KU LEUVEN

STADIUS
Center for Dynamical Systems, Signal Processing and Data Analytics

Citation/Reference	Jeremy Van den Eynde, Jeroen Verdyck, Chris Blondia, Marc Moonen (2016), **Delay Performance Enhancement for DSL Networks through Cross-Layer Scheduling** Proc. Of the 6th joint WIC/IEEE SP Symposium on Information Theory and Signal Processing in the Benelux (SITB), pp. 2-9.
Archived version	Final publisher's version / pdf
Published version	https://sites.uclouvain.be/sitb2016/Proceedings_SITB2016.pdf
Journal homepage	http://sites.uclouvain.be/sitb2016/
Author contact	jeroen.verdyck@esat.kuleuven.be + 32 (0)16 324723
IR	https://lirias.kuleuven.be/handle/123456789/539743

(article begins on next page)

Delay Performance Enhancement for DSL Networks through Cross-Layer Scheduling

Jeremy Van den Eynde[1] Jeroen Verdyck[2] Chris Blondia[1] Marc Moonen[2]
[1]University of Antwerp, Department of Mathematics-Computer Sciences
MOSAIC Modeling of Systems And Internet Communication
jeremy.vandeneynde@uantwerpen.be chris.blondia@uantwerpen.be
[2]KU Leuven, Department of Electrical Engineering (ESAT)
STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics
jeroen.verdyck@esat.kuleuven.be marc.moonen@esat.kuleuven.be

Abstract

The quality of experience of many modern network services depends on the delay performance of the underlying communications network. In DSL networks, cross talk introduces competition for bandwidth among users. In such a competitive environment, delay performance is largely determined by the manner in which the cross-layer scheduler assigns bandwidth to the different users. Existing cross-layer schedulers optimize a simple metric, and do not consider important information that is contained within individual packets. In this paper, we present a new cross-layer scheduler, referred to as the minimal delay violation (MDV) scheduler, which optimizes a more elaborate metric that closely resembles the quality of experience of the users. Complementary to the MDV scheduler, a fast physical layer resource allocation algorithm has been developed that is based on network utility maximization. Through simulations, it is shown that the new scheduler outperforms the state of the art in cross-layer scheduling algorithms.

1 Introduction

In communications, maintaining a low delay is important for many applications such as video conferencing, VoIP, gaming, and live streaming. If many delay violations occur, quality of experience (QoE) suffers considerably for these applications. In multi-user communication systems, competition for bandwidth among users motivates the need for a scheduler that assigns bandwidth to the users. This scheduler then has a significant influence on the achieved delay performance of all applications in the network. In DSL networks, competition for bandwidth arises from physical layer resource allocation techniques that combat crosstalk, i.e. interference that results from electromagnetic coupling between different wires in a single cable binder. In the design of a scheduler for DSL systems, these physical layer mechanisms can be taken into account through the framework of cross-layer optimization.

Part of this research work was carried out at UAntwerpen, in the frame of Research Project FWO nr. G.0912.13 'Cross-layer optimization with real-time adaptive dynamic spectrum management for fourth generation broadband access networks'. Part of this research work was carried out carried out at the ESAT Laboratory of KU Leuven, in the frame of 1) KU Leuven Research Council CoE PFV/10/002 (OPTEC), 2) the Interuniversity Attractive Poles Programme initiated by the Belgian Science Policy Office: IUAP P7/23 'Belgian network on stochastic modeling analysis design and optimization of communication systems' (BESTCOM) 2012-2017, 3) Research Project FWO nr. G.0912.13 'Cross-layer optimization with real-time adaptive dynamic spectrum management for fourth generation broadband access networks', 4) IWT O&O Project nr. 140116 'Copper Next-Generation Access'. The scientific responsibility is assumed by its authors.

A cross-layer scheduler makes its scheduling decisions based on the solution to a network utility maximization (NUM) problem. Existing cross-layer schedulers optimize a simple metric, such as queue length, head-of-line delay, or average waiting time, and do not consider important information that is contained within the individual packets. In this paper, we introduce the new minimal delay violation (MDV) scheduler, which optimizes a function of the delay percentile, a measure that closely resembles the true quality of service requirements of delay sensitive traffic. Complementary to the new MDV scheduler, a fast physical layer resource allocation algorithm is developed that solves the corresponding NUM problem. The resource allocation algorithm, referred to as the NUM-DSB algorithm, is inspired by the distributed spectrum balancing (DSB) algorithm for spectrum coordination in DSL networks. The NUM-DSB algorithm decides on the appropriate power allocation for the physical layer, and can be shown to converge to a local optimum of the original NUM problem. Convergence is fast, which enables verification of the MDV scheduling algorithm through simulations.

