A SIMULATOR FOR A GENERAL PACKET NETWORK DEVICE - SIMULATING A NEW SCHEDULER

Anton Kos, Sašo Tomažič

University of Ljubljana, Faculty of Electrical Engineering
1000 Ljubljana, Tržaška 25, Slovenia

anton.kos@fe.uni-lj.si(Anton Kos)

Abstract

When developing new network protocols, new functionalities of network devices, new traffic models and other novelties, we face the problem of non-existing tools. If we want to simulate the behavior of the network with a newly developed feature we have to develop a new simulator or add a new functionality to an existing simulation tool. Both options can be quite complex and/or time consuming. To facilitate the testing and simulation of a new packet scheduler that we developed during our research we have developed a simulator for a general model of a network device. For this purpose we have used the Modular Simulation language, MODSIM III. It is a general-purpose, modular, block structured language that provides support for object-oriented programming, discrete event simulation and animated graphics. The simulator includes modules for the most important elements and functions of a packet network device, modules for collection of results, and modules for writing the results into a standard format files. We have tested and validated the operation of the simulator with analytically verifiable settings. The simulation and analytical results were practically the same. Encouraged with that we have then simulated newly developed packet schedulers and network device functionalities. Some of the simulation results are presented in this article, more you can find in corresponding references.

Keywords: network device simulator, simulation, packet network, Modsim III, DRR

Presenting Author's Biography

Anton Kos has finished his graduate, masters and Ph.D. studies, in the fields of electronics and telecommunications, at the Faculty of Electrical Engineering, University of Ljubljana. Presently he is a researcher and a teaching assistant at the Laboratory of Communication Devices, Department of Telecommunications, Faculty of Electrical Engineering, University of Ljubljana. His research and teaching work is connected mainly to communication networks, especially IP networks. Inside this field he is focused primarily on the ways and techniques of Quality of Service assurance. He has been involved in the simulation of communication networks for more than ten years. During that time he has been using different network simulation tools and developing simulation models of network devices.

ISBN 978-3-901608-32-2

1 Introduction

Communication networks are constantly evolving. Research and development are bringing ever-new network protocols, new functionalities of network devices, new traffic models and other novelties. If we want to simulate the behavior of the network with a newly developed feature, we most often face the problem of non-existing tools. This problem can be solved with development of a new simulator or with adding a new functionality to an existing simulation tool. Both options can be quite complex and/or time consuming. To facilitate the testing and simulation of a new packet scheduler that we developed during our research we have also developed a simulator for a general model of a network device. In the following sections we first present the motivation and related work. Then we explain the general structure and functionality of a packet network device. Next we present a newly developed schedulker. We introduce the development platform used for our simulator and give the detailed description of the developed simulator, its functionality and operation. At the end we present the testing and validation of the simulator and some interesting simulation results for a new packet scheduler.

2 Motivation and related work

During our research in the field of packet networks and Quality of Service (QoS) assurance we have found a lot of references to a plethora of schedulers and other elements of network devices. While the analytical results are plentiful, simulation results are scarcer. Where they exist, simulations are on one hand carried out by simulators written in non-dedicated simulation languages (like C++ and similar) resulting in non-reusable code, and on the other hand by complex and powerful simulation tools (like NS and similar) where adding a new functionality and gaining quality simulation results is not as trivial as it would seem, extremely expensive or even impossible.

Based on that we have decided to develop a simple model of a general packet network device, which would include all the necessary functionalities. Since it is written in a dedicated high-level object oriented simulation language MODSIM III it is easy to understand, upgrade and modify. Its modularity makes changing a scheduler an easy task. Since it is a network device simulator, it does not include functionalities of a higher protocol levels like TCP or similar.

Since the simulator has been written from scratch it is difficult to find any related work or make detailed comparison to other simulation tools as we have no in-depth experiences with them.

3 A model of a packet network device

In figure 1 we present a simplified model of a network device in packet networks.

Packets arriving at a network device are classified into appropriate queues where they wait to be scheduled for forwarding. Classification, queuing and scheduling are performed at a close co-operation with other elements of a network device, such as admission control (at connection setup or at changing the connection parameters), traffic policing and traffic shaping (during transmission).

