Theoretical and Applied Informatics
ISSN 1896–5334
Vol.21 (2009), no. 3-4
pp. 193–204

Simulation models of fair scheduling for the TCP and UDP streams

SŁAWOMIR NOWAK[a], JOANNA DOMAŃSKA[a], ADAM DOMAŃSKI[b]

[a]Polish Academy of Sciences
Baltycka 5, 44–100 Gliwice, Poland
{emanuel,joanna}@iitis.pl

[b]Institute of Informatics
Silesian Technical University
Akademicka 16, 44–100 Gliwice, Poland
adamd@polsl.pl

Received 13 October 2009, Revised 3 November 2009, Accepted 30 November 2009

Abstract: Nowadays, a lot of Internet applications are using UDP protocol to transport the data. The congestion control mechanisms built into TCP protocol in conjunction with the Active Queue Management mechanisms, during normal operation of the Internet network, favor UDP streams. The article investigates the influence of active queue menagement and scheduling algorithm on fairness of TCP and UDP data streams.

Keywords: fairness queueing, active queue management, conquestion control

1. Introduction

The algorithms of queue management in IP routers determine which packet should be deleted when necessary. The active queue management, recommended now by IETF, enhances the efficiency of transfers and cooperate with TCP congestion window mechanism in adapting the flows intensity to the congestion in the network. Nowadays, a lot of Internet applications using UDP protocol to transport the data. For UDP traffic the examined parameters of the queue with AQM are significantly worse. The congestion control mechanisms built into TCP protocol in conjunction with the Active Queue Management mechanisms, during normal operation of the Internet network, favor UDP streams [1][2]. In this article, authors try to describe the problem of scheduling packets in the node allowing fair treatment both types of data streams (TCP and UDP).

Most AQM algorithms do not differentiate between types of packages (RED, REM, BLUE, PI). Some of them ensure the equitable distribution of resources among active streams (WRED, SFBLUE, CHOKe). However, their usefulness for the solution of the problem described above seems to be (by authors of this article) questionable. Algorithms based on stochastic or weighted fair queuing do not consider, as shown later in this article, the problem of TCP self-discrimination. The second type of algorithm based on random comparing incoming packet with packages waiting in the buffer (CHOKe) better controls the UDP streams [4]. However, the problem of identifying the package in the buffer is very complex computationally. Hence, the actual implementation of the algorithm in the router may be uneconomic [2][3]. In addition, the algorithm may not correctly work for traffic associated with the exchange of P2P data or DDoS attacks (a large number of small transmissions). For these reasons, we proposed the simple solution, based only on two queues, the first for the TCP stream, the other for UDP.

In this article we present simulation results. The simulation evaluations were carried out with the use of OMNeT++ (in version 4.0) simulation framework extended with the INET package. We add some improvement to the INET implementation: PRIO and SFQ scheduling (with special modifications), new sets of parameters and some new statistics, distribution and traffic scenarios.

Section 2 gives basic notions on the transport layer congestion control and active queue management. Section 3 shortly presents simulation model. Section 4 discusses numerical results. Some conclusions are given in section 5.

2. The transport layer conguestion control and active queue menagement

The Internet applications can generate streams of network packets with different traffic profiles. Generally we can distinguish two types of traffic: stream traffic and elastic traffic [6]. Applications VoIP or VoD generate the stream network traffic. They often use the UDP protocol to transport data. Research shows that UDP packets traffic has recently been significant and the number of applications that use UDP growing. In contrast to the traffic stream, source adjusting the speed of sending data according to load the network, generate the elastic traffic. An example of such sources are applications that use the TCP protocol. There are many implementations of TCP. The most important modification, in relation to the original version [35] was the introduction of a mechanism to prevent overloading the network (congestion avoidance) [36]. Currently there are two types of congestion management in the network by TCP. The first type is based on packet loss during transmission. The most popular protocols in this family are TCP newRENO and TCP Sack [5]. Basically, for such protocols sources reduce the transmission speed for packet loss. For the second type of protocol (TCP VEGAS) packet generation rate depends on the time delays in transmission. In Fig. 1 you can see a significant reduction of transmission speed for packet loss for TCP Reno mechanism. For TCP Vegas (Fig. 2) trasmission speed is set at middle level that causes no loss in the queues.