Simulation results are obtained using the OMNeT++ framework and Matlab. The performance of the MDV scheduler is evaluated in a downstream DSL system, and is compared to the performance of both the max-weight (MW) and the max-delay utility (MDU) scheduler. Simulation results show that the MDV scheduler outperforms the MDU and MW scheduler. The MDV scheduler sometimes also demonstrates better performance with respect to throughput. Overall, when the MDV scheduler is used, it is seen that significantly fewer delay violations occur.

2 DSL system model

2.1 Physical layer

We consider an N user DSL system. DSL employs discrete multitone (DMT) modulation in order to establish K orthogonal sub channels or tones. As signal coordination is assumed not to be available, each of these tones k can be modeled as an interference channel.

$$\boldsymbol{y}_k = H_k \boldsymbol{x}_k + \boldsymbol{z}_k \tag{1}$$

In (1), $\boldsymbol{x}_k = \left[x_k^1, \ldots, x_k^N\right]^T$ is a vector containing the transmitted signal of all N users on tone k. Also, let $\boldsymbol{x}^n = [x_1^n, \ldots, x_K^n]^T$ and let $\boldsymbol{x} = \left[\boldsymbol{x}^{1T}, \ldots, \boldsymbol{x}^{NT}\right]^T$. Similar vector notation will be used for other signals, as well as for variables introduced later such as the bit loading, total power consumption, and data rate. Furthermore, $\boldsymbol{y}_k$ and $\boldsymbol{z}_k$ contain the received signal and noise for all N users on tone k. The average power of x_k^n is given as $s_k^n = \Delta_f \mathcal{E}\left\{|x_k^n|^2\right\}$, with $\mathcal{E}\{\cdot\}$ the expected value operator and Δ_f the tone spacing. Also, $\sigma_k^n = \Delta_f \mathcal{E}\left\{|z_k^n|^2\right\}$ is the average noise power received by user n on tone k. Finally, H_k is the $N \times N$ channel matrix, where $[H_k]_{n,m} = h_k^{n,m}$ is the transfer function between the transmitter of user m and the receiver of user n, evaluated on tone k.

The maximum achievable bit loading for user n on tone k, given transmit powers $\boldsymbol{s}_k$, is calculated as

$$b_k^n(\boldsymbol{s}_k) = \log_2\left(1 + \frac{1}{\Gamma}\frac{|h_k^{n,n}|^2 s_k^n}{\sum_{n \neq m}|h_k^{n,m}|^2 s_k^m + \sigma_k^n}\right), \tag{2}$$

with Γ the SNR gap to capacity, which incorporates the gap between ideal Gaussian signaling and the actual constellation in use. The SNR gap also accounts for the coding gain and noise margin. The data rate of user n, and the total transmit power

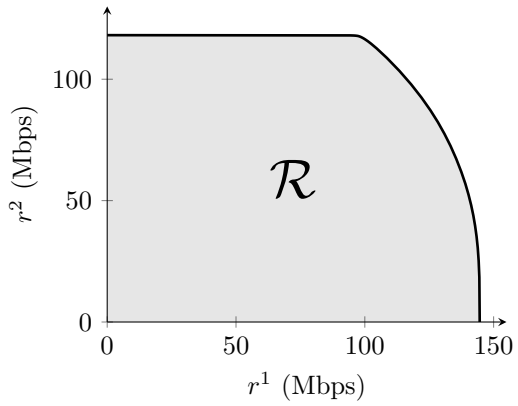

Figure 1: Rate region of a 2-user G.Fast system.

