Virtualization of Links with Partial Isolation and a Packet-Wise Policy

Tomasz Fortuna and Andrzej Chydzinski

Abstract — In this paper a proposition of the packet scheduling algorithm for virtualization of links is presented. The algorithm is based on the time-limited polling model with a constant maximum server attendance time. Such design allows for computing throughput and delay guarantees for the scheduling algorithm and assures that the influence of one virtual link on the performance of other links is limited (i.e. the partial link isolation is provided). Contrary to typical polling models, the scheduling algorithms uses a packet-wise policy, which enables serving packets in a round-robin manner within one operating cycle. This optimization allows for a less CPU-demanding implementation and decreases the queueing delay of the virtual links.

Keywords — partial performance isolation, virtualization of links, work-conserving scheduling, packet-wise scheduler

I. Introduction

Virtualization of networking resources is considered as a possible catalyst for Internet development acceleration. Virtualization allows for merging of several disparate networking layers into single one, based on the same physical medium [2]. Using this technology, the users are able to design a networking topology with desired QoS parameters and use it with a chosen network stack. Reconfiguration does not require any alteration for the physical layer, which can be shared between all topologies, which therefore reduces required hardware and maintenance costs.

This paper deals with packet scheduling, which is one of two mechanisms required for link virtualization (another one being the packet classification). Namely, packets arriving at a physical node are classified according to a special header, which enables handling them distinctly, depending on the virtual network they belong to. Secondly, packets leaving the node need to be ordered onto a common physical link. While the classification is a simple $O(1)$ operation, the scheduling can get quite complex. Parameters of the physical link (the throughput and propagation delay) cannot be altered, but the final parameters of the virtual links depend on the algorithm used for scheduling.

There are many different types of schedulers in use today, but not all of them are fit for the described purpose. This is because creating virtual links requires certain guarantees of their parameters (to make possible guaranteeing quality of service on higher levels of network stack).

The ideal scheduler creates a virtual link which is indistinguishable from a real physical medium. In particular, it provides the full performance isolation, meaning that the parameters and rate of the traffic on one virtual link does not influence the performance of all the remaining virtual links.

Unfortunately, the full performance isolation comes at a cost, i.e. the schedulers having this property are non-work-conserving, which means that they may not fully utilize the physical link. In some cases, in spite of having packets to sent, the physical link is idle, which causes resources to be wasted.

Exemplary scheduler with full isolation is proposed in [3] and is a part of the IIP System design [4], a proposed Future Internet architecture, built using the devices described in [5]. This scheduler attends cyclically all input queues connected with virtual links and transmits packets from each queue for a configured, constant period of time. What is important, each queue is attended for a constant time even if there are no packets in the queue or there is too little time left to send the next packet.

Time during which the scheduler serves a link is called a "work phase" of this link. All repeated phases form a "cycle". By having the constant phase and cycle times, a full performance isolation is easily achieved and the transmission parameters of one virtual link are completely uninfluenced by the other links.

To deal with the resource wastage in the full isolation schedulers, a partly work-conserving scheduler has been proposed in [6]. Its design allows some prioritized, heavy loaded links to benefit in situations where other links do not utilize fully their assigned throughput. This is achieved at a cost of loosing the perfect isolation. However, some boundaries for the performance characteristics are preserved and the algorithm is still useful for scheduling QoS traffic. Therefore, we may say that this scheduler provides the partial performance isolation.

In this paper we focus on possible optimizations of the scheduler with the partial isolation, [6]. We redesign the scheduler (up to some extent) and show, how this affects the virtual links parameters (especialy the latency). Moreover, we argue that the new design is much more implementation-friendly.

The rest of the paper consists of a description of the scheduler model in Section II, followed by Section II-A, which presents the basic analytical background. The deficiencies of the scheduler are highlighted in Section II-B. Then, in Section III, a possible solution is demonstrated. Section IV contains results of simulations for some selected scenarios. Finally, remarks concluding the paper are gathered in Section V.

This is extended version of the paper [1] presented during ITCN'13 conference, Antalya, Turkey, October 8-10, 2013.

The authors are with the Silesian University of Technology, Institute of Informatics. Address: Akademicka 16, 44-100 Gliwice, Poland, e-mail: Tomasz.Fortuna@polsl.pl, Andrzej.Chydzinski@polsl.pl

II. Description of the model

Fig. 1 depicts the model of the scheduler [6].

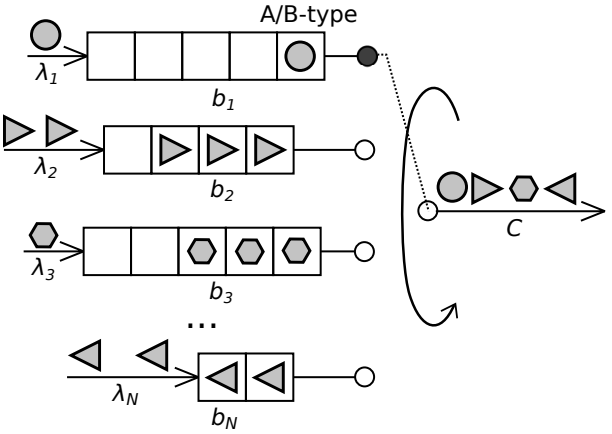

Fig. 1. The model of the scheduler.

