achieve good performance under different scenarios, for example, round trip time independent. Despite obtaining rather good results, authors admit that E-RED oscillates more severely as delay increases. In [7], Christiansen et al. concluded that adapting RED for efficient operation is difficult. Firoiu and Borden [11] presented the relationship between queue size and dropping probability, which illustrates reason, why RED queue acts unstable. Network load i.e. number of flows and the physical link throughput are main causes for RED instability. Stabilized RED (SRED) [19] uses a list called zombie, to estimate the number of active TCP connections, and update the dropping probability accordingly. Although its insensitivity to the network load is increased, the estimation about active sessions is too coarse. Another modifications of RED is SPI-RED [28]. SPI-RED is based on a self-tuning proportional and integral feedback controller, which additionally considers the past queue lengths during RTT, to smooth the impact of short lived flows. Algorithm has not been studied in large delay scenario.

The adaptive RED (ARED) uses link throughput to adjust w_q and p_{max}. The ARED keeps the average queue size halfway between the two thresholds (min_{th}, max_{th}), using the additive-increase-multiplicative-decrease policy (AIMD). The results obtained with ARED are more satisfying, than the results achieved using RED, however the queue oscillation and low link utilization are still a problem, especially in low congestion scenarios.

Except for RED, there are many other AQM techniques, namely PI, REM, AVQ, ANAQM ([3, 6, 16, 18, 24, 9]). The PI (Proportional Integral) controller uses the knowledge of the instantaneous queue size, to maintain the steady value of the queue size, to the specified reference value. It applies the well known idea from the control theory: a) the proportional (P)-control that uses the instantaneous queue size, to improve responsiveness and b) the Integral (I-control), to stabilize the queue size around the reference point, regardless of the load level. The main flaws of the PI controller are [5]: a) lags introduced by the integral control, b) fixed controller gains related to the network conditions, which results in either large oscillation but fast responsiveness or small oscillation however at the cost of sluggish response to the traffic changes. A stable queue-based adaptive proportional-integral controller (Q-SAPI) is introduced in [5]. Through simulation authors depict that the algorithm maintains its steady state performance over a wide range of uncertainties in round trip time and the number of active flows. The main flaw of this algorithm is that it needs some global information concerning the number of active flows and the maximum RTT, therefore it heavily depends on the correctness of estimating algorithms.

AVQ and REM belong to the AQM algorithms with optimization-based approach i.e. TCP/AQM dynamics is formulated from an optimization standpoint directed to obtaining optimal source rates. REM (Random Exponential Marking) tries to regulate the queue length to a desired value q_{ref} as well. It updates periodically the probability of dropping packets with step γ. The probability is calculated using parameter Φ. AVQ (Adaptive Virtual Queue) uses the idea of the virtual queue. A router with AVQ algorithm maintains a virtual queue, whose capacity is less than or equal to the capacity of the link, c. To configure AVQ, the two parameters are required, namely γ and α which are the desired utilization and damping factor respectively. Optimization-based approaches lead to steady-state equilibrium [18].

AN-AQM is rather novel Active Queue Management algorithm aiming at fast queue length stabilization, which is crucial for maintaining rational drop rate to buffer delay ratio. Various AQM strategies purport to reduce packet delay in routers through advanced management of the internal packet queue. Queue stabilization impacts not only the retardation of network traffic, but also other parameters such as fairness and bandwidth utilization.

AN-AQM is based on estimation of dropping probability of incoming packets by an adaptive neuron. The neuron uses both integrated Hebbian Learning and Supervised Learning based on two main error factors – queue length error ($e(k)$) and normalized rate error ($y(k)$) described respectively by following equations:

$$e(k) = q(k) - Q_t \tag{1}$$

$$y(k) = \frac{r(k)}{C} - 1 \tag{2}$$

where $q(k)$ is the queue length and Q_t is target queue length, $r(k)$ is the input rate at the bottleneck link and C is the capacity of the bottleneck link. There are 6 inputs of the AN-AQM scheme embracing both the error factors and their historical values. The values of parameters describing the adaptive neuron and the learning strategies used for latter simulations are the same as the ones proposed in [24].

3. The proposed solution

The idea of the proposed approach is to drop the incoming packets, with the estimated drop rate (dr). The drop rate denotes, that every dr-th packet should be dropped. The dr variable is continuously adapted, according to the varying conditions.

