USE OF MODIFIED ADAPTIVE HEURISTIC CRITIC ALGORITHM FOR NOVEL SCHEDULING MECHANISM IN PACKET-SWITCHED NETWORKS

Mariusz Jednoralski[1], Tomasz Kacprzak[2]

Abstract. In this paper a novel scheduling algorithm of packet selection in a switch node for transmission in a network channel, based on Reinforcement Learning and modified Adaptive Heuristic Critic is introduced. A comparison of two well known scheduling algorithms: Earliest Deadline First and Round Robin shows that these algorithms perform well in some cases, but they cannot adapt their behavior to traffic changes. Simulation studies show that novel scheduling algorithm outperforms Round Robin and Earliest Deadline First by adapting to changing of network conditions.

Keywords: Reinforcement Learning, telecommunication networks, packets scheduling

1. Introduction

An increase of traffic intensity and its time-varying nature of bursty shape in a telecommunication networks, such as Internet, ATM, and Next Generation Networks requires application of new and advanced methods of information flow control, in order to avoid degradation in the network performance due to for example saturation of its resources (communication links, buffers, network switches, etc.). Importance of this problem drastically increases when two or more information flows with significantly difference in their priorities must be multiplexed and contend to a common transmission output while using the full throughput of the switch and avoiding starvation of any data stream. A possible solution of this problem is a proper selection of packets and generation a decision of their sending during a given time slots. One of the approach is well-known packet scheduling which is dynamic process that selects packets to be transmitted during the next time-slot based on for example a so-called Round Robin (RR) and Earliest Deadline First (EDF) techniques. However, most of the scheduling techniques cannot adapt to dynamically changing network conditions, like packet bursts. In order to make this adaptivity possible a different methods of computer-based decision algorithms can be applied, amongst them artificial neural networks, fuzzy logic, genetic calculations or any other method of the computational intelligence discipline.

[1] PhD student, Doctoral Study at the Faculty of Electrical and Electronic Engineering, Technical University of Łódź, Stefanowskiego Str. 18/22, 90-924 Łódź, Poland.
[2] Institute of Electronics and Telecommunication, Technical University of Lodz, Wólczańska Str. 223, 90-924 Łódź, Poland

In this paper, we propose a novel scheduler technique based on Reinforcement Learning, which provides appropriate levels of service according to the internet traffic priority. Reinforcement Learning refers to a class of learning tasks in which the system learns desired behaviour while maximizing a scalar evaluation function taking into account signals generated back from the environment. Application areas where reinforcement learning ideas are often implemented are related to such a problems like intelligent control, game playing, scheduling, optimization and network routing [1].

The new scheduling algorithm derived from modified Reinforcement Learning technique is based on Adaptive Heuristic Critic [1]. A Reinforcement Learning algorithm is used to optimise scheduling criteria of IP packet delay. We also compare the performance of two well-known scheduling algorithms: Round Robin and Earliest Deadline First with our novel algorithm in a simulation of internet traffic. Delay is a very important issue especially for real time internet traffic. VoIP packets delayed over 150ms can cause degradation of speech quality, or even be dropped by receiving application [2]. Through the simulations of real-time application traffic we show that the Reinforcement Learning based mechanism achieves a decrease in the number of delayed packets for higher priority IP traffic which results in improved level of quality of service in the network.

2. Convergent IP networks

Today market is focused on convergent solutions that integrate telecommunications and computer networks based on IP protocol. The need for convergence results from necessity for integration a number of different applications, that are base for today society and business. Particularly important matter in convergence services is ability to use a real time applications [3].

A number of applications such as IP telephony, videoconferencing, video streaming and internet browsing rely on traffic scheduling algorithms in network nodes to guarantee performance bounds and meet the Quality-of-Service (QoS) requirements [4,5]. The function of the scheduler is to determine the exact sequence in which these packets will be transmitted. The various data services require different levels of service: a few seconds delay delivering an e-mail is unimportant, but 300 m seconds delay makes conversation almost unsustainable [2].

2.1. Packet network congestions

In spite of growing networks throughput, congestions are still very important matter. Congestion can be caused by vast amount of data that are exchanged by modern applications, DoS attacks, viruses and internet worms. It can be divided into several classes based on congestion

duration: from short time to long time [6]. Depending on time scale of congestion different prevention methods can be used [7].