There are N incoming packet streams connected to N separate input queues, with limited capacities of $b_1, \cdots, b_N$ packets. The packets from these queues are transmitted cyclicly through an output physical link with capacity of C bits per second, thus creating N virtual links. Each link, say i-th, has defined its maximum work phase length, W_i. This means that the i-th link may be attended for W_i seconds or less during each cycle.

We assume that all the arrival packet streams conform to the Poisson distribution with rates $\lambda_1, \ldots, \lambda_N$. If the queue is full at time when a new packet arrives, the packet is dropped. There is no service position outside the queues, so the packet being sent still occupies a position in the input queue.

Finally, we distinguish two link types – A and B – which differ in the way their service time is handled:

- for a type A link, the work phase duration is constant and equal to W_i, irrespective of the input queue content (it may be empty),
- for a type B link, the work phase ends immediately when the input queue becomes empty during its work phase.

Therefore, for an A-type link the work phase is unaltered, while the work phase of a B-type link may be shorter than W_i, if a queue underflow (empty buffer) occurs.

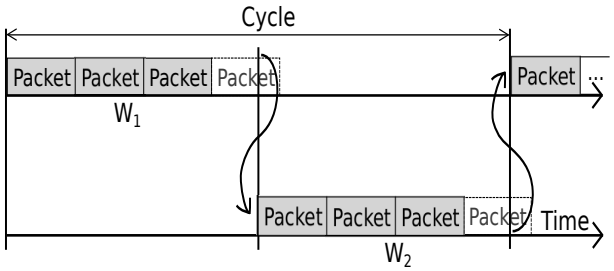

Fig. 2. AA or BB scheduler behavior with non-empty buffers.

Summarizing, the complete scheduler configuration consists of a set of defined b_i's, W_i's and link types. For example, a scheduler with four virtual links of types A, B, A, A will be called the ABAA scheduler.

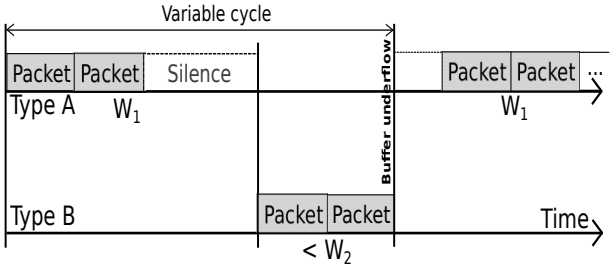

Fig. 3. AB scheduler behavior with a buffer underflow.

In packet networks, packets are atomic units which cannot be further divided. If the remaining link phase time W_i is too short to fit a transmission of the next packet from the queue, the packet will have to remain buffered for the next cycle, and will be transmitted as the first packet in the next phase of the relevant link. In this case the scheduler keeps the physical medium unused, waiting for the remaining W_i time to end. This non-work-conserving behavior causes some loss of the physical link throughput. The scheduler behavior with non-empty input queues is depicted in Fig. 2. Fig. 3 shows behavior of AB scheduler with a queue underflow.

The idea of the A-type links comes from the IIP System scheduler – the pure A-type scheduler works in the same way as the one proposed in [3] (see also [7] for its analysis). The B-type link policy is similar to the exhaustive, time-limited disciplines studied with vacation queues, e.g. [8]-[12].

A. Performance guarantees

Important property of the scheduler with A or B-type links is that it is possible to give boundaries of their performance parameters. Namely, the maximum link response time (latency), T_{max} is given by:

$$\begin{aligned} T_{\max} = & \frac{MTU}{C} \\ & + \left(\left\lceil b_j / \lfloor \frac{W_j \cdot C}{MTU} \rfloor \right\rceil - 1 \right) \sum_{i=1}^{N} W_i + \sum_{i \neq j} W_i \\ & + \left(b_j - \left(\left\lceil b_j / \lfloor \frac{W_j \cdot C}{MTU} \rfloor \right\rceil - 1 \right) \cdot \lfloor \frac{W_j \cdot C}{MTU} \rfloor \right) \\ & \frac{MTU}{C}. \end{aligned} \tag{1}$$

Second guaranteed property is a minimal virtual link throughput under a heavy load, given by:

$$\gamma_{\min} = \frac{W_j - MTU/C}{\sum_{i=1}^{N} W_i}. \tag{2}$$

In these formula, MTU stands for the Maximum Transmission Unit, a size of the largest possible packet. Both formulas are derived in [6].

Described scheduler privileges A-type links in cases where there is usually a low traffic on B-type links with occasional packets bursts. This paper focuses on possible optimizations

of the mixed type scheduler (containing both A and B-type links), which do not affect performance guarantees.

B. Scheduler deficiencies

Implementing the described scheduler as a software solution for Xen Hypervisor is associated with certain problems (see the related papers [13], [15], [16]). Popular computer architectures (like x86 or x86_64) do not behave like real-time systems. An interrupt scheduled to arrive at a certain moment in time will usually suffer a variable delay before being handled by the operating system. This delay is dependent on the current CPU utilization or the CPU model itself. The schedulers [3] and [6] and are defined in the time domain, using defined periods of time. A direct approach to implementation, i.e. using timer interrupts scheduled in future to change currently attended link, will not work. Therefore, the algorithm is rather required to either busy-wait for the right time to swap phases, or use a hybrid approach, where an interrupt is scheduled before the next phase swap and then, for a small amount of time, busy-wait until the packet from the next queue can be transmitted. This approach allows for an accurate implementation of the scheduler, but comes with a cost. The busy-waiting causes a single processor core to be used exclusively for scheduler operations for some time. This might not be a critical problem, as nowadays computers, especially those used for virtualization, are equipped with multiple CPU cores and dedicated computer systems can allocate a single core only for scheduler operations. However, this approach decreases the power-efficiency of used servers, which is usually unacceptable in large data centers.