The initial value of dr is set to dr_{init}. Subsequently, a non decreasing function f has been chosen and with every incoming packet, the $f(avg)$ is calculated, where avg denotes the Exponentially Weighted Moving Average (EWMA) of the queue size. The alterations of the f values induce changes of dr variable. The newly calculated value of f is compared, to the value calculated at the previous packet arrival. If the $\Delta f > 0$, the dr is decreased, otherwise it is increased. Therefore when the function f grows, the number of packets' drops increases in the time course. The queue overflow doesn't influence the dropping scheme. The dr_{max} and dr_{min} are the maximum and minimum values of the dr. The dropping rate changes exponentially. It grows with the growth factor n_{pd}, where $n_{pd} > 1$ and decreases with the decay factor n_{pu}, where $0 < n_{pu} < 1$. The n_{pu} mainly influences the average queue size, higher value results in shorter queue. The n_{pu} slightly influences the queue stability.

A linear function, depicted in Fig. 1, has been selected for proposed LINDROP algorithm. The pseudo code of LINDROP algorithm is presented in the listing below. The values of all parameters were set, basing on the simulation results. The simulations were performed for LAN topology (2ms delay on bottleneck link), under low congestion scenario (for details see section 4.2.). The initial value of dropping rate drinit is set to 500000, $dr_{max} = 1000000$ and $dr_{min} = 5$. Setting proper value of avg_{min} and avg_{max} is also very important. If avg_{min} is to high, the queue oscillates between 0 and avg_{min}, which results in low bandwidth utilization. Higher avg_{max} induces slower changes in dropping rate, which also causes queue instability. Parameter avg_{min} is set to 0. The results for different settings of avg_{max}, concerning average throughput, average queue size and standard deviation are presented in Fig. 5. The best results are achieved when $avg_{max} >= 30$ and $50 <= avg_{max}$. Simulation results for different values of n_{pu} and n_{pd} are presented in Figures 3 and 4. The results in Fig. 3 concern configuration where $avg_{max} = 150$, results in Fig. 4 are for configuration $avg_{max} = 50$. From these facts we can conclude that the lower value of avg_{max} reduces the influence of n_{pu} and n_{pd}, however it also affects the algorithm stability.

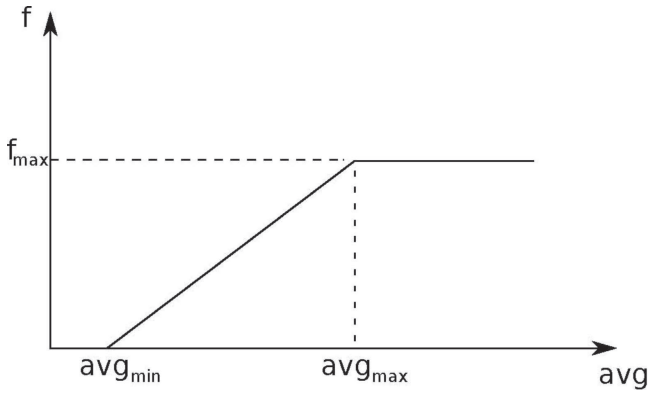


Fig. 1. Dropping function for LINDROP

Algorithm 1 The pseudo code of the LINDROP algorithm

Require: $n_{pd} > 1,\ 0 < n_{pu} < 1,\ 0 <= avg_{min} < avg_{max},\ avg_{max} < BufferSize$

$dr(0) \leftarrow dr_{init}$
$avg \leftarrow 0$
$f(avg) \leftarrow 0$
$n_{drops} = 1$
for for i-th packet **do**
  $avg \leftarrow CalculateEWMA(queue)$
  $f_i \leftarrow f(avg, avg_{min}, avg_{max})$
  if $f_i - f_{i-1} > 0$ **then**
    $dr_i = dr_{i-1} * n_{pu}$
  else
    $dr_i = dr_{i-1} * n_{pd}$
  end if
  if $dr_i > dr_{max}$ **then**
    $dr_i = dr_{max}$
  else if $dr_i < dr_{min}$ **then**
    $dr_i = dr_{min}$
  end if
  if $n_{drops} > dr_i$ **then**
    $DropPacket(packets(i))$
    $n_{drops} = 1$
  else
    $n_{drops} + +$
  end if
end for

URL	Average RTT [ms]
Europe (Oxford - www.oxford.ox.ac.uk)	47.521
Europe (Paris - www.paris-sorbonne.fr)	58.395
Europe (Moscow - www.msu.ru)	79.782
USA (Chicago - www.ccc.edu)	147.135
USA (California - www.ucla.edu)	216.234
Asia (Hong Kong - www.cuhk.edu.hk)	310.338
Asia (Tokyo - www.u-tokyo.ac.jp)	326.622
Australia (Sydney - www.usyd.edu.au)	354.550