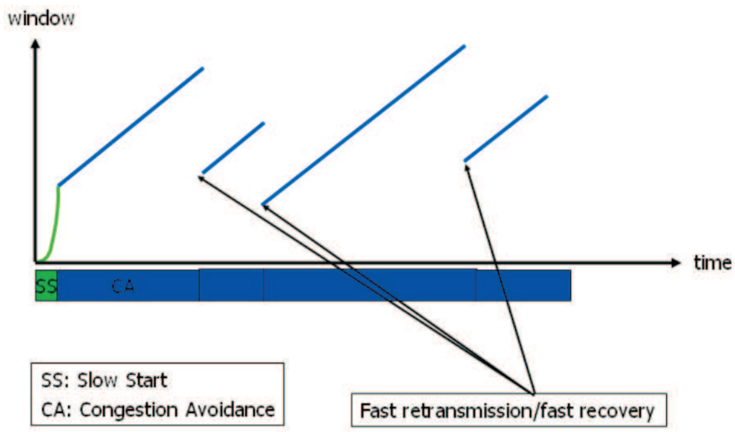

Fig. 1. Resizing the window transmission for TCP Reno protocol [37]

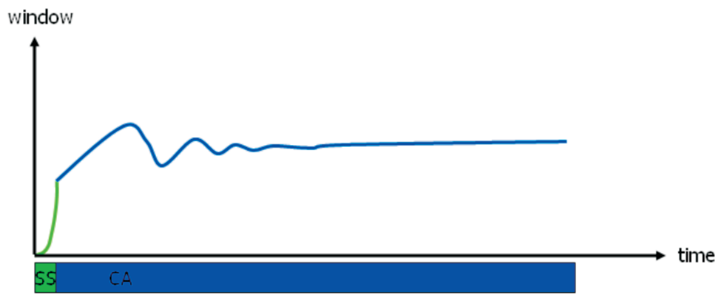

Fig. 2. Resizing the window transmission for TCP Vegas protocol [37]

In *passive* queue management, packets coming to a buffer are rejected only if there is no space in the buffer to store them and the senders have no earlier warning on the danger of growing congestion. In this case all packets coming during saturation of the buffer are lost. To enhance the throughput and fairness of the link sharing, also to eliminate the synchronisation, the Internet Engineering Task Force (IETF) recommends *active* algorithms of buffer management. They incorporate mechanisms of preventive packet dropping when there is still place to store some packets, to advertise that the queue is growing and the danger of congestion is ahead. The probability of packet rejection is growing together with the level of congestion. The packets are dropped randomly, hence only chosen users are notified and the global synchronisation of connections is avoided [17]. As you can see, the rejection of the package from the queue lowers the transmission speed. Unfortunately, this situation occurs only for the TCP transmissions. Discussed above mechanism must lead to unequal treatment of TCP and UDP streams during the competition for a common link.

3. Simulation models

The simulation evaluations were carried out with the use of OMNeT++ (in version 4.0) simulation framework extended with the INET package. The OMNeT++ is the modular, component-based simulator, with an Eclipse-based IDE and a graphical environment, mainly designed for simulation of communication networks, queuing networks and performance evaluation. The framework is very popular in research and for academic purposes [32], [33]. The INET Framework is the communication networks simulation extension for the OMNeT++ simulation environment and contains models for several Internet protocols: UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, IEEE 802.11, MPLS, OSPF, etc. [34].

The simulated network was based on the example provided with the INET package to evaluate the behavior of the queues in a simple network with different traffic scenario. The INET built in queue algorithms are drop tail queue and RED tail drop algorithm. We add some improvement to the INET implementation eg. SFQ and PRIO scheduling, some new statistics, distribution and traffic scenarios. The network's topology is presented on Fig. 3.

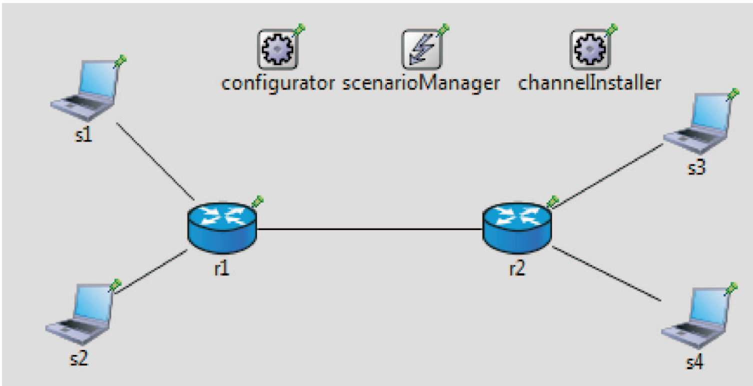

Fig. 3. The simulation network's topology.