Congestion can be measured based on parameters of incoming packets and buffer queue lengths [8]. Congestions can cause degradation of QoS in the network. Forceful network management requires steering congestion counteraction. Main results of congestions are:

- session disconnection (VoIP)
- throughput reduction
- transmission of overdue packets.

Congestions are present in most of telecommunication and data networks, of which short time lasting from tens of μs up to several seconds are the most probable [9]. Depending on congestion duration different prevention mechanisms can be used. For congestions lasting from μs to ms a proper scheduling mechanisms can be used [7].

2.2. Packet delay

Different classes of service have different QoS parameters like delay and packet loss. The impact of congestions, in terms of packet delay and loss, is different for real-time traffic than for data traffic. For real time transmissions like voice, packets delay over 150 ms causes conversation incomprehensive [2,10]. Streaming of video and audio needs delay bounds about 300 ms. In order to assure transmission parameters for real time applications like mentioned above, applications are classified to different priority queues. Real time applications are scheduled before other types of traffic. The highest priority is given for conversational lass while lower priority for streaming class. In further part of the article we deal with real-time traffic priority classes.

It is very important to provide mechanisms for assuring Quality of Service for real time transmissions. Videoconferencing and IP telephony are transported in IP networks using UDP protocol, that is unreliable and connectionless. Other types of traffic like e-mail, data are transported using TCP protocol based on connections and reliable. Another feature that differentiate TCP from UDP is a mechanism that counteracts congestion [10].

2.3. Scheduling mechanisms

Scheduling disciplines determine the packets forwarding order. They affect packet delay, congestions and queue lengths [11]. The aim of scheduling is to decide which packet to send first in order to guarantee the quality of service parameters, for example delay. We compared the efficiency of two well known scheduling algorithms Round Robin and Earliest Deadline First.

In Round Robin mechanism incoming packets are placed in queues according to their source; packets are scheduled one at a time from each queue in a round robin fashion. This scheduler is fair in the sense that it

gives an equal chance of service to all data flows.

Using Earliest Deadline First for any scheduling decision time, a packet with the earliest deadline is scheduled. Deadline is calculated as an upper bound on the tolerable end-to-end delay, and thus, measures the usefulness of data packets at the destination.

3. Reinforcement Learning

Reinforcement Learning is based on an agent that learns the behaviour through trial-and-error interactions with the environment [1, 12]. Two main strategies are known for solving this problem, the first one is searching in the space of behaviours in order to find strategy that performs well in the environment. The second one is using statistical techniques and dynamic programming methods to estimate the utility of actions taken in the environment.

3.1. Environment model

Reinforcement learning is a learning of strategy of behaviour that maximizes a numerical reward signal. The agent is not told which actions to take but is discovering which actions yields the highest reward by trying them.

Figure 3.1. Reinforcement Learning environment model

3.2. Reward function

At each time step t the agent receives as an input some representation of the current state s_t of the environment. The agent chooses an action a_t that is an output signal to the environment. The action changes the state and as a consequence of its action, the agent receives a scalar reinforcement signal φ_t

$$\varphi : S \times A \rightarrow R \quad (1)$$

that is a reward (1).

The main purpose of using Reinforcement Learning is to maximize received rewards by an actor from environment. Defining rewards that are received after N discrete time steps starting from time n:

$$\varphi_{n+1}, \varphi_{n+2}, \varphi_{n+3}, \ldots, \varphi_N$$

$$R_N = \varphi_{n+1} + \varphi_{n+2} + \varphi_{n+3} + \ldots \varphi_N \quad (2)$$

$$R_N = \sum_{k=1}^{N} \varphi_{n+k} \quad (3)$$

in general, the agent tries to maximize the expected return

$$R_n = \varphi_{n+1} + \gamma\,\varphi_{n+2} + \gamma^2\varphi_{n+3} + \ldots = \sum_{k=0}^{\infty}\gamma^k\varphi_{n+k+1} \quad (4)$$

selecting actions so that the sum of the discounted rewards it receives over the future is maximized (4). The discount rate γ determines the present value of the future rewards: a reward received t time steps into the future is worth γ^{t-1} times what it would be worth if it were received immediately. If $\gamma=0$, the agent is only concerned with maximizing the immediate rewards.

Problems with delayed reinforcement are well modelled as Markov decision processes, because decisions and values are assumed to be functions of the current state only.