III. Optimized scheduler

The implementation of the scheduler can be made more efficient by defining its algorithm in a packet-wise manner, instead of phase-wise.

Namely, instead of attending links for their full phase time (or shortened in the case of the type B phase), a different approach might be devised. At the beginning of the cycle, each link is assigned a total service time equal to W_i – its maximum phase duration. Then all links are attended in a round-robin fashion. If a link has enough residual service time left in the current cycle, it is allowed to sent a single packet, as shown in Fig. 4. Naturally, sending a packet reduces the residual service time.

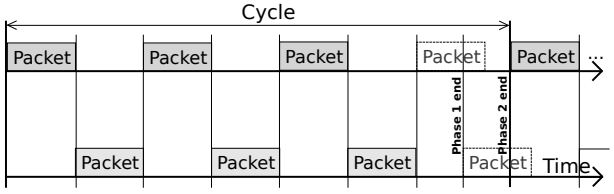

Fig. 4. AA or BB scheduler behavior with full queues.

When the input queue becomes empty, the scheduler behavior remains the same: for B-type links the residual time is zeroed, for A-type links the scheduler waits until a new packet arrives or the time runs out. As long as the number of packets of each link transmitted within a single cycle is not lower than in the phase-wise approach, the scheduler guarantees of minimal virtual link throughput will remain valid. The moment of handling idle time within the cycle can therefore be displaced to happen always at the end of the cycle.

Packet-wise approach has a slight impact on a maximum link response time. By allowing packets from N phases to mix within a single scheduler cycle, the i-th packet in the link queue, which would be served within a current phase time, can be additionally delayed. Instead of being sent after $i-1$ link packets, it can be preceded by a total of $i*N$ packets from all links. Therefore maximum link response time boundary for packet-wise scheduler has the last component in equation (1) multiplied by N.

More generic equation, but less accurate, could instead include a whole additional cycle during which the packet is served. This allows for scheduler which guarantees performance, yet is able to serve queues in a different manner:

$$T_{\max} = \frac{MTU}{C} + \left(\left\lceil b_j / \lfloor \frac{W_j \cdot C}{MTU} \rfloor \right\rceil \right) \sum_{i=1}^{N} W_i + \sum_{i \neq j} W_i \tag{3}$$

The advantages of the presented packet-wise approach are the following:

1) The implementation. The scheduler is able to order more packets to be transmitted during a single interrupt by copying them to the network card buffer. When an A-type link with an empty queue is attended, it can be skipped, as long as the other virtual links can transfer packets in this cycle. Timed interrupt will still be necessary at the end of the cycle.
2) The scheduler achieves a lower latency in scenarios with low bandwidth and occasionally arriving packets. In the phase-wise approach, a packet arriving to an empty queue has to wait for all other links to finish their full phases. In the packet-wise approach, it will be transmitted (pessimistically) when all other links have sent single packets, and the current transmission finishes. This pessimistic time, denoted as T_{empty}, can be easily derived as:

$$T_{empty} = N\frac{MTU}{C}. \tag{4}$$

3) In some settings, the probability of wasting physical medium due to waiting for packets on an A-type link is decreased, because the algorithm is able to transmit packets of other links while the A-type queue is empty.

Another possible improvements of the basic algorithm lies within the handling the time remaining at the end of the work phase. Namely, the introduction of B-typed links made

the algorithm closer to the work-conserving schedulers, when compared to pure A-type schedulers, but still, there are two cases when the output link is wasted:

- in A-type links, when the input queue is empty,
- in both types of links, when the next packet in the queue needs more time for its transmission than is left in the current phase (tail-time).

While the former is important for maintaining the high performance of A-type links (during their idle time, a packet entering currently attended queue could be transmitted immediately), the latter could be altered. Instead of waiting for the phase to end – skip to the next phase. This alteration, called later a *tail optimization*, changes the total cycle length, making it shorter. Because only the idle time is dropped from the cycle, the link guarantees will not degrade. This modification may, however, complicate by far the analysis of the scheduler, when the goal is finding the exact (not boundary) performance characteristics, therefore it will not be used a default configuration.

IV. Results

In the following section we present simulation results which underline the differences between both presented approaches (phase-wise and packet-wise). They were obtained by means of the OMNet++ simulator [14] in version 4.2.2.

Each performed experiment lasted 30s of the simulated time, each queue had capacity of 20 packets, the phase durations were all set to $30\mu s$. There were two virtual links created - the first of type B, the second of type A, served by a physical link of 1Gbps capacity. To both virtual links traffic characterized by Poisson distribution was offered, to the type A link with an unchanging average rate of 50Mbps, to the type-B link of rate increasing from 0 to 600Mbps in 30Mbps increments. Packet sizes were constant and equal to 500 bytes.

Fig. 5 shows average transmission latency of the type-A link. As can be seen, in the phase-wise scheduler, the latency increases as the traffic on type-B link rises. This happens because the average length of the B-type link phase is getting longer as the load increases. Because of that, packets on link A need to wait longer until the phase B finishes.