The link between r1 and r2 routers was the bootleneck of the network (changed as 100kbps, 1Mbps, 10Mbps, 100 Mbps). The rest of links was 100Mbps. The evaluated queue was the output queue for the r1 router. For the purposes of the research, we proposed the following modification of the router. Buffer in the router consisted of two queues, one for the TCP stream and one for UDP stream. The simulation were perfomed for different algorithms of the packet scheduling (PRIO, RR – rounf and robin , weighted

	QUEUE1		QUEUE2	
Cases:	type	Mean size (std dev)	type	Mean size (std dev)
One queue	3DropTail	36.7 (12.3)	–	–
One queue	RED (TailDrop)	32 (7.9)	–	–
Priority TCP	RED (TailDrop)	11 (3.2)	FIFO	960 (570)
Priority TCP	FIFO	69 (30)	FIFO	986 (555)
SFQ (4UDP/ 1TCP)	FIFO	111 (38)	FIFO	0.8 (0.68)
SFQ (1UDP/ 1TCP)	FIFO	90.9 (34)	FIFO	960 (563)
SFQ (1 UDP/ 1TCP)	RED (TailDrop)	6.9 (2.0)	FIFO	959 (562)
Priority UDP	RED (TailDrop)	7.7 (2.3)	FIFO	0.80 (0.68)

	TCP 1		TCP 2		TCP 3	
Cases:	RTT	Smoothed RTT	RTT	Smoothed RTT	RTT	Smoothed RTT type
One queue	0.23	0.24	0.24	0.25	0.22	0.2
One queue	0.24	0.22	0.24	0.23	0.23	0.24
Priority TCP	0.12	0.09	0.12	0.11	0.11	0.12
Priority TCP	0.47	0.22	0.27	0.17	0.40	0.34
SFQ (4UDP/ 1TCP)	1.3	0.7	2.4	1.7	2.4	1.7
SFQ (1UDP/ 1TCP)	0.95	0.29	0.94	0.69	0.47	0.59
SFQ (1 UDP/ 1TCP)	0.14	0.13	0.13	0.12	0.16	0.15
Priority UDP	0.29	0.27	0.26	0.25	0.28	0.28

Table 1. Round-trip delay time

RR) and different algorithms of queue menagement (RED, FIFO). The simulation were perfomed for TCP connections only and for a TCP which operates together with a UDP. The connection between hosts s1 and s3 was a TCP connection (TCP Reno). The UDP connection between hosts s2 and s4 corresponded to a simple video frames transmission. The general queue parameters were: DropQueue size = 25, RED (both cases): wq = 0.02, minth = 15, maxth = 25, maxp = 0.03. The simulated time was 200[s].

4. Numerical results

In this section we present more interesting results achieved in the simulation.

One queue with the RED for TCP and UDP streams.
During the first experiment the both data streams (TCP and UDP) are placed in a one RED queue.

Distribution of the moving average queue length is presented in Fig. 4. As you can see queue was completely unstable. TCP parameters (RTT:0.24 (see Table 1), great number of unacknowledged segments, long data transmission time) indicate starvation TCP over UDP.

Additionally observing TCP traffic parameters (average: 0.24 RTT, large number of unacknowledged segments, long data transmission time) we can indicate that UDP stream fully appropriated link.