Table 1. Measured mean RTT values

4. Performance of AQM schemes in large delay networks

4.1. Distribution of RTT

According to the Internet measurements presented in [23], roughly 75 – 90% of flows have RTTs less than 200 ms. In [17] authors show that the average RTT is distributed around 180 ms. Table 1 shows the round trip time to different overseas and continental websites. Measurements were performed using ping tool from IP address 157.158.55.120 (Poland). Rather large delays have been selected to simulate the overseas transmission rates.

4.2. Simulation setup

The ns-2 simulator was used to study AQM performance [1], release 2.34. The single bottleneck, dumbbell topology is used, see Fig. 2). All nodes are connected to the routers with 100Mbps links. The propagation delays vary according to the list below:

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

The link from router RA to router RB is a bottleneck for each connection. The link has the bitrate 100 Mbps. The mean distribution of RTT for all possible connections is 100ms without the delay on the bottleneck link. Three values of the bottleneck link delay are used, namely 2ms, 100ms and 300ms. First scenario, further reffered as LAN scenario, is a basic scenario for an AQM comparison. Second and third scenarios are

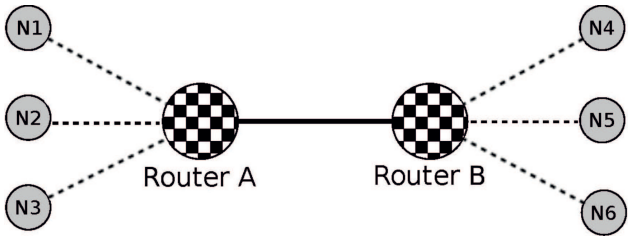


Fig. 2. A dumbbell topology

considered large delay and are further referred as WAN_100 and WAN_300 scenario. The longest RTT are 300ms and 700ms accordingly.

Three traffic scenarios described in [8] are used:

- The uncongested network scenario – 10 TCP connections,
- The moderate congestion scenario – 100 TCP connections,
- The heavy congestion scenario – 1000 TCP connections,

The TCP SACK senders are located in nodes N1-N3 and transmit data to nodes N4-N6. All nine transmission paths are used, TCP connections are uniformly distributed among the transmission paths. 90% of TCP connections use 1500 bytes long packets, the remaining 10% use 536 bytes long packets. All 536-bytes-long connections and 75% of the 1500-bytes-long connections are FTP flows, the flows are active throughout the whole simulation. The remaining 25% flows are short lived flows imitating the HTTP traffic. The short lived flows are configured as follows. Each flow is initiated randomly according to the Poisson process and is assigned the fixed number of bytes to transmit. The number of bytes are generated using Pareto distribution with the average value of 50kB and the shape parameter of 1.3. The Poisson process rates are set to 12.5, 1.25 and 0.125 in considered scenarios.

Additionally, to simulate the reverse direction traffic, UDP (CBR) flows in backward direction, with 1000-bytes long packets are used. Their sending rate is set to fulfil 10% of the available backward bandwidth.

Six AQM are evaluated along with classic FIFO queue referred as DT. The AQM algorithms are: RED, ARED, PI, REM, AVQ, ANAQM. Simulation scenarios' parameters are set according to [8], which is based on [2]. The AQM configuration parameters are set according to Table 2. All AQM algorithms are configured to maintain queue size around 100 packets in LAN secnario, namely with 2ms delay on bottleneck link.

The following metrics on the bottleneck link are collected: the link utilization and the average queue size.

DT	Buffer Size = 100
RED	Buffer Size = 300, $min_{th} = 50$, $max_{th} = 150$
ARED	Buffer Size = 300, $min_{th} = 50$, $max_{th} = 150$
AVQ	Buffer Size = 300, $\gamma = 0.98$
PI	Buffer Size = 300, $q_{ref} = 100$
REM	Buffer Size = 300, $q_{ref} = 100$
ANAQM	Buffer Size = 300, Target ratio = 0.3
LINDROP	Buffer Size = 300, $avg_{min} = 0, avg_{max} = 50, n_{pu} = 0.98, n_{pd} = 1.05$