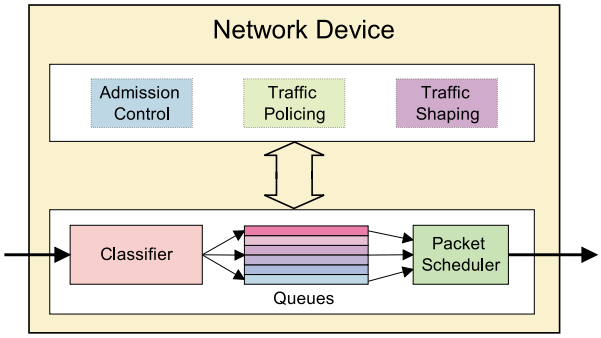

Fig. 1 A simplified model of a packet network device

Let us define the basic notions and elements that concern the operation of a network device.

Network device in packet networks is any device that forwards packets and performs other operations on packets in transmission.

Flow is a group of packets that share the same properties. In most cases the flow is defined by packet source and destination point. The flow can also be defined through values in the packet header fields.

Admission control acts on entire flows. It grants or denies a flow the access to network resources.

Traffic policing acts on packets. It monitors the conformance of each packet to agreed traffic specifications. It can drop the packet or change packet's service levels. Traffic policing can also include marking and metering functions.

Traffic shaping is linked to traffic policing function. It monitors traffic parameters of individual flows and makes sure that they stay within the agreed boundaries (buffering, dropping).

Classifier places the admitted packets into appropriate queues. Classification is usually a simple and operation based on values in packet header fields.

Queues in network devices keep packets awaiting transmission (forwarding). A network device can have one or more queues on each output port. One queue can, for instance, store packets that belong to the same service class or packets that belong to the same flow.

Packet scheduler takes care of the correct order of packet forwarding. The operation of the scheduler and its scheduling algorithm are probably the most important functions of network devices, especially in connection to the quality of service assurance. Let us explain the configuration and control of queues in more detail.

Queue configuration encompasses type, location, and number of queues. In our work we used multiple FIFO

ISBN 978-3-901608-32-2

(First In First Out) queues on output ports of the network device.

Queue control encompasses classification, queue memory management, and scheduling algorithm. In our work we have used service class and flow based classification of packets into queues. We have not used any queue memory management, but the simulator itself does support it. We have used a number of different scheduling algorithms, including a newly developed one.

More detailed explanation of functionality and operation of network devices can be found in [1].

4 Newly developed scheduler

During our research of Quality of Service (QoS) assurance in packet networks, we have found out that the most used and promising, especially for the IP networks, are the two solutions developed through IETF: Integrated Services (IS) [7] and Differentiated Services (DS) [8]-[9]. A more detailed study of mentioned solutions has shown, that both of them have their advantages and disadvantages. When comparing the properties of IS and DS (see [6] for more detail) we concluded that the best solution is to merge the two concepts into a new solution called a network with Differentiated and Integrated Services (DIS) [1], [6].

In a DIS network the type of traffic is recognized and processed accordingly to its QoS demands. Some packets are forwarded without detailed examination; other packets are examined in greater detail and then forwarded, yet another packets will be examined in greater detail, changed accordingly to defined rules and then forwarded.

The operation of DIS network is easiest presented through example. Let us define 6 service classes denoted with letter A to F. In a DS network each of these service classes would be represented with a certain value in IP header DS field.

When a packet travels through a network, each network device on its path checks its service class and acts accordingly. Let us assume that packets from classes D to F do not have high priority. They are placed in the appropriate queue based only on their service class without further processing. The class is determined by the DS field in the packet header. Let us assume that class C packets belong to a signaling protocol. They must be examined in more detail in each network device. Let us assume that that class A and B packets have high priority. They are examined, processed and put into appropriate queue.

It can be said that packets of classes D to F receive only one-stage processing (based on a DS field), while packets from classes A, B and C receive two-stage processing. The first stage is based on their DS field in IP header, where it is determined that they need more detailed processing of the second stage. A simplified processing flow for this example is depicted in figure 2. In our case the traffic of classes A to C requires IS functionality, while for classes D to F the DS functionality suffice. Our proposal is certainly reasonable when classes A and B (and consequently also class C) traffic share is low comparing to class D, E and F traffic.