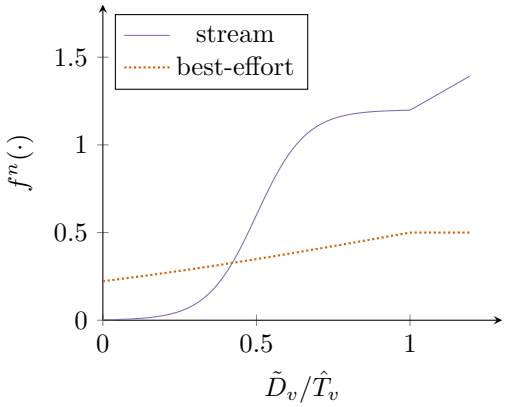

Figure 2: Weight functions for best-effort and streaming applications.

consumption of user n, are given as

$$R^n(\boldsymbol{b}^n) = f_s \sum_{k=1}^{K} b_k^n \qquad P^n(\boldsymbol{s}^n) = \sum_{k=1}^{K} s_k^n, \tag{3}$$

where f_s is the symbol rate.

The total transmit power of each user is limited to P^{tot}. The set of all possible power loadings of user n can thus be described as

$$\mathcal{S}^n = \left\{ \boldsymbol{s}^n \in \mathbb{R}_+^K \mid \; P^n(\boldsymbol{s}^n) \leq P^{\text{tot}} \right\}. \tag{4}$$

The set of all possible power loadings of the whole multi-user system is $\mathcal{S} = \mathcal{S}^1 \times \ldots \times \mathcal{S}^N$. The resulting set of achievable bit loadings is

$$\mathcal{B} = \boldsymbol{b}(\mathcal{S}) \tag{5}$$

Finally, we define the rate region as

$$\mathcal{R} = \left\{ \boldsymbol{r} \in \mathbb{R}_+^N \mid \exists\, \boldsymbol{r}' \in \boldsymbol{R}(\mathcal{B}) : \boldsymbol{r} \leq \boldsymbol{r}' \right\}. \tag{6}$$

For DSL networks with tone spacing small relative to the coherence bandwidth of the power transfer function, the rate region is a convex set [1].

As an example, the rate region of a 2-user G.Fast system that employs spectrum coordination is depicted in Figure 1. Generally, there is no power allocation that simultaneously maximizes the data rate of all users, as observed in the rate region of Figure 1. Instead, there are a number of Pareto optimal power allocation settings that achieve a data rate on the edge of the rate region. This implies the need for scheduling, i.e. choosing one of these Pareto optimal power allocation settings as the point of operation.

2.2 Upper layer & scheduling

The scheduling occurs in the upper layer, since it has the information that can help deciding the optimal point of operation. We assume that each of the N users has one traffic stream with delay upper bound $\hat{T}^n$ and allowed violation probability ϵ^n, or equivalently, conformance probability $\eta^n = 1 - \epsilon^n$. Time is divided in slots of length

τ. At slot $t \in \mathbb{N}$ the upper layer requests the physical layer for new rates, based on all available info up to time t, such as queue lengths and arrival rates. At the start of slot $t+1$, rates $\boldsymbol{r}(t+1)$ are applied in the interval $[t+1, t+2[$. There is thus a delay τ between the request and application of rates.

Traffic arrives in an infinite buffer. We denote by $a_l^n(t)$ and $Q_l^n(t)$ respectively the arrival time and length in bit of user n's l-th queued packet at the beginning of time slot t, and $Q^n(t) = \sum_{l=0}^{N^n(t)-1} Q_l^n(t)$ where $N^n(t)$ is the number of packets in user n's queue.

At the start of every slot the scheduler has to find a feasible scheduling policy that maximizes the system performance with respect to the QoS requirements. Such a policy will pick a rate $\boldsymbol{r}$ within the rate region $\mathcal{R}$. The requirements are expressed using utility functions. Such a function $u^n(r^n)$ quantifies the usefulness to user n of receiving a service r^n. Data rates $\boldsymbol{r} \in \mathcal{R}$ are then selected such that they maximize the sum of the utilities.

$$\arg\max_{\boldsymbol{r} \in \mathcal{R}} \sum_{n=1}^{N} u^n(r^n) \tag{7}$$

Ideally, $u^n(\cdot)$ is monotonically increasing, concave, and differentiable for all n.