Table 2. Simulation parameters

		Congestion		
	AQM	low	moderate	heavy
LAN	DT	92.75	99.86	99.95
	RED	90.54	99.64	99.91
	ARED	99.19	99.97	99.94
	AVQ	75.90	99.96	99.98
	PI	95.60	99.95	99.98
	REM	98.92	99.98	99.98
	ANAQM	89.39	99.73	99.93
	LINDROP	97.33	99.96	99.98

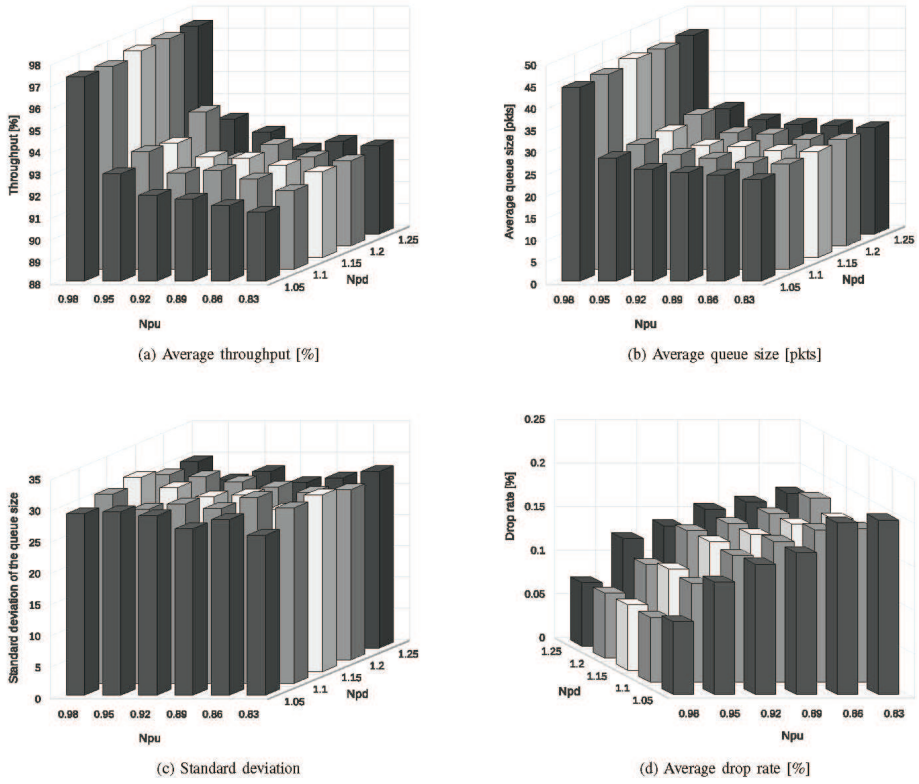
Table 3. The average throughput on bottleneck link in LAN scenarios [Mbps]

4.3. Simulation results for LAN scenarios

Firstly, we evaluate the proposed solution under the LAN scenario, to verify the LINDROP properness. The average throughput of the bottleneck link for LAN topology and all traffic scenario are gather in Table 3.

The throughput concerns the bottleneck link, the number of bytes departed at the router A throughout the simulation course was measured. Results are presented in Megabits per second (Mbps). The average queue size was estimated on the router B.

In the low congestion scenario, almost all AQM algorithms, along with DT, achieve high link utilization, i.e 90% and more. The exception is AVQ, which uses 75% of the available bandwidth. The more connections, the better performance, in terms of the achieved throughput. In moderate and heavy congestion scenarios, the differences are negligible for all algorithms, in terms of the link utilization.



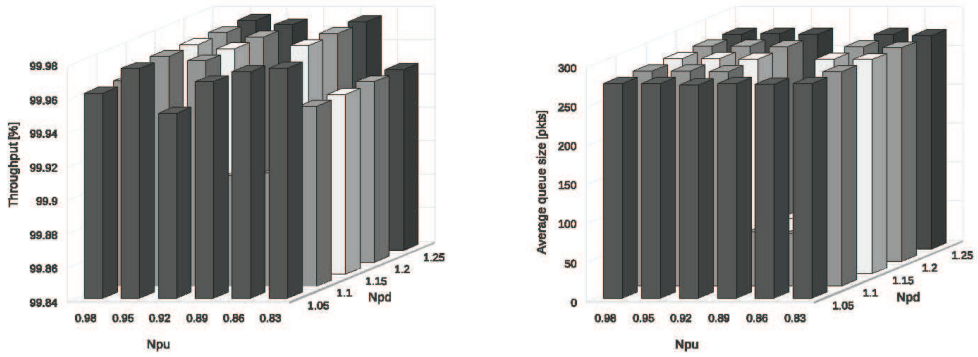
(a) Average throughput [%]