In the packet-wise approach, the type-A link latency rises only slightly, as the packets get mixed into the B-type phase.

Fig. 6 shows a different effect on the other, B-typed, link. In the beginning, for low link B bitrates, the latency is lower. As the input rate increases and link B saturates the time it is assigned within a cycle, the latency difference between the two approaches disappears. The observed output rate of both links is the same, as can be seen in Fig. 7. The maximum packet latencies are shown in Fig. 8 and Fig. 9.

Figures 10 and 11 compare the average and maximum latency between the packet-wise scheduler and the same scheduler with tail optimization (described at the end of the previous section). In the latter approach, the latency is lower. The difference is the most noticeable if the average packet size is only slightly too large to fill a full link phase.

Now, in Tabs. I-IV the performance of the new scheduler is presented in 15 different scenarios of link configurations and offered loads. Again, 30μ phase durations, 500 byte packets and buffers for 20 packet were used, as well as the physical link of 1Gbps capacity. Analogous table for the phase-wise scheduler can be found in [6] (Tab. 1).

In particular, Tabs. I and II present performance results for the packet-wise scheduler without and with tail optimization, respectively.

Tabs. III and IV present the differences between results for the packet-wise scheduler (without and with tail optimization, respectively) and the phase-wise scheduler ([6], Tab. 1). Cells with bold fonts represent test results better for the packet-wise scheduler – a positive output rate difference and negative latency difference.

As we can see, the packet-wise scheduler with tail optimization outperforms the phase-wise scheduler in terms of the throughput of the virtual links and in terms of the average latency of the virtual links. As for the maximum latency, the packet-wise scheduler is also better in most scenarios. However, in some cases it may offer larger maximum latency.

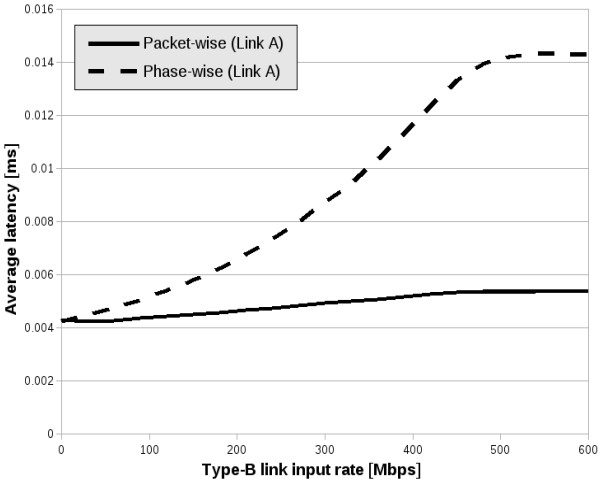

Fig. 5. Average latency of the type-A link.

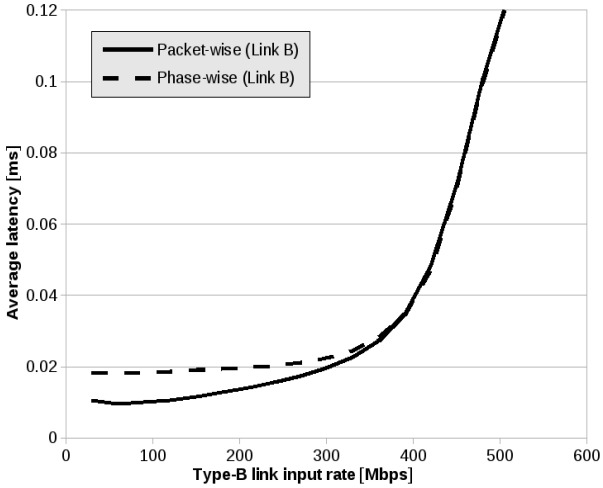

Fig. 6. Average latency of the type-B link.

	Link type			Input rate [Mb/s]			Output rate [Mb/s]			Avg T [μs]			Max T [μs]		
No.	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	A	A		500	500		464.77	464.85		121	120		198	195	
2	A	A		800	200		466.64	199.81		162	7		200	64	
3	A	B		500	500		466.84	464.14		117	122		172	192	
4	B	A		800	200		466.64	199.81		162	7		200	64	
5	A	B		800	200		739.15	199.81		74	20		172	90	
6	A	B		200	200		199.71	199.78		6	13		55	86	
7	A	B		100	100		99.89	99.84		5	11		32	62	
8	A	A	A	333	333	333	309.85	309.91	309.88	180	179	179	286	292	304
9	A	A	A	700	200	100	311.09	200.02	99.96	247	13	7	310	254	89
10	A	A	A	100	100	100	99.82	99.86	99.85	8	8	8	103	97	104
11	A	B	B	333	333	333	311.59	310.12	310.18	173	181	181	258	281	282
12	A	B	B	700	200	100	644.04	200.02	99.96	83	23	23	240	125	113
13	B	B	A	700	200	100	370.96	200.02	99.96	205	36	8	282	259	73
14	B	A	B	700	200	100	419.85	200.02	99.96	179	9	32	280	109	131
15	A	B	B	100	100	100	99.82	99.86	99.85	5	19	18	43	86	87

TABLE I

THE PERFORMANCE OF THE VIRTUAL LINKS IN THE PACKET-WISE SCHEDULER.