3.3. Value function

Reinforcement learning algorithms are based on estimating value functions that are functions of states. They estimate how good it is for an agent to be in a particular state, defined in terms of future rewards. Value function (5) is defined with respect to policy π.

$$V^{\pi}(s) = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k\varphi_{t+k+1}\middle| s_t = s\right\} \quad (5)$$

where k represents the number of visits in a given state.

A policy π, that is a transition function, is a mapping from states $s \in S$ and actions $a \in A(s)$, to the probability $\pi(s,a)$ of taking action a when in state s. Given policy π is defined to be better than or equal to another policy π' if its expected return is greater than or equal to that of π' for all states $s \in S$ and is called an optimal one. The optimal value of state $V^*(s)$ is the expected infinite discounted sum of rewards that the agent will gain if it starts in that state and executes the optimal policy (4).

$$V^*(s) = \max_{\pi} E\left(\sum_{t=0}^{\infty}\gamma^t\varphi_t\right) \quad (6)$$

Reinforcement learning is primarily concerned with obtaining the optimal policy when no model of the environment is known in advance. The agent interacts with the environment obtaining information that can be processed to produce the optimal policy.

3.4. Temporal Difference Algorithm

Temporal Difference (TD) algorithms are learning methods for estimating value functions and discovering optimal policies [13, 14]. These methods are able to learn directly from experience without a model of the environment's dynamics and to update estimates based in part on other, previously learned estimates, without waiting for the final outcome. TD methods need wait only until the next time step. At time $t+1$ they immediately form a target and make a useful update using the observed

reward φ_{t+1} and the estimate $V(s_{t+1})$. The simplest TD method is known as TD(0)

$$V(s_t) \leftarrow V(s_t) + \alpha[\varphi_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \quad (7)$$

where s_t is the agent's state before the transition, α is the step size, φ_{t+1} is the instantaneous reward received, and s_{t+1} is the resulting state. Finally, the target for the TD update is

$$\varphi_{t+1} + \gamma V_t(s_{t+1}) \quad (8)$$

3.5. Adaptive Heuristic Critic

This algorithm is an adaptive version of policy iteration in which the value-function is computed by algorithm TD(0). A block diagram for this approach is given in Fig. 2.

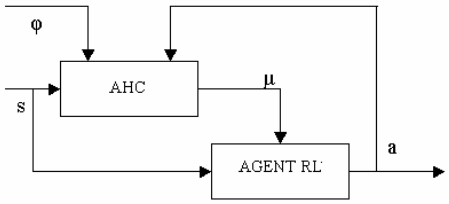

Figure 3.2. Adaptive Heuristic Critic architecture model

Learning the value of a policy is done using TD(0) algorithm update rule:

$$V(s_t) := V(s_t) + \alpha[\varphi_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \quad (9)$$

AHC algorithm uses two components: evaluation V and policy μ functions. The evaluation function computes evolution of the environment states. State evolution taking high values reflects the usefulness of that state as long as it is expected to be good. It also takes low values for states, which are expected to be bad with respect to the performance measure (4). The policy function μ is defined for each state-action pair. The merit of an action taken in state s_t is increased, changing its value for the future.

AHC Algorithm:

a) observe current state s_t
b) select an action a_t for the state s_t
c) perform action a_t, observe reinforcement φ_t and state s_{t+1}
d) estimate:
   $\Delta = \varphi_t + \gamma V_t(s_{t+1}) - V_t(s_t)$
e) $V_{t+1}(s_t) = V_t(s_t) + \alpha \bullet \Delta$
f) $\mu_{t+1}(s_t, a_t) = \mu_t(s_t, a_t) + \alpha \bullet \Delta$

The most interesting step of the algorithm is d), where the idea of secondary reinforcement is implemented. Value $\beta V_t(x_{t+1})$ is a secondary reinforcement, and φ_t is an immediate reinforcement.

4. Novel scheduling mechanism

Our novel packet scheduling mechanism is based on the methods outlined above. Its main idea is to protect one chosen class of traffic from experiencing an excessive packet delay. It is realized using an adaptive mechanism that chooses between two algorithms, RR and EDF, to realize the above mentioned goal.