(b) Average queue size [pkts]

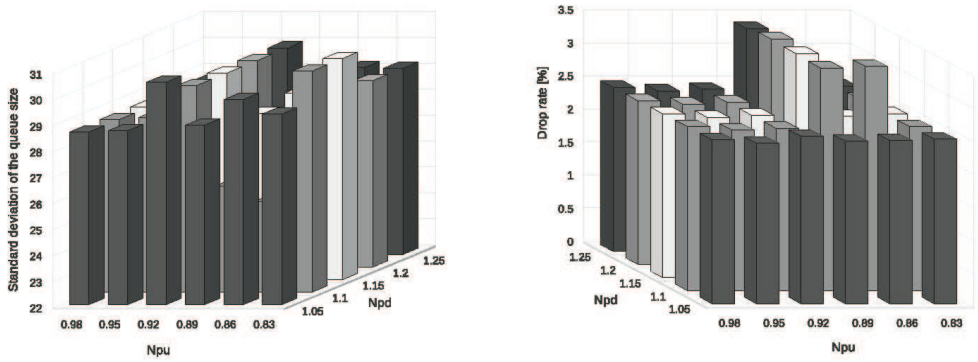
(c) Standard deviation

(d) Average drop rate [%]

Fig. 3. The influence of n_{pu} and n_{pd} on the performance of LINDROP in low congestion scenario, $max_{th} = 150$

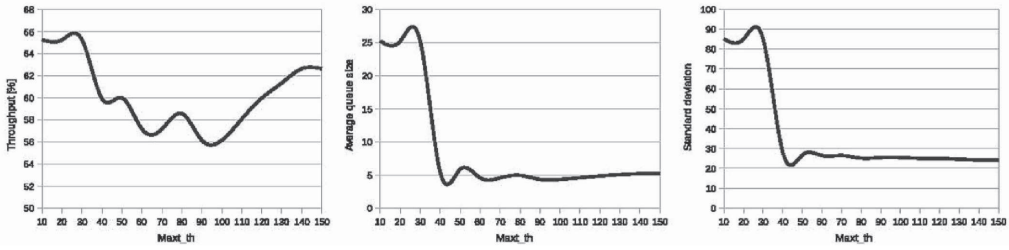


(a) Average throughput [%] (b) Average queue size [pkts]



(c) Standard deviation (d) Average drop rate [%]

Fig. 4. The influence of n_{pu} and n_{pd} on the performance of LINDROP in low congestion scenario, $max_{th} = 50$



(a) Average throughput [%] (b) Average queue size [pkts] (c) Standard deviation

Fig. 5. The influence of th_{max} on the performance of LINDROP, $n_{pu} = 0.98$, $n_{pd} = 1.05$

		Congestion					Congestion		
	AQM	low	moderate	heavy		AQM	low	moderate	heavy
WAN_100	DT	32.95	84.04	92.51	WAN_300	DT	15.32	45.56	81.42
	RED	40.36	80.71	92.11		RED	17.16	50.44	79.68
	ARED	64.04	91.34	92.97		ARED	**20.09**	63.88	84.98
	AVQ	63.63	91.51	93.62		AVQ	19.24	63.50	86.56
	PI	40.71	91.09	93.61		PI	16.21	56.99	86.35
	REM	**64.11**	91.35	93.57		REM	18.85	62.52	86.99
	ANAQM	33.70	80.04	92.26		ANAQM	17.22	42.83	80.45
	LINDROP	59.97	**95.64**	**98.19**		LINDROP	**20.00**	**66.74**	**90.89**

Table 4. The average throughput on bottleneck link in WAN scenarios [Mbps]

		Congestion					Congestion		
	AQM	low	moderate	heavy		AQM	low	moderate	heavy
WAN_100	DT	1.40	24.01	77.00	WAN_300	DT	0.37	4.41	44.04
	RED	2.06	20.53	75.29		RED	0.48	6.85	39.18
	ARED	18.79	97.57	106.74		ARED	0.68	31.78	111.36
	AVQ	19.76	143.57	267.88		AVQ	0.60	28.98	169.05
	PI	2.00	82.10	161.37		PI	0.45	13.05	108.61
	REM	20.22	131.72	184.91		REM	0.63	28.80	159.40
	ANAQM	1.51	17.66	66.82		ANAQM	0.46	3.85	35.90
	LINDROP	5.41	94.06	282.66		LINDROP	1.38	20.52	138.98