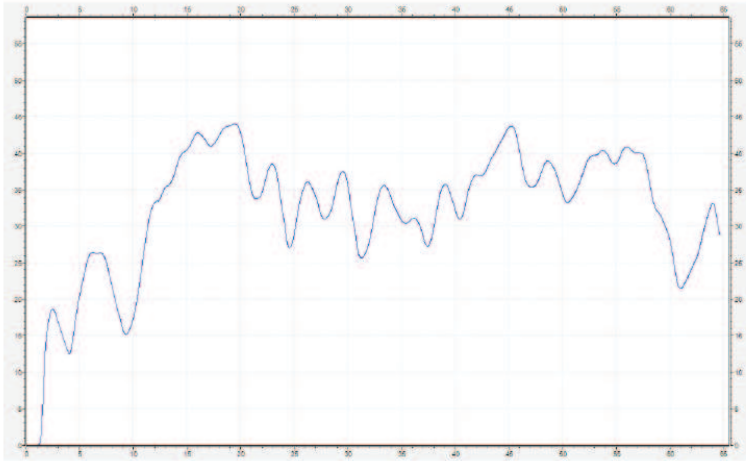

Fig. 4. Moving average queue length as a function of time – RED queue

Two queues for TCP and UDP streams with PRIO scheduling. In the next phase of the experiment, we tried to increase the chances of the TCP stream. In the router r1 we created a double queue with PRIO scheduling. The first queue for the TCP segments and second for the UDP datagrams. UDP packets were sent when the first queue is empty. The queue for the TCP has implemented the RED mechanism. The UDP queue was FIFO. The results of the experiment are shown in Fig. 5.

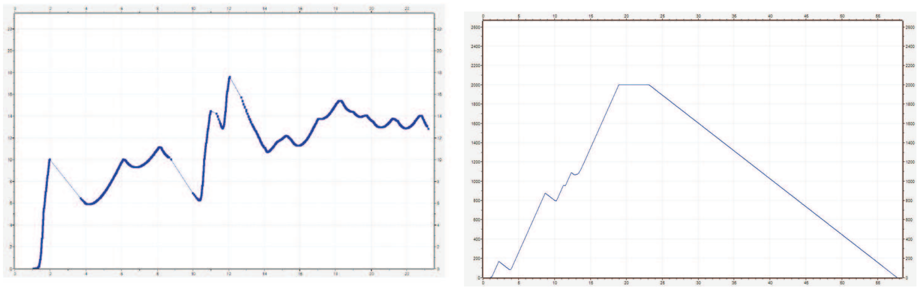

Fig. 5. Real queue length as a function of time – TCP RED queue (left), UDP FIFO queue (right) PRIO scheduling

This case is very comfortable for TCP. RTT, depending on the particular transmission ranges from 0.09 to 0.12. However, looking at the udp queue, we see that the UDP broadcast is being held in queue until the completion of the TCP transmission. Therefore, in the next simulation we modified the algorithm PRIO. UDP packets were sent when the first queue was empty or when the RED alghoritm dropped packet from the first queue. Obtained results was practically the same. The reason for this situation was too small number of packets dropped by the RED algorithm.

Fig. 6 shows a totally opposite situation. The UDP traffic was redirected to the first queue. Contrary to expectations, this situation has proved to be the best. UDP broadcast video transmission was sent without problems. TCP transmission speed is very satisfactory. The obtained results were caused by very specific distribution of packets in a stream of UDP.

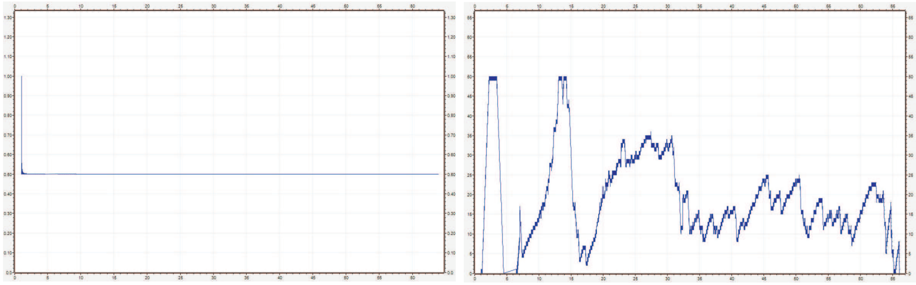

Fig. 6: Real queue length as a function of time – UDP RED queue (left), TCP RED queue (right) PRIO with modification scheduling

Two queues for TCP and UDP streams with RR – Round And Robin scheduling.
It is clear that the PRIO algorithm is suitable only in very specific network conditions. Therefore, in subsequent simulations, we used to receive packets from the queues algorithm SFQ (Stochastic Fairness Queueing) – one packet from the first queue, one packet from the second. This case also is not favorable, because the UDP need more bandwidth for trouble-free transmission (about three-quarters), and this case gived him only half 7. This simulation showed that we can exactly predict what the band was needed for smooth motion video broadcast.