A large family of scheduling algorithms is linear in $\boldsymbol{r}$, i.e.

$$u^n(r^n) = \omega^n r^n. \tag{8}$$

For example, the Max-Weight scheduler (MW) [2] has $\omega^n(t) = Q^n(t)$. For the Max-Delay Utility (MDU) scheduler [3], the authors give $\omega^n(t) = \frac{|u'^n(\bar{W}^n)|}{\bar{\lambda}^n}$, where u'^n is the derivative of the utility function, $\bar{W}^n$ the average waiting time, and $\bar{\lambda}^n$ the average arrival rate. It is important to note that for these linear scheduling algorithms, efficient DSL physical layer resource allocation algorithms exist [4].

The QoS requirements are expressed as a delay upper bound T^n with delay violation probability ϵ^n: $P\{D^n > \hat{T}^n\} \leq \epsilon^n$, where D^n is the packet's delay. If this delay exceeds the upper bound T^n, the packet is useless to the application. The considered performance metrics are delay violations and throughput.

3 Minimal Delay Violation Scheduler

In general, schedulers that take QoS into account aim to minimize the average delay. However, this metric offers a skewed view. Imagine twenty packets, alternating between a delay of 5ms and 55ms. This gives an average delay of 30ms per packet. If the delay requirements were 40ms, then 50% of the packets could be considered useless.

The Minimal Delay Violation (MDV) scheduler aims to minimize the delay violations, rather than the average delay. First it estimates the η^n-percentile delay $\tilde{D}^n(t)$ for the coming slots, based on the queue and observed past delays. Then, depending on the proximity of $\tilde{D}^n(t)$ to $\hat{T}^n$, a weight is defined for the user to reflect its importance. For example, if for a video v the normalized delay $\frac{\tilde{D}^v(t)}{\hat{T}^v}$ is small, then v is not important, as its delay requirements will probably not be violated, and hence it can have a lower rate assigned. If, on the other hand, $\frac{\tilde{D}^v(t)}{\hat{T}^v}$ approaches 1, then its weight should be much larger, to express it is approaching its delay upper bound.

This updated delay is then finally converted into a bit length c^n which, when divided by $r^n(t+1)$, gives an approximation to the η^n-percentile of user n's delay. It is this $\boldsymbol{c}$ that is passed on to the physical layer to find the optimal rates $\boldsymbol{r}$.

The Minimal Delay Violation (MDV) scheduler uses the utility function $u^n(r) = -\frac{c^n}{r^n}$, which is increasing, concave and differentiable on $]0, +\infty[$. At the start of every

slot, it minimizes the average of all users' η^n-percentile of the delay:

$$\arg\max_{\boldsymbol{r}\in\mathcal{R}} \sum_{n=1}^{N} -\frac{c^n(t)}{r^n} = \arg\min_{\boldsymbol{r}\in\mathcal{R}} \sum_{n=1}^{N} \frac{c^n(t)}{r^n} \tag{9}$$

We now look how c^n is constructed. Let's call $\tilde{D}^n(t) = \alpha^n \frac{\bar{q}^n(t)}{\tilde{\lambda}^n(t)} + (1-\alpha^n)\bar{d}^n(t)$, the weighted average of predicted and observed delays. Here $\alpha^n \in [0,1]$ indicates the importance of the queue. A small value means that mainly past behavior, i.e. $\bar{d}^n(t)$ which is the η^n-percentile of past delays, will influence the weight. This is useful for users that prefer a long-term average data rate, such as background jobs. A large α^n on the other hand will place more importance on the predicted delay $\frac{\bar{q}^n(t)}{\tilde{\lambda}^n(t)}$. Here $\bar{q}^n(t)$ is a measure for the queue and further explained below, and $\tilde{\lambda}^n(t) = \frac{1}{4}(\bar{\lambda}^n(t) + \sum_{s=t-2}^{t} r^n(s))$ is an estimate of the future $r^n(t+1)$, with $\bar{\lambda}^n(t)$ an average of the arrival rate. Streaming traffic benefits from this, as it can fluctuate heavily.