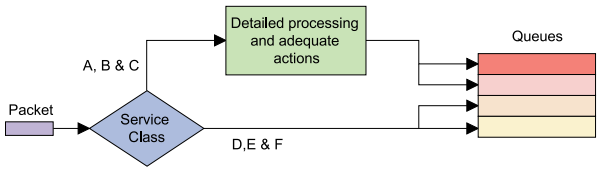

Fig. 2 A simplified packet processing flow in a proposed DIS network

At the first glance, our proposed DIS network would work like a DS network, meaning that the traffic differentiation would be based on traffic classes. Packets would be processed according to the values DS fields in IP header. This mechanism would introduce relative levels of QoS. Since it is anticipated, that for some time the majority of traffic on public IP network would require only relative QoS, the advantages of DS could be exploited.

But for the minority of traffic with more complex and stringent QoS demands a few service classes would be defined. Those classes would receive better service with more processing, similar to IS. In addition to higher demands, such traffic would also use more network resources. This kind of traffic would have to exploit advantages of IS concept. And since it is anticipated that it will only be a small part of the total traffic, disadvantages of IS should not come out. At the same time (at least for this traffic) disadvantages of DS would be eliminated.

At this point it should be stressed, that this is not some sort of ”IS over DS”, but merging of both concepts. In the first stage the network works similar to a DS network. But in the second stage, only for the traffic of certain service classes, the network works similar to an IS network. For more details see [1], [6].

The most critical part of QoS assurance in packet networks and in network devices are schedulers. In general, a scheduler should satisfy the following, sometimes contradictory, demands:

- simple implementation,
- low complexity,
- fairness and flow protection,
- operation inside agreed boundaries,

High speed networks demand simple schedulers that can be implemented in hardware. The complexity should be as low as possible, desirably $O(1)$. This would ensure the scalability, what is not the case with $O(N)$ or $O(\log(N))$ schedulers. Fairness is a desirable feature between flows with the same QoS demands. A

ISBN 978-3-901608-32-2

fair scheduler also provides flow protection (misbehaving flow can not affect the service received by other flows). Operation inside agreed boundaries means that the scheduler has predictable and measurable parameters, for instance latency. These parameters could be set for each individual packet or statistically for each flow.

Since our DIS network introduces two-stage packet processing, it is straightforward to configure output queues in the same manner. This also implies two-stage scheduling. The first scheduling stage should assure different levels of QoS between existing service classes, and the second stage should, when necessary, assure fairness and isolation between flows within the same service class. Based on above conditions, we proposed the use of Strict Priority scheduling on the first stage and the use of Deficit Round Robin (DRR) scheduling on the second stage. The justification for this is:

- Both schedulers have complexity of $O(1)$.
- **Strict Priority scheduling** assures relative QoS levels between service classes. Referring to the example above, we conclude that class A has the highest priority and class F the lowest priority. According to the properties of Strict Priority scheduling, class A packets see the entire link bandwidth, class B packets essentially share bandwidth only with class A packets, and so on till class F packets. Since Class A and B traffic is limited, it should not happen that their packets use entire link bandwidth and in that way starve lower class packets.
- **DRR** scheduling [4] assures fairness and isolation between flows within each service class, meaning that in long term none of the active flows can get more resources than their reservation. Second stage scheduling is reasonable only for high priority service classes like A (guaranteed service) and B (controlled-load service) in our example.

The detailed analytical and simulation analysis of two-stage network operation with two-stage scheduling is presented in [1]. Some results are also presented later in this paper.

5 Development platform

For the development of our simulator of a general packet network device we have used the Modular Simulation language, MODSIM III. It is a general-purpose, modular, strongly typed, block structured simulation language, which provides support for object-oriented programming, discrete event simulation and animated graphics. It is intended to be used for building process-based discrete event simulation models through modular and object-oriented development techniques. MODSIM's syntax and control mechanisms are similar to those of Modula-2, Pascal and Ada.

Simulation capabilities of Modsim III are provided both in library modules and directly through the language. These modules provide a direct support for all capabilities needed to program discrete-event simulation models. All MODSIM III objects have the capability of using Process methods. A "Process" method is a method which can elapse simulation time. A process might WAIT in simulation time and interact at specific simulation times with other processes inside the same object or included into other objects [10].

An extremely useful feature of the MODSIM III libraries is not only a broad choice of modules, functions and objects, but also the fact that most of those objects have the built-in possibility for statistical data collection. That eliminates the need for the time-consuming post-processing statistical analysis of gained simulation results.

6 Simulator

Our simulator of a network device is built of several modules. Each of the modules includes definitions of one or more simulator objects.