Table 5. The average queue size in WAN scenarios [pkts]

4.4. Simulation results for WAN scenarios

The simulation results for both WAN topologies and all traffic scenario are gather in the following tables:

- the average throughput of the bottleneck link (Table 4),
- the average queue size at the bottleneck link (Table 5),

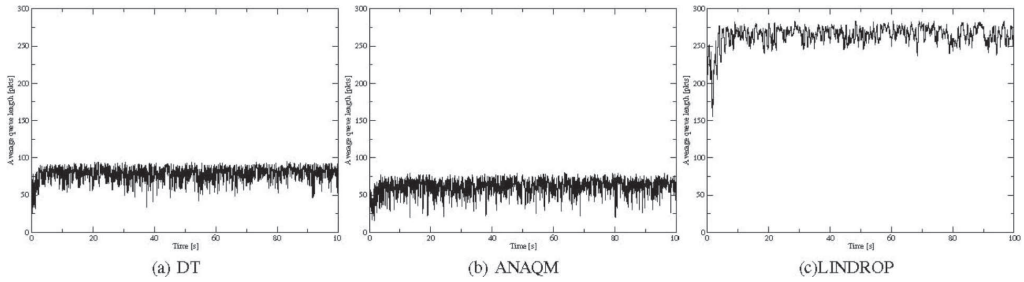
Best results in every congestion scenario are featured. The throughput concerns the bottleneck link, the number of bytes departed at the router A throughout the simulation course was measured. Results are presented in Megabits per second (Mbps). The average queue size was estimated on the router B.

It is striking, that for large delay networks, link utilization is rather low, especially in low congestion cases. In WAN_300 scenario, none of the AQM algorithm is able to utilize the available bandwidth more than 20% and in WAN_100 scenario, the link utilization is around 50-60%. In WAN_100 scenario with 10 active connections, the best link utilization (around 60%) is achieved when ARED, AVQ, REM and LINDROP is used. The remaining algorithms have throughput between 30 and 40%. When the delay at the bottleneck link grows, the differences are no longer so visible. When the number of connection grows, the link utilization gets better, however when the bottleneck

link delay is 300ms, the link utilization is only 80-90% even under heavy congestion scenario. When the number of active connections increases, LINDROP achieves the highest throughput among all algorithms. The results are up to 5% better than for any other algorithms. In large delay WAN_300 scenario, regardless of the congestion intensity, LINDROP provides the highest link utilization.

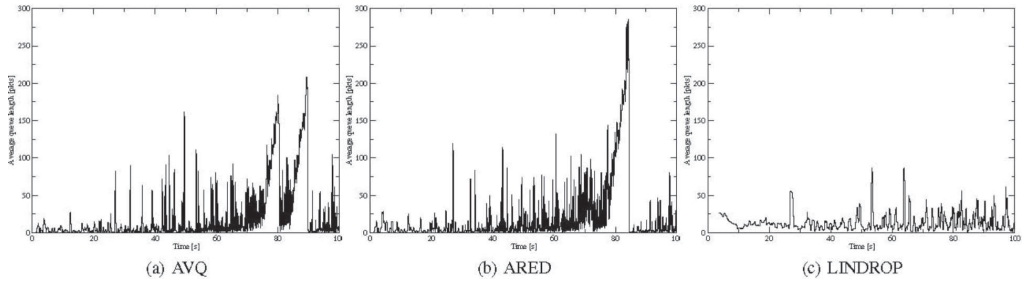
LINDROP maintains low average queue size only in low congestion scenarios. When the number of connections increases, the average queue size grows. In WAN_100 and low congestion level scenario, the average queue size of LINDROP is four times smaller, than for other AQMs with similar performance. Under heavy congestion scenario, the queue is utilized almost in 100%, which unfortunately introduces large delays. In WAN_300 scenario, the trend is preserved, i.e. the highest congestion level, the higher average queue size of LINDROP algorithm, however in general, the average queue size is smaller in comparison to other AQM algorithms of lower performance.

Comparison of average queue size of LINDROP to average queue size of other, exemplary AQM algorithms is depicted in Figures 6, 7, 8.