$\bar{q}^n(t)$ is the η^n-percentile of the user's cumulative queue size $\check{Q}^n_l(t)$, $l \in [0, N^n - 1]$:

$$\check{Q}^n_l(t) = a^n_0 r^n + \sum_{m=0}^{l'-1} Q^n_m \frac{\tilde{\lambda}^n}{r^n} + \sum_{m=l'}^{l} Q^n_m$$

The first term accounts for the head-of-line delay. The second for the packets that will be sent in the interval $[t, t+1]$, for which we already know the rates. l' is the number of packets that are transmitted in $[t, t+1[$. The final term accounts for the packets that depart in the slots $[t+1, \dots[$ at a yet unknown rate. The delay of queued packet l at a rate r^n can now simply be calculated using $\check{Q}^n_l / r^n$.

The parameter $\boldsymbol{c}$ can be expressed by $c^n = \left[\tilde{\lambda}^n \hat{T}^n f^n(\frac{\tilde{D}^n}{\hat{T}^n})\right]_1$.

The weight function $f^n(\cdot)$ transforms its argument, the proximity to $\hat{T}^n$, into a weight that reflects its importance with respect to the QoS requirements. The following functions have been defined

$f_{stream}(d) = s(\gamma = 1.2, \mu = 0.5, \sigma = 0.08, \rho = 1, x = d)$

$f_{be}(d) = s(\gamma = 1.0, \mu = 1.0, \sigma = 0.80, \rho = 0, x = d)$

with

$$s(\gamma, \mu, \sigma, \rho, x) = \begin{cases} S(x) & \text{if } x \leq 1 \\ S(1) + (x-1)\rho & \text{if } x > 1 \end{cases}$$

and the sigmoid

$$S(x) = \frac{\gamma}{1 + e^{-\frac{x-\mu}{\sigma}}}$$

They are depicted in Figure 2. These functions are tuned such that video and best-effort cooperate: if a video's delay is low then it will spare best-effort channel capacity. However if the video's delay is close to or over its delay upper bound, its weight will increase more quickly than best-effort's, which causes video's rate to increase at the cost of best-effort receiving less capacity.

4 Distributed Spectrum Balancing for Network Utility Maximization

Here, the NUM-DSB algorithm is delineated, which solves an instance of (7) for every slot t. NUM-DSB yields the optimal data rate $\boldsymbol{r}^*$, as well as the corresponding power

allocation $\boldsymbol{s}^*$. The NUM problem is non-convex on account of the bit loading being a non-convex function of the power allocation (2). Inspired by the DSB algorithm for spectrum coordination [4], our solution strategy is to construct successive per-user approximations of the rate region by defining an approximation for the bit loading that is a convex function of the power allocation. By iteratively constructing new approximations at the solution of the previous iteration, a local solution, i.e. a stationary point, of the original problem can be found.

In each iteration ℓ of the NUM-DSB algorithm, a user n will construct its own convex inner approximation of the original rate region $\mathcal{R}$. The approximation of $\mathcal{R}$ depends on the current power allocation $\boldsymbol{s}^{(\ell)}$, and is denoted as $\tilde{\mathcal{R}}(\boldsymbol{s}^{(\ell)})$. Let it be clear that, although this is not reflected in notation, the approximation $\tilde{\mathcal{R}}(\boldsymbol{s}^{(\ell)})$ is specific to user n. In order to construct $\tilde{\mathcal{R}}(\boldsymbol{s}^{(\ell)})$, it is assumed that all other users do not change their power allocation, i.e. $\boldsymbol{s}^m = \boldsymbol{s}^{m(\ell)}, \forall m \neq n$. Furthermore, the bit loading of all other users m is approximated with a lower bound hyperplane, i.e.

$$\tilde{\boldsymbol{b}}^n(\boldsymbol{s}^n; \boldsymbol{s}^{(\ell)}) = \boldsymbol{b}^n(\boldsymbol{s}) \tag{10}$$

$$\tilde{\boldsymbol{b}}^m(\boldsymbol{s}^n; \boldsymbol{s}^{(\ell)}) = \boldsymbol{b}^m(\boldsymbol{s}^{(\ell)}) + \boldsymbol{\beta}^m(\boldsymbol{s}^{(\ell)}) \circ \left(\boldsymbol{s}^n - \boldsymbol{s}^{n(\ell)}\right), \tag{11}$$