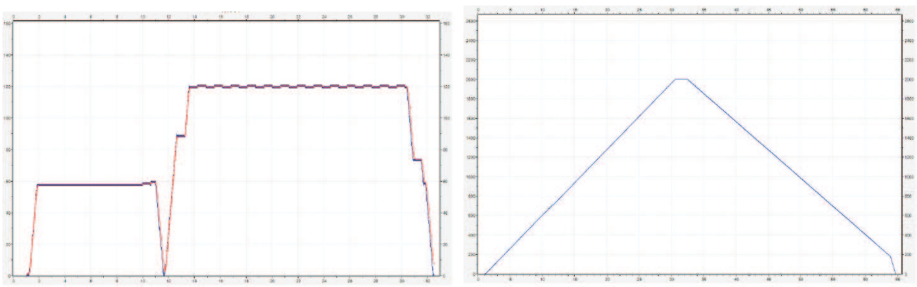

Fig. 7. Real queue length as a function of time – TCP FIFO queue (left), UDP FIFO queue (right) RR scheduling

For this reason, the last two simulations used the weighted Round And Robin schedule. Dequeueing mechanism fetched one packet from the TCP queue and four from the UDP queue. Results for both the FIFO queues are shown in Fig. 8. Situation for the RED queue for the TCP flow is shown in Fig. 9.

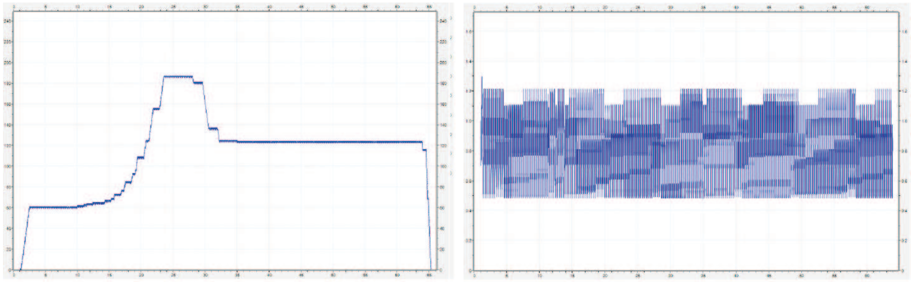

Fig. 8: Real queue length as a function of time – TCP FIFO queue (left), UDP FIFO queue (right) WRR scheduling 4*1 for UDP

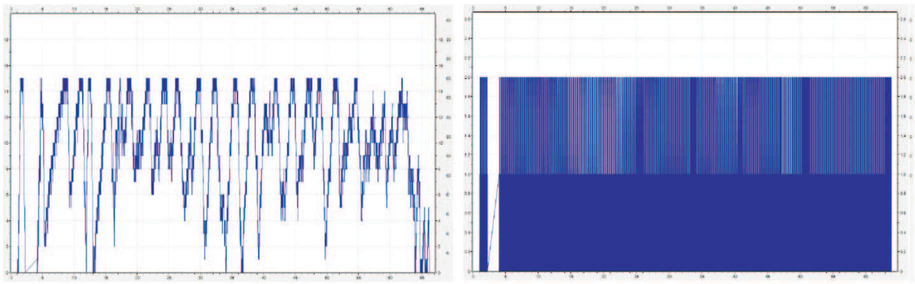

Fig. 9: Real queue length as a function of time – TCP RED queue (left), UDP FIFO queue (right) WRR scheduling 4*1 for UDP

Both cases were very good for the UDP stream. The RED mechanism for TCP queue guarantee the preservation of the optimum value of RTT (RTT average of 0.25 to 0.29 [s]).

5. Conclusions

In this article we present the problem of unfair distribution of links between the TCP and UDP streams. This problem is growing, where transmission takes place through the nodes, with AQM mechanisms. During the tests we tried to answer the question whether it is possible to create such conditions in the queue for both types of data streams to be treated fairly. Our research was carried out in the environment of new Omnet++ (in version 4.0) simulation framework extended with the INET package.