(a) DT (b) ANAQM (c)LINDROP

Fig. 6. The average queue size, moderate congestion scenario, LAN topology



(a) AVQ (b) ARED (c) LINDROP

Fig. 7. The average queue size, low congestion scenario, WAN_100 topology

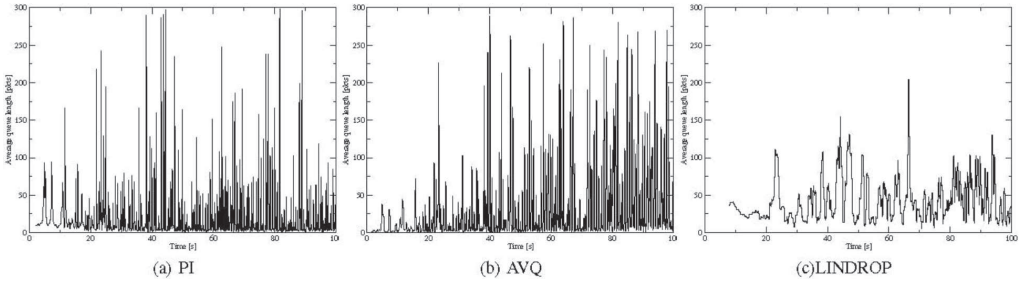


Fig. 8. The average queue size, moderate congestion scenario, WAN_300 topology

5. Conclusions and future work

In this paper, we studied the influence of large delays on the AQMs' performance in terms of the bottleneck link utilization, for most popular AQM algorithms and newly designed LINDROP solution. It has been shown, that the large delays, highly influence the algorithm efficiency and result in significant link underutilization. Moreover, in addition to low congestion level, all algorithms suffer serious performance degradation.

The LINDROP algorithm has been proposed to improve the bandwidth utilization in large delay networks. LINDROP provides high link utilization, in comparison to other solutions and in low congestion scenarios, it maintains short and stable queue. Unfortunately improving the overall throughput, has not been accomplished in all network scenarios so far. Providing good performance in low congestion scenario and large delay network remains an open issue.

Acknowledgements

This work is supported by the Ministry of Science and Higher Education under grant N N516 381134

References

1. The network simulator ns-2.
2. L. L. H. Andrew, S. Floyd, W. Gang: *Common TCP evaluation suite*, 6 Jul 2008.
3. S. Athuraliya, S.H. Low, V.H. Li, Q. Yin: *REM: active queue management.* Network, IEEE, 15(3):48 -53, May 2001.
4. 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, L. Zhang: RFC 2309: *Recommendations on Queue Management and congestion avoidance in the Internet*, April 1998. Status: INFORMATIONAL.

5. X. Chang, J. K. Muppala: *A stable queue-based adaptive controller for improving AQM performance.* Comput. Netw., 50:2204-2224, September 2006.
6. W.-C. Feng, K.G. Shin, D.D. Kandlur, D. Saha: *The BLUE active queue management algorithms.* Networking, IEEE/ACM Transactions on, 10(4):513-528, August 2002.
7. M. Christiansen, K. Jeffay, D. Ott, F. D. Smith: *Tuning RED for web traffic.* In in Proceedings of ACM SIGCOMM 2000, 139-150, 2000.
8. L. Chrost, A. Chydzinski: *On the evaluation of the Active Queue Management mechanisms.* In Evolving Internet, 2009. INTERNET '09. First International Conference on, 113-118, 2009.
9. G. Di Fatta, F. Hoffmann, G. Lo Re, A. Urso: *A genetic algorithm for the design of a fuzzy controller for active queue management.* Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 33(3):313-324, 2003.
10. W.-C. Feng, D.D. Kandlur, D. Saha, K.G. Shin: *A self-configuring RED gateway.* In INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 3, 1320-1328, March 1999.
11. V. Firoiu, M. Borden: *A study of active queue management for congestion control.* In INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 3, 1435-1444, March 2000.
12. S. Floyd, R. Gummadi, S. Shenker: Adaptive RED: An Algorithm for Increasing the Robustness of RED, 2001.
13. S. Floyd: Recommendation on using the "Gentle" variant of RED algorithm, March 2000.
14. S. Floyd, V. Jacobson: *Random early detection gateways for congestion avoidance.* IEEE/ACM Trans. Netw., 1:397-413, August 1993.
15. S. Guo, X. Liao, Ch. Li, D. Yang: *Stability analysis of a novel exponential-RED model with heterogeneous delays.* Comput. Commun., 30:1058-1074, March 2007.
16. C.V. Hollot, V. Misra, D. Towsley, W. Gong: *Analysis and design of con-trollers for AQM routers supporting TCP flows.* Automatic Control, IEEE Trans-actions on, 47(6):945-959, June 2002.
17. H. Jiang, C. Dovrolis: *Passive estimation of TCP round-trip times.* ACM Computer Communication Review, 32:75-88, 2002.
18. S.S. Kunniyur, R. Srikant: *An adaptive virtual queue (AVQ) algorithm for active queue management.* Networking, IEEE/ACM Transactions on, 12(2):286-299, 2004.
19. T. O. Lakshman, T. V. Lakshman, L. Wong: *SRED: Stabilized RED.* In Proceedings of INFOCOM, 1346-1355, 1999.
20. S.H. Low, F. Paganini, J. Wang, S. Adlakha, J.C. Doyle: *Dynamics of TCP/RED and a scalable control.* In INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 1, 239-248, 2002.
21. E. Plasser, T. Ziegler: *A RED function design targeting link utilization and stable queue size behavior.* Comput. Netw., 44:383-410, February 2004.