	Link type			Input rate [Mb/s]			Output rate [Mb/s]			Avg T [μs]			Max T [μs]		
No.	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	A	A		500	500		487.4	487.44		80	79		189	190	
2	A	A		800	200		486.36	199.82		155	7		193	58	
3	A	B		500	500		491.44	485.96		67	87		160	190	
4	B	A		800	200		486.36	199.82		155	7		193	58	
5	A	B		800	200		775.34	199.81		52	19		160	90	
6	A	B		200	200		199.71	199.78		6	13		47	81	
7	A	B		100	100		99.89	99.84		5	11		29	56	
8	A	A	A	333	333	333	324.58	324.8	324.65	120	117	118	285	278	282
9	A	A	A	700	200	100	321.86	200.02	99.96	238	12	7	306	214	77
10	A	A	A	100	100	100	99.82	99.86	99.85	8	7	7	96	93	96
11	A	B	B	333	333	333	328.48	325.66	325.52	90	120	119	240	263	268
12	A	B	B	700	200	100	675.57	200.02	99.96	59	21	22	210	117	93
13	B	B	A	700	200	100	387.68	200.02	99.96	196	34	7	273	217	73
14	B	A	B	700	200	100	437.85	200.02	99.96	171	8	31	256	87	115
15	A	B	B	100	100	100	99.82	99.86	99.85	5	19	18	39	84	85

TABLE II

THE PERFORMANCE OF THE VIRTUAL LINKS IN THE PACKET-WISE SCHEDULER WITH TAIL OPTIMIZATION.

	Link type			Input rate [Mb/s]			Output rate [Mb/s]			Avg T [μs]			Max T [μs]		
No.	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	A	A		500	500		**0.8**	**0.97**		*4*	*3*		*22*	*19*	
2	A	A		200	800		*-0.03*	*-0.01*		*1*	**-10**		*24*	**-25**	
3	A	B		500	500		**0.53**	**0.15**		*4*	*5*		**-4**	*16*	
4	B	A		800	200		*-0.03*	0		*1*	**-10**		*24*	**-31**	
5	A	B		800	200		*-1.05*	0		*1*	0		**-4**	*2*	
6	A	B		200	200		0	0		**-2**	**-7**		**-5**	**-6**	
7	A	B		100	100		0	0		0	**-8**		**-6**	**-2**	
8	A	A	A	333	333	333	**0.82**	**0.91**	**0.86**	*8*	*7*	*7*	*20*	*26*	*38*
9	A	A	A	700	200	100	*-0.02*	0	0	*3*	**-24**	**-23**	*44*	**-10**	**-45**
10	A	A	A	100	100	100	0	0	0	**-22**	**-22**	**-22**	**-32**	**-29**	**-30**
11	A	B	B	333	333	333	**0.28**	*-0.02*	*-0.02*	*9*	*13*	*13*	**-8**	*15*	*16*
12	A	B	B	700	200	100	*-1.44*	0	0	*1*	0	**-1**	0	**-8**	**-1**
13	B	B	A	700	200	100	*-1.15*	0	0	*3*	**-3**	**-15**	*16*	**-1**	**-55**
14	B	A	B	700	200	100	**14**	0	0	*2*	**-12**	**-2**	*14*	**-27**	*11*
15	A	B	B	100	100	100	0	0	0	**-2**	**-2**	**-3**	**-20**	**-10**	0

TABLE III

PERFORMANCE DIFFERENCES BETWEEN THE PACKET-WISE SCHEDULER AND THE PHASE-WISE SCHEDULER (TAB. 1 IN [6]).

A. Comparison to DRR scheduler

Fig. 12 and 13 depict results of experiments involving a packet-wise, tail-optimised scheduler and a very popular Deficit Round Robin (*DRR*) scheduler. In the experiment, traffic sent through an A-typed link has a constant average rate of 50Mbps. Traffic on a link-B gradually grows from 0Mbps to 720Mbps. Phase times and packet sizes are the same as in the previous experiments – $30\mu s$ and 500B – splitting the physical

	Link type			Input rate [Mb/s]			Output rate [Mb/s]			Avg T [μs]			Max T [μs]		
No.	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	A	A		500	500		**23.43**	**23.56**		**-37**	**-38**		*13*	*14*	
2	A	A		200	800		**19.69**	0		**-6**	**-10**		*17*	**-31**	
3	A	B		500	500		**25.13**	**21.97**		**-46**	**-30**		**-16**	*14*	
4	B	A		800	200		**19.69**	**0.01**		**-6**	**-10**		*17*	**-37**	
5	A	B		800	200		**35.14**	0		**-21**	**-1**		**-16**	*2*	
6	A	B		200	200		0	0		**-2**	**-7**		**-13**	**-11**	
7	A	B		100	100		0	0		0	**-8**		**-9**	**-8**	
8	A	A	A	333	333	333	**15.55**	**15.8**	**15.63**	**-52**	**-55**	**-54**	*19*	*12*	*16*
9	A	A	A	700	200	100	**10.75**	0	0	**-6**	**-25**	**-23**	*40*	**-50**	**-57**
10	A	A	A	100	100	100	0	0	0	**-22**	**-23**	**-23**	**-39**	**-33**	**-38**
11	A	B	B	333	333	333	**17.17**	**15.52**	**15.32**	**-74**	**-48**	**-49**	**-26**	**-3**	*2*
12	A	B	B	700	200	100	**30.09**	0	0	**-23**	**-2**	**-2**	**-30**	**-16**	**-21**
13	B	B	A	700	200	100	**15.57**	0	0	**-6**	**-5**	**-16**	*7*	**-43**	**-55**
14	B	A	B	700	200	100	**17.86**	0	0	**-6**	**-13**	**-3**	**-10**	**-49**	**-5**
15	A	B	B	100	100	100	0	0	0	**-2**	**-2**	**-3**	**-24**	**-12**	**-2**

TABLE IV

PERFORMANCE DIFFERENCES BETWEEN THE PACKET-WISE SCHEDULER WITH TAIL OPTIMIZATION AND THE PHASE-WISE SCHEDULER (TAB. 1 IN [6]).