During the tests we analyzed the following parameters of the transmission with AQM: the length of the queue, the number of rejected packets, transmission time and the RTT parameter. Classical RED algorithms are suitable only for TCP protocols with congestion algorithms. For UDP traffic the examined parameters of the transmission with AQM are significantly worse. Moreover, UDP transfers may in this case suppress the TCP transmissions. Performed studies have shown also that it is possible, using well-known queuing mechanisms, to ensure fair treatment for both TCP and UDP streams. The best solution would be here an automatic selection of the parameters of queuing algorithms, depending on the statistical data collected during normal operation of the router. Our research has shown that this is possible but very difficult.

References

1. S. Nowak, J. Domańska, A. Domański: “Performance modeling of selected AQM mechanisms in TCP/IP network” IWSI 2009.
2. J. Domańska, A. Domański, T. Czachórski: “Implementation of modified AQM mechanisms in IP routers”, Journal of Communications Software and Systems, Vol. 4, 1, March 2008.
3. J. Domańska, A. Domański: “Active Queue Management in Linux based routers” IWSI 2008.
4. A. Brachman: “Wplyw mechanizmow kontroli ruchu na jakosc uslug w sieciach bezprzewodowych”, Phd Thessis, Gliwice 2008.
5. H. Lee, Soo-hyeong Lee, Yanghee Choi: The Influence of the Large Bandwidth-Delay Product on TCP Reno, NewReno, and SACK, Japonia, 2001.
6. W. Burakowski, H. Tarasiuk, P. Krawiec: “Analiza algorytmow i mechanizmow sterowania ruchem na poziomie pakietow w sieci IP QoS”, Polska, 2008.
7. J. Domańska, A. Domański, T. Czachórski: “The Drop-From-Front Strategy in AQM”, Lecture Notes in Computer Science, Vol. 4712/2007, pp. 61-72, Springer Berlin/ Heidelberg, 2007.
8. A. Kapadia, W. Feng, R. H. Campbell: GREEN: A TCP Equation Based Approach to Active Queue Management; http://www.cs.dartmouth.edu/ akapadia/ papers/UIUCDCS-R-2004-2408.pdf
9. S. Athuraliya, V. H. Li, S. H. Low and Q. Yin: REM: Active Queue Management; http://netlab.caltech.edu/FAST/papers/cbef.pdf
10. S. S. Kunniyur: Member, IEEE, and R. Srikant, Senior Member, IEEE An Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management; http://comm.csl.uiuc.edu/srikant/Papers/avq.pdf
11. E. Hashem: Analysis of random drop for gateway congestion control; http://www.worldcatlibraries.org/oclc/61689324

12. W. Feng, D. Kandlur, D. Saha, K. Shin: Blue: A New Class of Active Queue Management Algorithms; http://citeseer.ist.psu.edu/feng99blue.html
13. W. H. Feng, D.D. Kandlur, D. Saha, K.G. Shin: A Self-Configuring RED Gateway; http://citeseer.ist.psu.edu/470052.html
14. http://en.wikipedia.org/
15. S. Floyd, V. Jacobson: Random Early Detection gateways for Congestion Avoidance; http://www.cs.ucsd.edu/classes/wi01/cse222/papers/floyd-red-ton93.pdf
16. Random Early Detection (RED): Algorithm, Modeling and Parameters Configuration; http://photon.poly.edu/ jefftao/JTao_RED_report.pdf
17. S. Floyd, R. Gummadi, and S. Shenker Adaptive RED: An Algorithm for Increasing the Robustness of REDs Active Queue Management; http://citeseer.ist. psu.edu/448749.html
18. D. Lin, R. Morris: Dynamics of Random Early Detection; https://pdos.csail.mit. edu/r̃tm/papers/fred.pdf
19. RFC 793 – Transmission Control Protocol, http://www.faqs.org/rfcs/rfc793.html
20. http://en.wikipedia.org/wiki/TCP_congestion_avoidance_algorithm
21. T. Alemu, A. Jean-Marie: Dynamic Configuration of RED Parameters; http:// citeseer.ist.psu.edu/728472.html
22. R.Verma, A. Iyer, A. Karandikar: Towards an adaptive RED algorithm for archiving daleloss performance http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1214606
23. X. Yang, H. Chen, S. Lang: Estimation Method of Maximum Discard Probability in RED Parameters http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1712588
24. J. Hong, C. Joo, S. Bahk: Active queue management algorithm considering queue and load states http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/9617/30391/01401608.pdf
25. S. Floyd: Discussions of setting parameters, http://www.icir.org/floyd/REDparameters.txt, 1997.
26. B. Zheng, M. Atiquzzaman: A framework to determine the optimal weight parameter of red in next generation internet routers, The University of Dayton, Department of Electrical and Computer Engineering, Tech. Rep., 2000.
27. M. May, T. Bonald, J. Bolot: Analytic evaluation of red performance, IEEE Infocom 2000, Tel-Aviv, Izrael, 2000.
28. W. Chang Feng, D. Kandlur, D. Saha: Adaptive packet marking for maintaining end to end throughput in a differentiated service internet, IEEE/ACM Transactions on Networking, vol. 7, no. 5, pp. 685-697, 1999.
29. M. May, C. Diot, B. Lyles, and J. Bolot: Influence of active queue management parameters on aggregate traffic performance, Research Report, Institut de Recherche en Informatique et en Automatique, Tech. Rep., 2000.