22. F. Ren, Ch. Lin, B. Wei: *A robust active queue management algorithm in large delay networks.* Comput. Commun., 28:485-493, March 2005.

23. S. Shakkottai, R. Srikant, N. Brownlee, A. Broido, K.C. Claffy: *The RRT distribution of TCP flows in the internet and its impact on TCP-based flow control*, 2004.

24. J. Sun, M. Zukerman: *An adaptive neuron AQM for a stable internet.* In Proceedings of the 6th international IFIP-TC6 conference on Ad Hoc and sensor networks, wireless networks, next generation internet, NETWORKING'07, 844-854, Berlin, Heidelberg, 2007. Springer-Verlag.

25. L. Tan, W. Zhang, G. Peng, G. Chen: *Stability of TCP/RED systems in AQM routers.* Automatic Control, IEEE Transactions on, 51(8):1393-1398, 2006.

26. Ch. Wang, J. Liu, B. Li, K. Sohraby, Y. T. Hou: *Lred: A robust and responsive AQM algorithm using packet loss ratio measurement.* IEEE Transactions on Parallel and Distributed Systems, 18:29-43, 2007.

27. J. Wang, L. Rong, Y. Liu: *Design of a stabilizing AQM controller for large-delay networks based on internal model control.* Comput. Com-mun., 31:1911-1918, June 2008.

28. N. Xiong, Y. Pan, X. Jia, J.- H. Park, Y. Li: *Design and analysis of a self-tuning feedback controller for the internet.* Comput. Netw., 53:1784-1797, July 2009.

Algorytm AQM dla sieci z dużymi opóźnieniami

Streszczenie

W obecnym Internecie odrzucanie bądź znakowanie pakietów ma na celu powiadomienie nadawcy o przeciążeniu. Ten fakt jest wykorzystywany przez źródła TCP w celu ograniczenia prędkości nadawania.

Rozwiązaniem problemu pełnej kolejki jest prewencyjne odrzucanie pakietów, aby nie dopuścić do zapełnienia bufora i powstania przeciążenia. Prawdopodobieństwo prewencyjnego odrzucenia pakietu rośnie wraz ze wzrostem poziomu przeciążenia.

Idea ta jest wykorzystywana w aktywnych algorytmach zarządzania kolejką. Prewencyjne odrzucanie pakietów wprowadza mechanizm sprzężenia zwrotnego informując nadawców o zbliżającym się przeciążeniu. Informacja jest wykorzystywana przez nadawców w celu zwolnienia szybkości nadawania. Losowe odrzucanie wybranych pakietów pozwala uniknąć sytuacji, w której wszystkie źródła zwalniają jednocześnie, co eliminuje problem globalnej synchronizacji.

Większość obecnych algorytmów AQM pomija wpływ cech charakterystycznych dla łączy szkieletowych, tj. dużej przepustowości i dużych opóźnień propagacji. W rezultacie sprawność algorytmów AQM rozumiana jako przepustowość łącz i stabilność kolejki jest znacznie mniejsza niż w sieciach dostępowych.

W pracy zaprezentowano badania wydajności popularnych algorytmów AQM w sieciach szkieletowych o dużych przepustowościach i dużych opóźnieniach propagacji. W dalszej części przedstawiono propozycje algorytmu, który ma na celu poprawić przepustowość transmisji na tych łączach i który pozwala wykluczyć użycie generatora liczb losowych. Zaproponowana metoda – algorytm LINDROP – wykorzystuje niemalejącą funkcję liniową, w zależności od średniej długości kolejki, do oszacowania współczynnika odrzucania nadchodzących pakietów. Algorytm poprawia przepustowość w łączu szkieletowym.