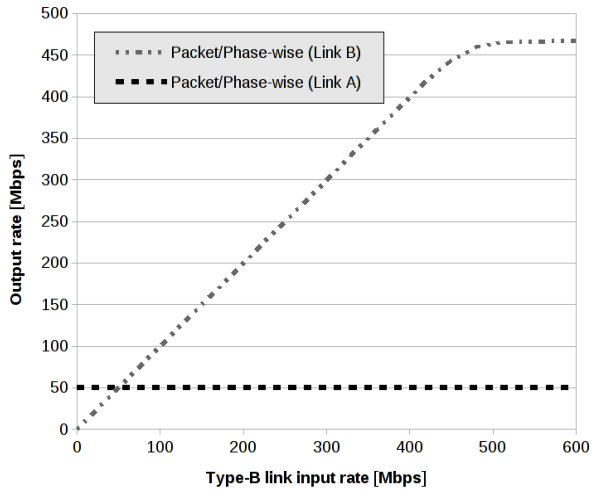

Fig. 7. Rate of links.

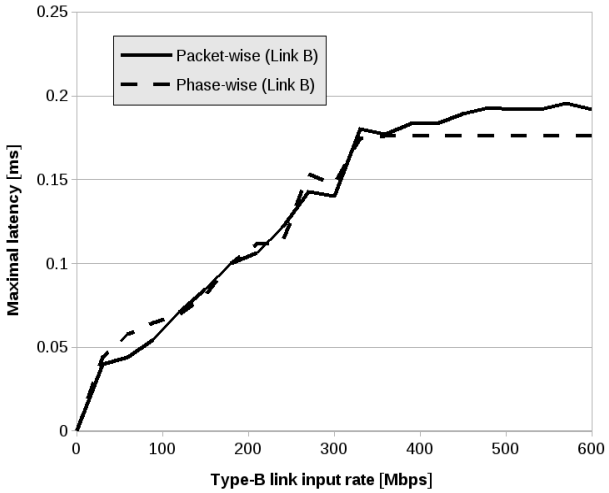

Fig. 9. Maximum latency of the type-B link.

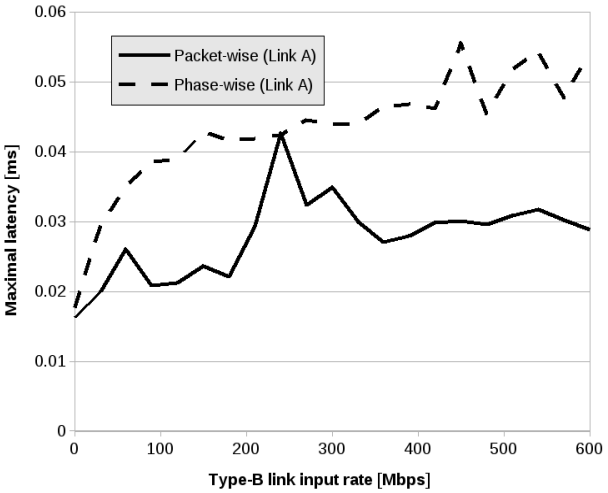

Fig. 8. Maximum latency of the type-A link.

link evenly. DRR does not have a concept of A/B-typed links and treats them equally.

It can be seen that in the moment the B-typed link exceeds its allotted bandwidth, the average latency of A-typed link continues to grow for DRR scheduler case. When the packet-wise scheduler (and phase-wise) is in use, it throttles ill-behaved link as can be seen on Fig. 13.

Both schedulers in this scenario allow link A to transmit without dropping its packets, therefore output rate for both schedulers is exactly the same and depicted as a solid black line on Fig. 13.

V. CONCLUSION

We have analyzed shortcomings of the recently proposed mechanisms for creation of virtual links and suggested two methods of optimization. The described methods can improve the parameters of the created virtual links, especially in terms of lowering the average latency and increasing throughput without giving up scheduling guarantees.