Each element or function of a network device (see section 3) is realized in a separate object with its own properties, settings, functionality and behavior. The most important network device objects are: admissioner object, classifier object, queue object, and scheduler object. Depending on settings of a simulation we can have one or multiple instances of listed network device objects. For traffic generation we have traffic generator object, which generates packet objects. In addition to this we have also other objects that take care of reports, reading simulation settings, writing simulation results and other tasks. During the simulation objects interact between themselves, what results in creating new objects, disposing of no longer needed objects, storing objects for later processing, etc.

The simulator is controlled through input files that contain settings and behavior of the simulator. We can, for instance, enable or disable some functions of the network device (admissioner), control the number of instances of a certain object (queues), set the algorithm (scheduler), and more. Simulation results are written into standard format files that provide the possibility of semi-automatic graphic representation of results.

A typical example of a simple simulation scenario would be the following. In a single network device we have multiple input ports and multiple output ports (we monitor one output port). On a monitored output ports we have a scheduler with multiple queues and a classifier. Admissioner is active. Each input port has a dedicated traffic generator with its own traffic pattern and generates one flow.

From a packet's point of view the simulator works like this. When a packet object is generated by one of the traffic generator objects, it is sent to one of the input ports of the network device. It is first processed by admissioner, and if it conforms to its flow specifications, it is admitted and sent to the classifier of the appropriate output port. It is then classified into one of the multiple output queues on that output port. Those queues

ISBN 978-3-901608-32-2

are served by the output port's scheduler according to the scheduling algorithm defined. When the packed is served, its statistics is written to the report file. After that the packet object is disposed. During that time many other parameters of the network device are collected, statistically evaluated, and available in the report files.

7 Simulator testing and validation

We have tested and validated the operation of the simulator with analytically verifiable settings. First we compared the results for the simplest queuing system M/M/1 (see details in [1] and [2]).

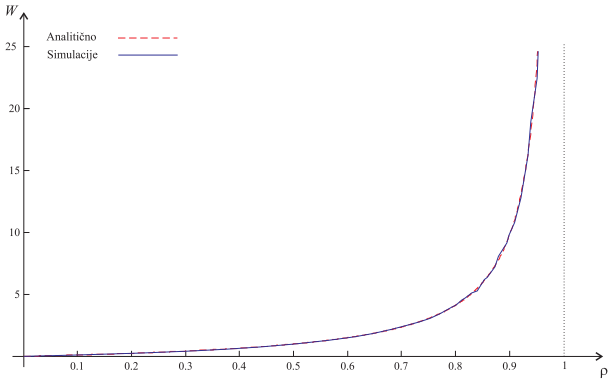

Fig. 3 The comparison of analytical and simulation results in the M/M/1 waiting system for the average waiting time W at system load $0 < \rho < 1$.

In figure 3 we see the results for the average packet waiting time W. Simulation and analytical results are practically the same and almost indistinguishable in the figure. With this result we have in the first place confirmed the correct operation of traffic generators, the correctness of procedures for choosing simulation parameters, and the correctness of their calculations. Since the M/M/1 queuing system is implemented as one FCFS (First Come First Serve) queue, it is hard to say that this also confirms the correct operation of the entire simulator. In the M/M/1 queuing system the network device does not use the functions of admissioner, classifier and scheduler, it essentially works as a buffer.

To test and validate the operation of the network device part of the simulator we have tested a more complex queuing system M/G/1 with priorities for different values of system load ρ and traffic priorities p, where $p = 4$ is the highest priority and $p = 1$ is the lowest priority (see details in [1] and [2]).

In figure 4 we see the results for the average packet waiting time W. As in previous example, also here the simulation and analytical results are practically the same. That confirms the correct operation of the other elements of our simulator (admissioner, classifier and scheduler).

We have done similar validation test for the DRR scheduler, two-stage processing and queuing. More on this can be found in [1].

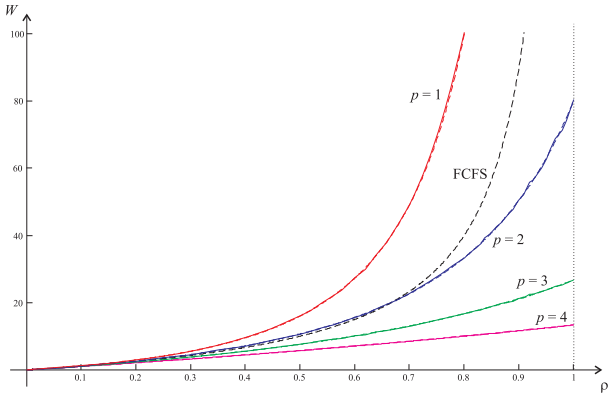

Fig. 4 The comparison of analytical and simulation results in the M/G/1 queuing system with priorities p for the average waiting time W at system load $0 < \rho < 1$.