30. M. Hassan, R. Jain: High Performance TCP/IP Networking. Pearson Education Inc., 2004.
31. C. Liu, R. Jain: Improving explicit congestion notification with the mark-front strategy. Computer Networks, Amsterdam Netherlands: 1999 2001.
32. OMNET++ homepage, http://www.omnetpp.org/.
33. J. Domańska, K. Grochla, S. Nowak: Symulator zdarzeń dyskretnych OMNeT++, Wyd. Wyzsza Szkola Biznesu w Dabrowie Górniczej, Dabrowa Górnicza 2009.
34. INET homepage, http://inet.omnetpp.org/.
35. J. Postel (ed.), et al.: "Transmission Control Protoco", Information Sciences Institute, University of Southern California, Internet RFC 0793, September 1981.
36. W.R. Stevens: "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithm", Internet RFC 2001, January 1997.
37. S. Low: "Equilibrium and Dynamics of TCP/AQM", netlab.caltech.edu, March 2002.

Model symulacyjny równomiernego kolejkowania strumieni TCP I UDP

Streszczenie

We współczesnych aplikacjach, działających w sieci Internet transmisje realizuje się korzystając najczęściej z protokołu UDP. Mechanizm kontroli przeciążeń, wbudowany w TCP, współpracujący z mechanizmami AQM, działającymi w kolejkach powoduje, że znaczący udział ruchu UDP jest w stanie zawłaszczyć pasmo transmisji. W artykule badano wpływ różnych wariantów kolejkowania na ruch TCP i UDP, aby zapewnić możliwie najlepszy (najbardziej sprawiedliwy) podział pasma. Badania prowadzono z wykorzystaniem pakietu symulacyjnego OMNeT++ wraz z pakietem INET (do symulacji protokołu TCP/IP). Do funkcjonalności tych narzędzi dodano implementacje dla kolejek PRIO oraz SFQ (osobne kolejki dla ruchu TCP i UDP, obsługiwanych zgodnie z regulaminem FIFO oraz RED). Badania przeprowadzano dla różnych konfiguracji usług korzystających z UDP i TCP, współdzielących łącze, stanowiące wąskie gardło pomiędzy podsieciami (Rys. 3). W symulacjach badano parametry kolejek (długość, liczba odrzuconych pakietów) oraz parametr RTT transmisji TCP. Dobranie właściwego sposobu kolejkowania jest zagadnieniem złożonym. Dla typowego mechanizmu RED obserwujemy zawłaszczanie łącza przez usługę UDP. Dla kolejek priorytetowych TCP oraz mechanizmu SFQ sytuacja jest odwrotna. Transmisja TCP o odpowiednio dużym natężeniu staje się dominująca, co bardzo niekorzystnie wpływa na UDP, szczególnie jeśli jest związany z usługami o określonych wymaganiach QoS (typu np. wideo w czasie rzeczywistym). Określone, drobne modyfikacje regulaminów także nie dają

zauważalnej poprawy. Dla określonego przypadku można dobrać możliwie najlepszy mechanizm kolejkowania z odpowiednimi parametrami (w badanym przypadku było to PRIO dla kolejki UDP), w innej konfiguracji ruchu parametry i regulaminy kolejek dla najlepszego przypadku będą już inne. Trudno jest więc na tym etapie zaproponować rozwiązanie uniwersalne. Zaproponowanie takiego mechanizmu jest motywacją dla przyszłych prac w tej dziedzinie.