Assume that there are two classes of real time traffic scheduled on one hierarchy level. Packets of each class are placed in different queues. In queue 1 there are packets with lower delay constraint (low priority class), packets with harder delay constraint (high priority class) occupy queue 2. With high priority class queue two tables with μ values are associated. One table stores μ values of EDF other for RR scheduling actions. One μ value in table is used for 10 Byte blocks of data in high priority queue.

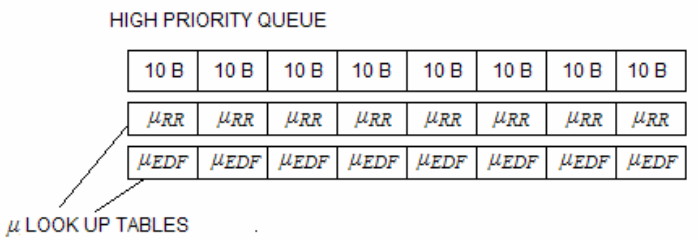

Figure 4.1. Architecture of μ values tables

The states are represented here as high priority queue lengths. In every state an algorithm can choose one of two actions using RR or EDF scheduling mechanism. The purpose of the novel adaptive algorithm is to use one of two actions to satisfy delay constraints for the higher priority class for a given queue length.

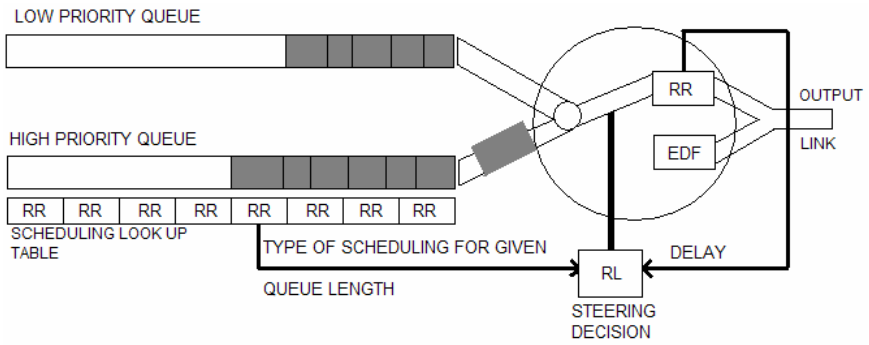

Figure 4.2. RL scheduling architecture model in case when RR is used

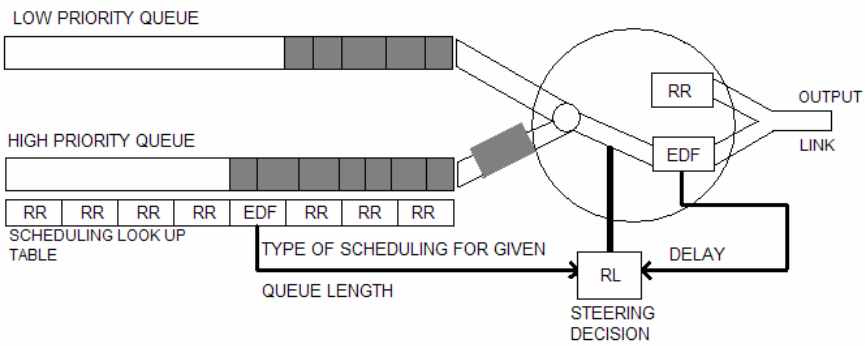

Figure 4.3. RL scheduling architecture model in case when EDF is used

The novel mechanism called RL starts using RR algorithm. Every time the algorithm schedules packets from low or high priority queue it checks μ values of EDF and RR that are connected with given high priority queue length. In a given state a mechanism with the higher policy function value is chosen for scheduling. Every time a high priority packet is scheduled its delay is checked. If the assumed maximum delay T is not exceeded a positive reward is given to the agent, by the means of positive value φ. In case of delay excess the algorithm is punished by obtaining a negative value of reward from the environment. Achieving positive reward increase μ value that is connected with algorithm that already was used for scheduling and observed queue length of high priority class. Negative reward decrease μ value. For every state the algorithm compares two policy functions. The first one, μ_{tEDF}, is connected with EDF mechanism, the second one, μ_{tRR}, with RR. Each time the following sequence of actions is taken:

If EDF was chosen

$$\{\Delta = \varphi_{t+1EDF} + \gamma_{EDF} V(s_{t+1}) - V(s_t) \quad (10)$$

$$\mu_{t+1EDF}(s_t, a_t) = \mu_{tEDF}(s_t, a_t) + \alpha_{EDF} \bullet \Delta \quad (11)$$

$$V_{t+1}(x_t) = V_t(x_t) + \alpha_{EDF} \bullet \Delta\} \quad (12)$$

if RR was chosen

$$\{\Delta = \varphi_{t+1RR} + \gamma_{RR} V(s_{t+1}) - V(s_t) \quad (13)$$

$$\mu_{t+1RR}(s_t, a_t) = \mu_{tRR}(s_t, a_t) + \alpha_{RR} \bullet \Delta \quad (14)$$

$$V_{t+1}(x_t) = V_t(x_t) + \alpha_{RR} \bullet \Delta\} \quad (15)$$

where γ_{EDF} and γ_{RR} are discount factors, α_{EDF} and α_{RR} are step sizes. In the standard AHC algorithm there is one discount factor, in our case there are two: one connected with EDF and one connected with RR thus $\gamma_{EDF} \neq \gamma_{RR}$.

5. Scheduling algorithms comparison

The novel scheduling algorithm is tested through the comparison with two standard mechanisms: EDF and RR. The tests were conducted using a simulator written in C++.

5.1. Test architecture model

For testing purpose it is assumed a traffic from 60 clients is arriving at a node. One half of clients generate low priority traffic, the other half generate high priority traffic. Throughput generated for each class by each client equals 64 kb/s. Packets are inserted into queues after prior classification. There are three scheduling algorithms implemented in the simulator. Each scheduling algorithm receives high priority traffic from 10 clients and low priority traffic from 10 clients.

Packet arrival in time window is generated according to Poisson distribution and their length is given by normal distribution. In order to check behavior of the schedulers packet bursts are generated periodically. A burst is an additional traffic that appears in the classes. The maximum delay has value $T = 10$ *ms.*

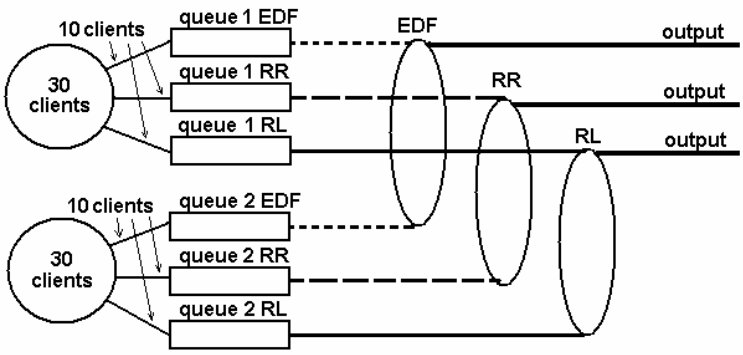

Figure 5.1. Test architecture model

5.2. Scenario 1

In the first scenario the low level traffic class generates burst. The volume of traffic burst is 2.5 times higher than mean throughput of the low level class. The burst period is 50 ms and its duration changes from 2.5 ms to 20 ms. As shown in Figure 5.2. and Figure 5.3. applying EDF scheduler for the given traffic makes the burst affect packets in queue 2 in addition to packets in queue 1. The high traffic parameters deteriorate since some packets are delayed more than 10 ms. It is important that RL algorithm approximates RR algorithm that protects the higher level (in queue 2) traffic.

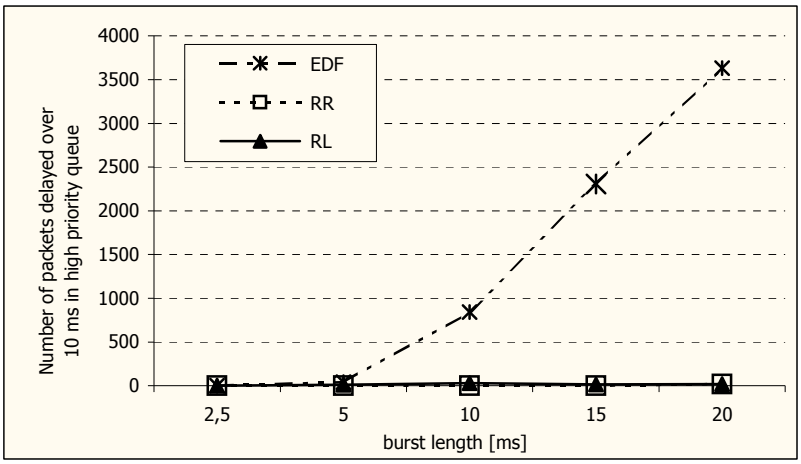

Figure 5.2. Number of high priority packets delayed over 10 ms versus burst length for scenario 1