8 Simulation results for a new scheduler

Encouraged with that we have then simulated our newly developed packet schedulers and network device functionalities. We have carried out extensive simulations with many different scenarios that included many different settings. In this article we present our results only briefly, more you can find in [1].

We have simulated a modified version of DRR scheduler. The operation of an original DRR scheduler is defined in [4], our modifications are explained in detail in [1].The main characteristic of all Deficit Round Robin (DRR) like scheduling algorithms is their ability to provide guaranteed service rates for each flow (queue).

DRR services flows in a strict round-robin order. It has complexity $O(1)$ and it is easy to implement. Its latency is comparable to other frame-based schedulers. A detailed operation of DRR algorithm can be found in [4]. Below is the list of variables used:

R	transmission rate of an output link,
N	the total number of active flows,
r_i	the reserved rate of flow i,
w_i	weight assigned to each flow i,
Q_i	quantum assigned to flow i.

Because all flows share the same output link, a necessary constraint is that the sum of all reserved rates must be less or equal to the transmission rate of the output link:

$$\sum_i r_i \leq R \tag{1}$$

Let r_{min} be the smallest of r_i: $r_{min} = \min_{\forall i} r_i$. Each flow i is assigned a weight that is given by:

$$w_i = \frac{r_i}{r_{min}}. \tag{2}$$

Note that $\forall i \in 1, 2, \cdots, N$ holds $w_i \geq 1$. Each flow i is assigned a quantum of Q_i bits, that is a whole positive value, i.e. $Q_i \in \mathcal{N}$. This quantum is actually

the amount of service that the flow should receive during each round robin service opportunity. Let us define with Q_{min} the minimum of all the quanta. Then the quantum for each flow i is expressed as:

$$Q_i = w_i Q_{min}. \quad (3)$$

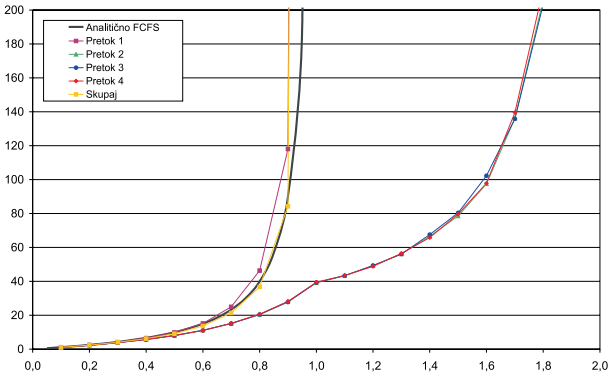

Fig. 5 Average packet delays of DRR scheduler at $Q_i = 2000$ bytes, at exponential packet length distribution. Flow 1 is misbehaved and sends 5 times more data than agreed.

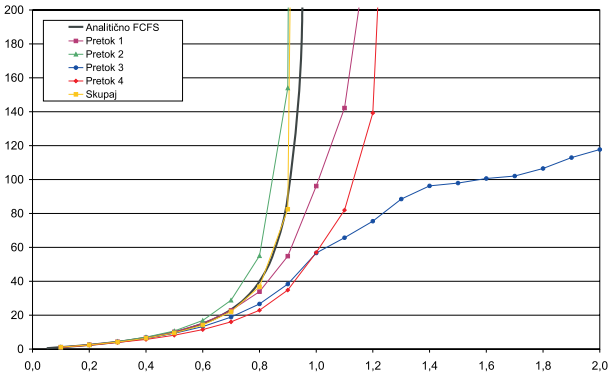

Fig. 6 Average packet delays of DRR scheduler at $Q_1 = Q_2 = 3000$ and $Q_3 = Q_4 = 9000$ bytes, at exponential packet length distribution. Flow 2 is misbehaving.

The consequence of the above constraints and definitions is, that if a flow i sends more data as it is entitled to through the size of its quantum Q_i (misbehaved flow), its queue will become larger and packets will experience greater delay, while the other flows will remain unaffected. This implies that DRR is fair and provides flow protection.

To prove the above claims, we have simulated a scenario where one of the flows is misbehaving and sending 5 times more data that it is entitled to through its quantum.