where $A \circ B$ denotes the Hadamard product of matrices A and B, and with $\beta_k^m(\boldsymbol{s}_k{}^{(\ell)})$ the directional derivative of $b_k^m(\cdot)$ at $\boldsymbol{s}_k{}^{(\ell)}$ along the nth vector in the standard basis of $\mathbb{R}^n$. We want to guarantee that the value of the approximate bit loading $\tilde{b}_k^m$ remains non-negative. This can be ensured by adding a constraint on $\boldsymbol{s}^n$. Keeping in mind that $\beta_k^m(\boldsymbol{s}_k{}^{(\ell)}) < 0$, the appropriate constraint is

$$s_k^n \leq \hat{s}_k = s_k^{n(\ell)} - \max_{m \neq n} \frac{b_k^m(\boldsymbol{s}_k{}^{(\ell)})}{\beta_k^m(\boldsymbol{s}_k{}^{(\ell)})}. \tag{12}$$

The corresponding sets of all possible power loadings and resulting achievable approximate bit loadings are

$$\tilde{\mathcal{S}}^n(\boldsymbol{s}^{(\ell)}) = \{\boldsymbol{s}^n \in \mathcal{S}^n \mid \boldsymbol{s}^n \leq \hat{\boldsymbol{s}}\} \qquad \tilde{\mathcal{B}}(\boldsymbol{s}^{(\ell)}) = \tilde{\boldsymbol{b}}\big(\tilde{\mathcal{S}}^n(\boldsymbol{s}^{(\ell)}); \boldsymbol{s}^{(\ell)}\big). \tag{13}$$

Finally, the approximate rate region is defined as

$$\tilde{\mathcal{R}}(\boldsymbol{s}^{(\ell)}) = \left\{\boldsymbol{r} \in \mathbb{R}_+^N \mid \exists\, \boldsymbol{r}' \in \boldsymbol{R}\big(\tilde{\mathcal{B}}(\boldsymbol{s}^{(\ell)})\big) : \boldsymbol{r} \leq \boldsymbol{r}'\right\}. \tag{14}$$

User n thus solves the following problem, and extracts the power allocation $\boldsymbol{s}^n$ that achieves the optimal $\boldsymbol{r}$.

$$\arg\max_{\boldsymbol{r} \in \tilde{\mathcal{R}}(\tilde{\boldsymbol{s}})} \sum_{n=1}^{N} u^n(r^n) \tag{15}$$

The algorithm of choice to solve (15) is the Frank-Wolfe algorithm, which exhibits linear convergence [5] and requires no parameter tuning. This algorithm can be used as the utilities $u^n(\cdot)$ are concave and continuously differentiable by assumption, and as the rate region $\tilde{\mathcal{R}}(\boldsymbol{s}^{(\ell)})$ can be shown to be a compact convex set. The delails of the optimization algorithm are however omitted for conciseness. Then, after problem (15) has been solved, a subsequent approximation is constructed by another user at the obtained power allocation. The solutions of these successive approximations can be shown to converge to a stationary point of (7).

5 Performance

5.1 Simulation setup

The simulation consists of two parts. The NUM-DSB algorithm which is run in Matlab. The simulation of the network and upper layer scheduling is run in the OMNeT++ framework. Every $\tau = 50$ms , OMNeT++ gathers $\boldsymbol{c}$, and sends it to Matlab using the *MATLAB Engine API for C*. In the next slot, the rates $\boldsymbol{r}$ are read from Matlab, and applied to the simulated channels.

The physical layer parameters are the following. The transfer function and noise are obtained from a 99% worst case model for the physical layer of a G.Fast system with $N = 2$ users, where the respective line lengths are 450m for $n = 1$, and 390m for $n = 2$. The twisted pair cables have a line diameter of $0,5$mm, which corresponds to 24AWG. For a G.Fast system, the available per-user total transmit power is $P^{\text{tot}} = 4$dBm, the symbol rate is $f_s = 4009$Hz, the number of tones is $K = 2047$, and the tone spacing is $\Delta_f = 51.75$kHz. The SNR gap is chosen to be $\Gamma = 12.6$dB, which corresponds to BER $= 10^{-7}$, a coding gain of 3dB, and a noise margin of 6dB. The rate region that corresponds to these physical layer parameter settings is depicted in Figure 1.