Furthermore, the software implementation of the resulting scheduler needs to wake up the CPU less frequently, which

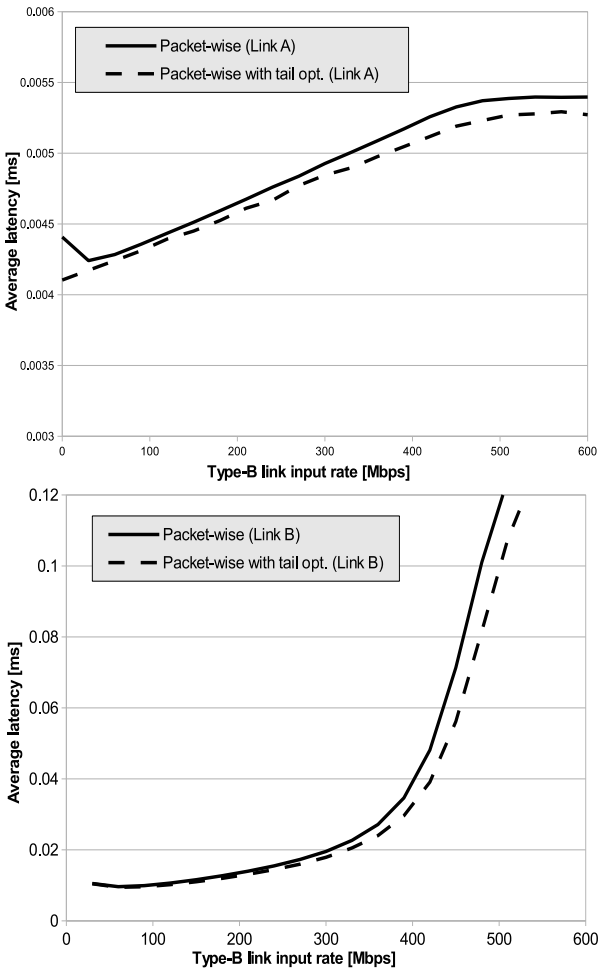

Fig. 10. Average latency of the type-A and B link in scheduler with and without the tail-time optimization.

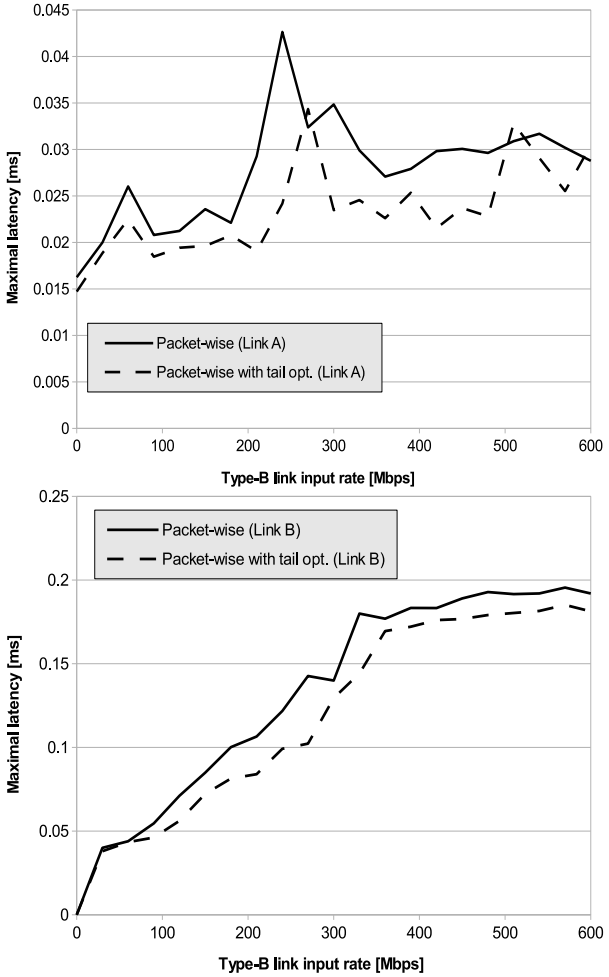

Fig. 11. Maximum latency of the type-A and B link in scheduler with and without the tail-time optimization.

results in a more efficient resource consumption within virtualized environments.

Acknowledgement

This work was partially supported by the Polish National Science Centre under Grant No. N N516 479240.

REFERENCES

[1] T. Fortuna, A. Chydzinski, Packet-wise Scheduler for Virtualization of Links with Partial Performance Isolation, Proc. of International Conference on Information Technology and Computer Networks, pp. 17-23, Antalya, October 2013.

[2] Kurt Tutschku, Towards the Future Internet: virtual networks for convergent services, Elektrotechnik und Informationstechnik 126(7-8), pp. 250–259, 2009.

[3] W. Burakowski et. al. Ideal device supporting virtualization of network infrastructure in System IIP (in Polish), Proc. of KSTiT'11, Lodz, Poland, pp. 818–823, September 14-16, 2011.

[4] W. Burakowski, H. Tarasiuk, A. Beben, G. Danilewicz, Proc. of SNPD, Kyoto, August 2012. IEEE Computer Society, pp. 679–684, August 8–10, 2012.

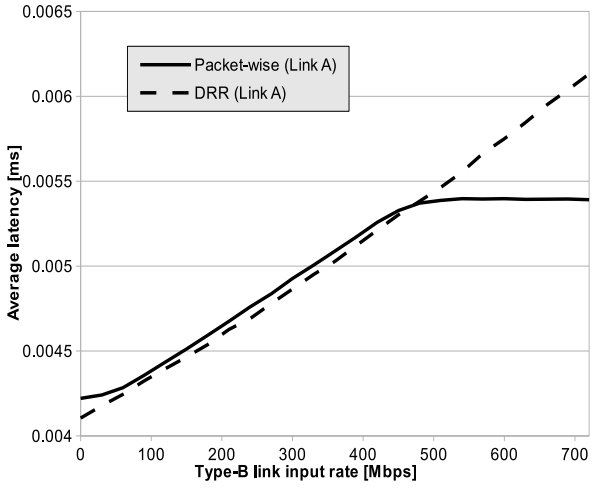

Fig. 12. Comparison of average latency of the type-A and B link between packet-wise tail-optimized scheduler and DRR.