In figure 5 we see that the delay of the misbehaving flow 1 quickly rises above all boundaries as the system load ρ approaches 1. When ρ exceeds 1 (overload that is caused by the misbehaving flow) it only affects flow 1, the other three flows are unaffected.

In figure 6 we see another scenario with average delays of flows that have quantum values $Q_1 = Q_2 = 3000$ and $Q_3 = Q_4 = 9000$ bytes, and exponential packet length distribution. Flow 2 with reservation $Q_2 = 3000$ bytes is misbehaving - sending more data than agreed. The results show that its delays very quickly rise above all limits while delays of other flows, that behave according to agreed parameters, experience expected delays.

We gain similar results for two-stage scheduler. We use the strict priority scheduler on the first stage and the DRR scheduler on the second stage.

In figure 7 all flows have the same quantum. We see that flows are differentiated by its delay according to the priority class they belong to. But flows inside the same priority class have exactly the same average delay.

Situation is s bit different at the presence of a misbehaving flow. In figure 8 we have one misbehaving flow in each priority class. We see that the delay rises only for the misbehaving flow and does not affect other flows inside the same priority class. But since we have strict priority queuing on the first stage, misbehaving flows of high priority classes affect all the flows in low priority classes.

9 Conclusion

The developed simulator has proven its functionality and provided us with interesting results for a newly developed scheduler. Since the simulator is built-up of modules it can be easily upgraded, we can easily add new functionality or reuse some of its modules in other simulators (what we have recently successfully done).

The main contribution of our work and at the same time also the implication is the simulator itself. The simulator makes a solid foundation for further research - adding a new scheduler, admission policy or classifier settings is easy and straightforward. Another contribution is the proposal of a DIS network with a simulated set of schedulers with proven behavior.

10 References

[1] Anton Kos, Zagotavljanje različnih stopenj kakovosti storitev v omrežjih s paketnim prenosom podatkov. *Ph.D. Thesis*, Faculty of Electrical Engineering, University of Ljubljana, 2006.

[2] Leonard Kleinrock, Queueing Systems, Volume I: Theory. *John Wiley & Sons*, 1975

[3] Robert Verlič, Anton Kos, Sašo Tomažič, Zagotavljanje kakovosti storitev v paketnih omrežjih, *Elektrotehniški vestnik*, vol. 73(2-3), 2006

[4] M. Shreedhar, George Varghese, Efficient Fair Queuing Using Deficit Round Robin. *IEEE/ACM Transactions on Networking*, Volume 4, Issue 3, June 1996.

[5] Anton Kos, Jelena Miletić, Sašo Tomažič, On Latency of Deficit Round Robin. *ERK 2006, Zvezek A, strani 171-174*, Portorož, september 2006.