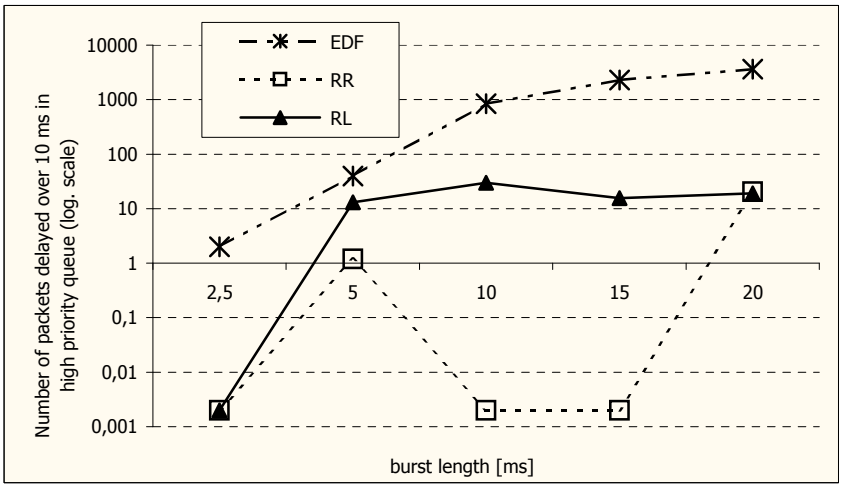

Figure 5.3. Number of high priority packets delayed over 10 ms in logarithmic scale versus burst length for scenario 1

5.3. Scenario 2

In this case the generated burst has the same characteristic as in scenario 1, but this time it is generated by the higher priority class. As shown in Figure 5.4. applying EDF scheduler for the given traffic causes that the number of packets delayed more than 10 ms is smaller than with RR. Round Robin divides the available bandwidth equally among flows which shows that none of the priority mechanisms can be applied. The

novel scheduling algorithm approximates EDF mechanism at this time lowering the number of packets delayed by more than T.

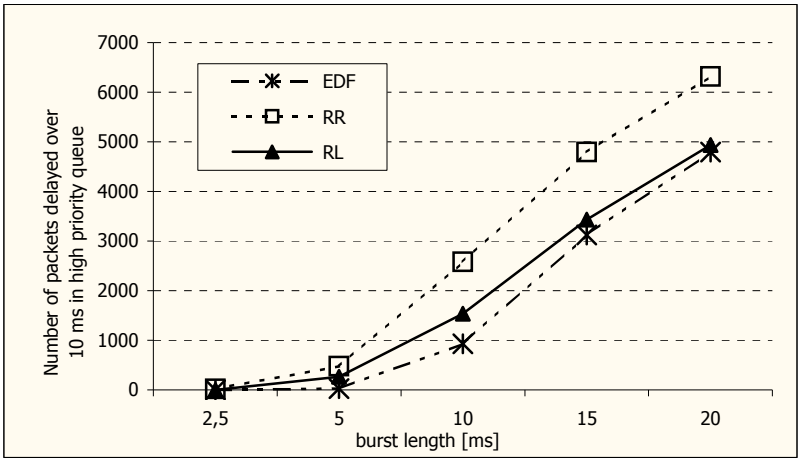

Figure 5.4. Number of high priority packets delayed over 10 ms versus burst length for scenario 2

5.4. Scenario 3

This time the burst is generated by the low level traffic class, with parameters like in scenario 1, except the volume of additional traffic, which is 5 times higher than mean throughput of the low level class. As shown in Figure 5.5. and Figure 5.6. RL algorithm approximates RR scheduler, that is advisable because of the high priority class.