[5] A. Chydzinski, M. Rawski, P. Wisniewski, B. Adamczyk, I. Olszewski, P. Szotkowski, L. Chrost, P. Tomaszewicz, D Parniewicz, Virtualization Devices for Prototyping of Future Internet, Proc. of SNPD, Kyoto, August 2012. IEEE Computer Society, pp. 672-678, 2012.

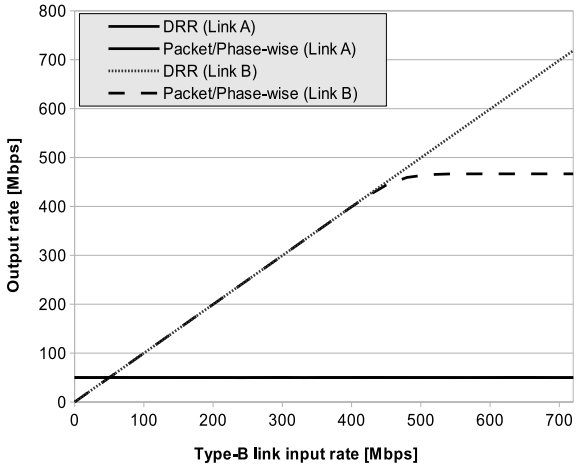

Fig. 13. Comparison of link output rates for a packet-wise scheduler and DRR.

[6] T. Fortuna and A. Chydzinski, Scheduler for Virtualization of Links with Partial Performance Isolation, Communications in Computer and Information Science, Springer, Volume 370, pp. 446–455, 2013.

[7] M. Sosnowski and W. Burakowski, Analysis of the system with vacations under Poissonian input stream and constant service times, Proc. of Polish Teletraffic Symposium, pp. 9–13, Zakopane, December 6-7, 2012.

[8] K. K. Leung and M. Eisenberg, A Single Server Queue with Vacations and Gated Time-Limited Service, IEEE Transactions on Communications, vol. 38, no. 9, pp. 1454–1462, 1990.

[9] K. K. Leung and M. Eisenberg, A single-server queue with vacations and non-gated time-limited service, Performance Evaluation, Volume 12, Issue 2, pp. 115–125, 1991.

[10] H. Takagi and K. K. Leung, Analysis of a discrete-time queueing system with time-limited service, Queueing Systems, Volume 18, Issue 1-2, pp. 183–197, 1994.

[11] T. Katayama, Waiting time analysis for a queueing system with time-limited service and exponential timer, Naval Research Logistics vol. 48, issue 7, pp. 638–651, 2001.

[12] N. Tian and Z. G. Zhang, *Vacation Queueing Models - Theory and Applications*, Springer, New York, 2006.

[13] B. Adamczyk, A. Chydzinski, On the performance isolation across virtual network adapters in Xen, Proc. of International Conference on Cloud Computing, GRIDs, and Virtualization. pp. 222-227, Rome, September 25-30, 2011.

[14] `http://www.omnetpp.org`

[15] B. Adamczyk, A. Chydzinski, Performance Isolation Issues in Network Virtualization in Xen, International Journal on Advances in Networks and Services, vol. 5 no 1 & 2, pp. 139-148, 2012.

[16] T. Fortuna, B. Adamczyk, Improving packet reception and forwarding within virtualized Xen environments, Communications in Computer and Information Science, Springer, Volume 291, 153-160, 2012.

Tomasz Fortuna received his MSc degree in computer science from the Silesian University of Technology, Gliwice, Poland, in 2011. He is interested in a broadly defined information technology and science including computer networks, databases, security and microelectronics. He is currently professionally involved at Ministry of Internal Affairs group COI as a systems architect for a government database systems.

His academic work focuses on the aspects of performance isolation in computer networks. He designs and tests discrete-event network simulators of packet schedulers which later are implemented within the Linux kernel and Xen hypervisor. He is an author of articles on the topic of isolation in network schedulers and problems of packet reception and transmission on the x86 hardware.

Andrzej Chydzinski received his MS (in applied mathematics), PhD (in computer science) and DSc (in computer science) degrees from the Silesian University of Technology, Gliwice, Poland, in 1997, 2002 and 2008, respectively. He is currently a professor in the Institute of Informatics of this university. His academic and professional interests are with computer networking, in particular with performance evaluation of computer networks, Future Internet design, active queue management in Internet routers, mathematical modelling, queueing theory and discrete-event network simulators.

Prof. Chydzinski authored and co-authored 4 books, about 40 conference papers and about 40 journal articles, including papers in Telecommunication Systems, Performance Evaluation, Pattern Recognition, Microprocessors and Microsystems, Queueing Systems, Stochastic Models, Mathematical Problems in Engineering, Applied Mathematical Modelling and other. He is also reviewing articles for several high-quality journals, including IEEE/ACM Transactions on Networking, Annals of Operations Research, Applied Mathematical Modelling, Queueing Systems, Mathematical Problems in Engineering, International Journal of Applied Mathematics and Computer Science, Journal of Network and Computer Applications, Performance Evaluation and other.

Prof. Chydzinski is a Technical Program Committee member for several conferences. He was (and is) a leader of several scientific projects founded by Polish state and European Union. Since January 2011 he has been an IEEE Senior Member.