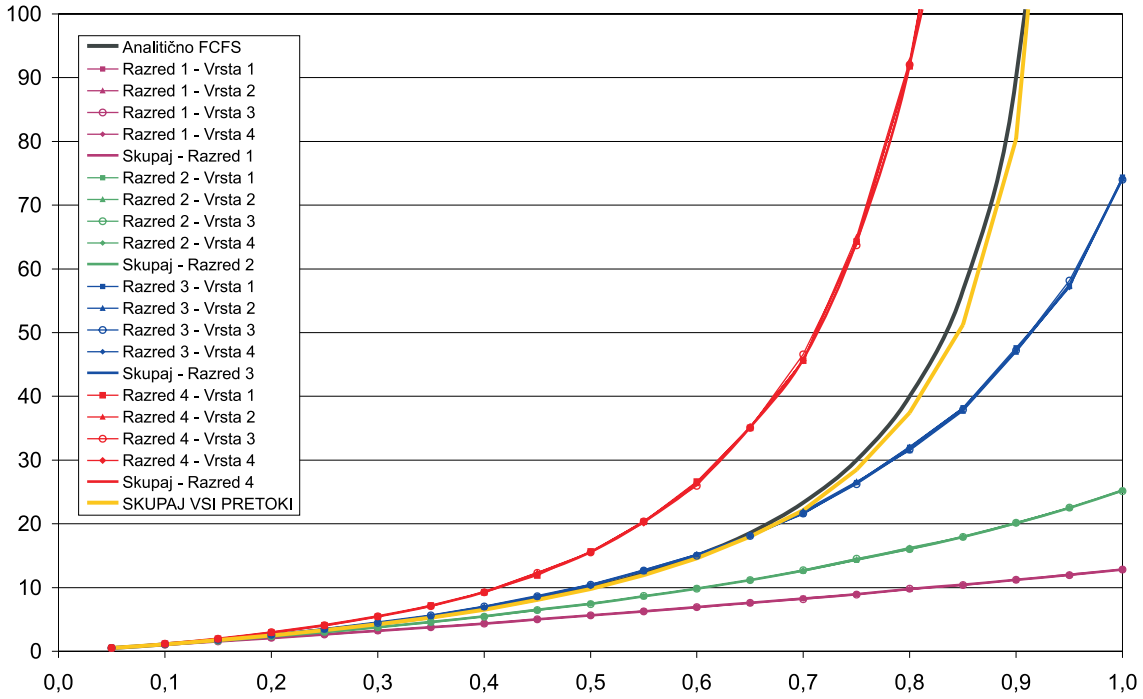

Fig. 7 Average packet delays of two-stage scheduler with strict-priority scheduler on the first stage and DRR (Q = 3000) scheduler on the second stage.

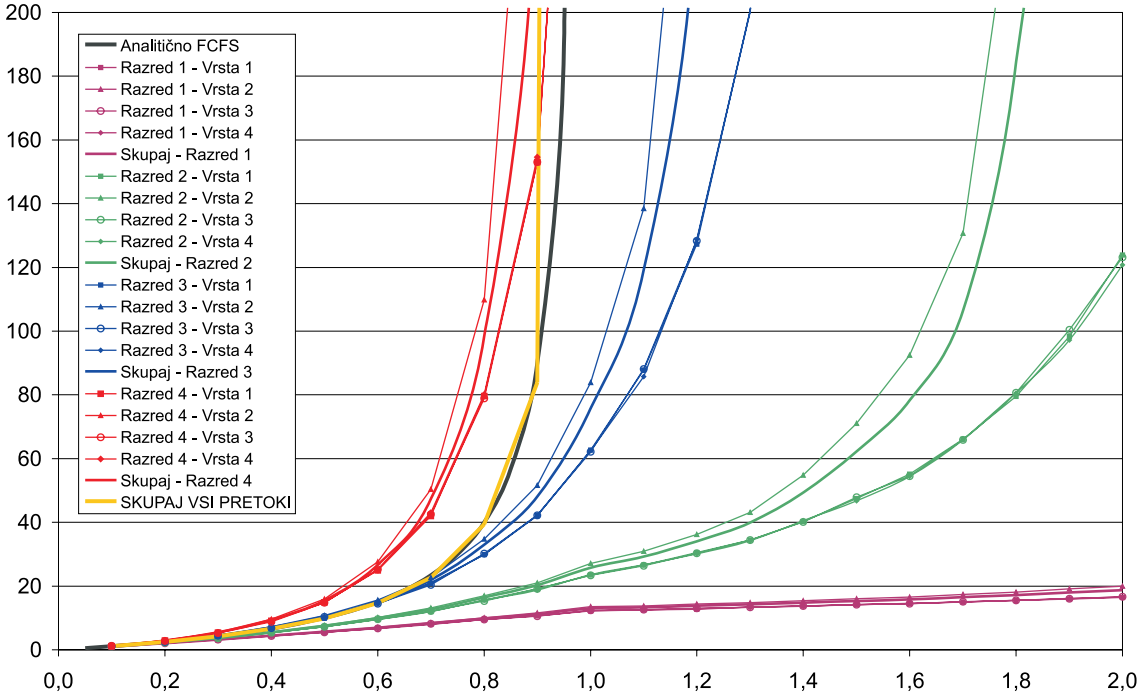

Fig. 8 Average packet delays of two-stage scheduler with strict-priority scheduler on the first stage and DRR (Q = 3000) scheduler on the second stage. Flow 2 in each of the priority classes is misbehaving.

[6] Anton Kos, Sašo Tomažič, Merging of Integrated and Differentiated Services. *VIPSI-2006 Montreal*, Montreal, Canada, 2006

[7] RFC 1633, Integrated Services in the Internet Architecture: an Overview. *IETF*, July 1994

[8] RFC 2474, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. *IETF*, December 1998.

[9] RFC 2475, An Architecture for Differentiated Services. *IETF*, December 1998

[10] CACI, MODSIM Reference Manual. *Reference manual*, CACI Products Company, 1997.

ISBN 978-3-901608-32-2