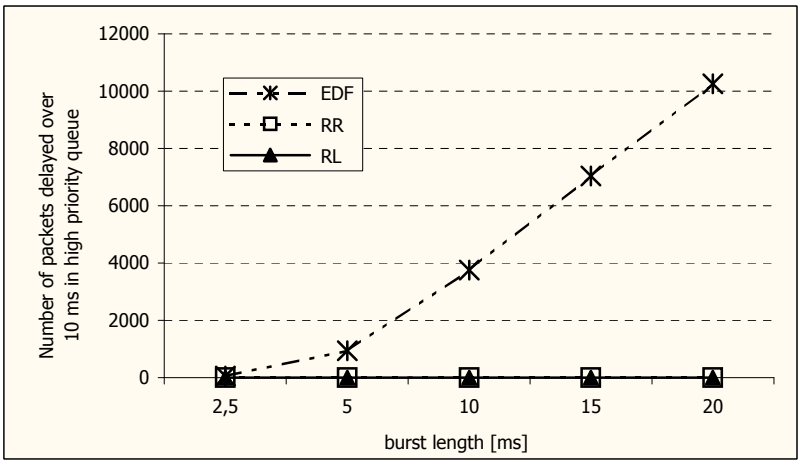

Figure 5.5. Number of high priority packets delayed over 10 ms versus burst length for scenario 3

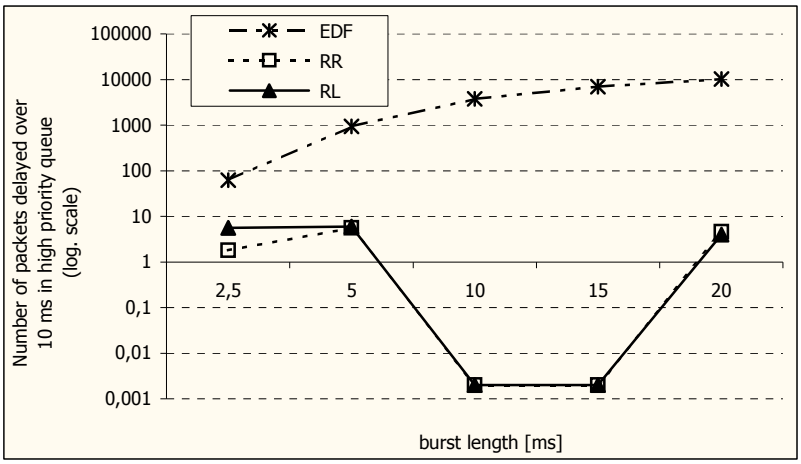

Figure 5.6. Number of high priority packets delayed over 10 ms in logarithmic scale versus burst length for scenario 3

5.5. Scenario 4

In this case the generated burst has the same characteristic as in scenario 3, but this time it is generated by the higher priority class. As shown in Figure 5.7. applying EDF scheduler for the given traffic lowers the number of packets delayed over 10 ms in comparison with RR. The novel scheduling algorithm approximates EDF mechanism that is better for delay sensitive traffic.

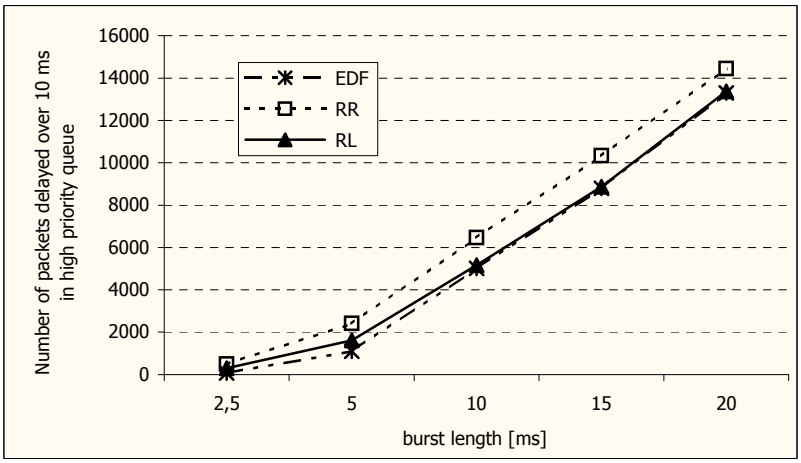

Figure 5.7. Number of high priority packets delayed over 10 ms versus burst length for scenario 4

Burst volume = 2.5 times throughput of mean one class's traffic

Burst generated by	low level class (scenario 1)					high level class (scenario 2)				
Burst Length / Algorithm	2,5 ms	5 ms	10 ms	15 ms	20 ms	2,5 ms	5 ms	10 ms	15 ms	20 ms
EDF	0,0%	0,2%	4,2%	11,5%	18,2%	0,0%	0,2%	4,7%	15,6%	24,0%
RR	0,0%	0,0%	0,0%	0,0%	0,1%	0,1%	2,4%	12,9%	24,0%	31,6%
RL	0,0%	0,1%	0,1%	0,1%	0,1%	0,0%	1,3%	7,7%	17,2%	24,7%

	Burst volume = 5 times throughput of mean one class's traffic									
Burst generated by	low level class (scenario 3)					high level class (scenario 4)				
Burst Length / Algorithm	2,5 ms	5 ms	10 ms	15 ms	20 ms	2,5 ms	5 ms	10 ms	15 ms	20 ms
EDF	0,3%	4,7%	18,8%	35,2%	51,3%	0,4%	5,5%	25,0%	44,0%	66,5%
RR	0,0%	0,0%	0,0%	0,0%	0,0%	2,5%	12,0%	32,4%	51,6%	72,2%
RL	0,0%	0,0%	0,0%	0,0%	0,0%	1,5%	8,0%	25,8%	44,3%	66,9%

Figure 5.8. Number of packets delayed over 10 ms in percents

6. Summary

In the paper new packets scheduling algorithm based on application of reinforcement learning method to adaptive switching between Round Robin and Earliest Deadline First algorithms has been presented. The EDF algorithm in case of bursts traffic tries to lower delay for each of the scheduled classes. This mechanism makes the burst effect propagating from congestion affected queue to other queues. If a high priority class generates bursts of packets, using EDF is advantageous for this class. However, if the burst is caused by a low priority traffic, EDF will worsen the conditions for the high priority traffic. Round Robin divides bandwidth equally among classes. This behavior causes burst to affect only the class of packets that it was generated by. If a low priority class generates burst, the high priority one is not affected by. However, if the burst is caused by the high priority class, its delay parameters will be worsen.

The novel scheduling mechanism uses features that differentiate the EDF and RR methods in order to lower delay of high priority packet class. The simulations showed that reinforcement learning based scheduling mechanism is able to approximate EDF or RR. Moreover, its behavior protects high priority traffic by lowering the number of packets delayed by more than 10 miliseconds. To achieve this effect, a modification of the standard AHC algorithm was made by introducing two different discount factors for each action in a given state.

References

1. P. Cichosz, 1997: Reinforcement Learning by Truncating Temporal Differences, PhD Thesis, Warsaw University of Technology.
2. Cisco IP Telephony QoS Design Guide, www.cisco.com, Cisco Systems 2001.
3. www.cisco.com, www.siemens.com, www.avaya.com, www.alcatel.com
4. R. Guerin U. Pennsylvania, Dept. Elec. Eng. V. Peris IBM T.J. Watson Research Center "Quality-of-Service in Packet Networks Basic Mechanisms and Directions" Computer Networks, vol 31, no 3 February 1999, pp: 169-179.
5. 3GPP, 3G TS 23.207: End-to-end QoS concept and architecture, www.3gpp.org, 2002.
6. “A brief overview of ATM protocol layer LAN emulation and traffic management” Computer Review ACM SIGCOMM April 1995.
7. R. Jain, Congestion Control and Traffic Management in ATM Networks: Recent Advances and A Survey, Computer Networks ISDN Systems Vol. 28, Issue 13 (October 1996) pp: 1723-1738.
8. Adam Grzech, Sterowanie ruchem w sieciach teleinformatycznych, OWPW, Wrocław, 2002.
9. Fukuda K, Takayasu H, Takayasu M., 2000: Origin of critical behavior of Ethernet, Physica A287, pp: 289-301.
10. Zdzisław Papier “Ruch telekomunikacyjny i przeciążenia sieci pakietowych” WKŁ Warszawa 2001.
11. Adam Grzech, Sterowanie ruchem w sieciach teleinformatycznych, OWPW, Wrocław , 2002.
12. R.S. Sutton and A.G. Barto, 1998: Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, p. 89.
13. Leslie P. Kaelbling, Michael L. Littman, 2000: Reinforcement Learning: A Survey, Journal of artificial Intelligence Research 4, 1996, pp: 237-285.
14. P.Cichosz, Systemy uczące się, WNT, Warszawa.