The performance of the network is evaluated for 12 different traffic scenarios. Every scenario is the equivalent of one hour simulated time. Each of the N users is assigned exactly one traffic stream, the characteristics of which depend on the traffic scenario. A mix of three different kinds of traffic has been used. For video traffic, "Starwars" and "Alice in Wonderland" [6] and a 4k video entitled "The Beauty of Taiwan"* are used. Each video's packet lengths are multiplied by a constant such that the load would be closer to 1. For the second type of traffic, arrivals are determined by a Poisson process with fixed-length packets. The final traffic type kept the user's queue backlogged at all times, saturating the line. The users send packets that are encapsulated in UDP datagrams. At arrival at the next hop, the delay statistics of unfragmented packets are tracked.

5.2 Results

The simulations have been executed for the MDV scheduler, as well as for the MDU and MW scheduler. Results are displayed in Figure 3. The left plot shows the percentage of packets that violate their delay requirements. On average, MW has 7.2% of delay violations, MDU 7.4% and MDV 5.6%. Both MDV and MDU have non-zero violations in four scenarios, while MW violates delays in nine scenarios. These violations for MDV and MDU occur for scenarios in which the 4k video was playing, a very bursty video. On three out of the four scenarios, MDV outperforms MDU. The right plot of Figure 3 shows the throughput in Mbps. The results show that on average the MDV scheduler has a higher throughput (122.8 Mbps) than both the MW and MDU scheduler (121 Mbps), with differences of up to 7 Mbps (compared to MDU).

6 Conclusion

The novel cross-layer MDV scheduler has been presented, which employs a utility function to communicate its rate requirements to the physical layer. An accompanying power allocation algorithm for the physical layer (NUM-DSB) has been developed. NUM-DSB displays exceedingly fast convergence, which in turn enables the efficient

*`http://tempestvideos.skyfire.com/Sales_Optimization_Demo/beauty_taiwan_4k_final-ed.mp4`

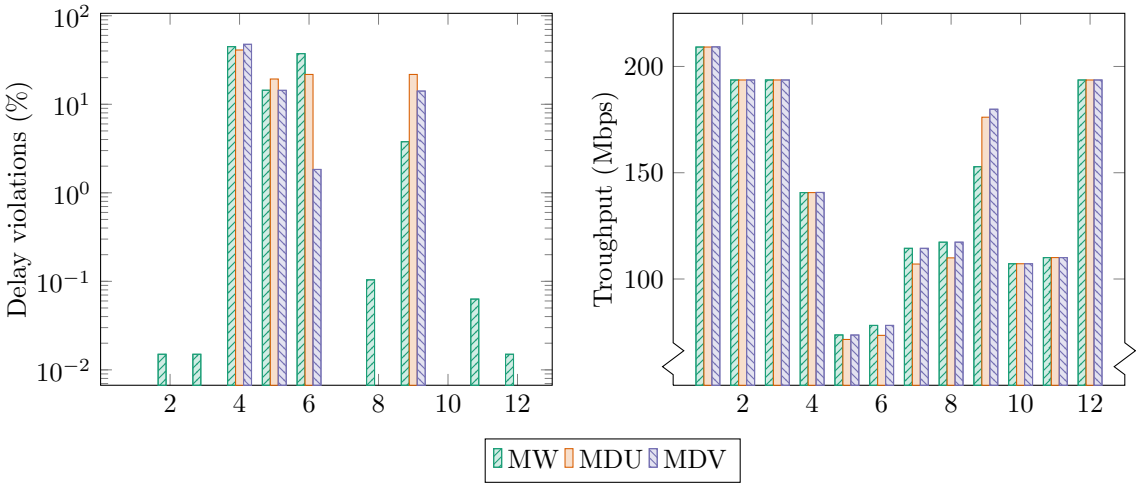

Figure 3: Delay violations (left) and throughput results (right) for the MW, MDU, and MDV schedulers. The results are displayed for 12 different traffic setups (x-axis).

execution of computer simulations that evaluate the performance of the different schedulers. These simulations have shown that, when compared to the MW and MDU scheduler, the MDV scheduler displays a significant performance improvement.

References

[1] R. Cendrillon, Wei Yu, M. Moonen, J. Verlinden, and T. Bostoen, "Optimal multiuser spectrum balancing for digital subscriber lines," *IEEE Transactions on Communications*, vol. 54, no. 5, pp. 922–933, may 2006. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1632106

[2] M. J. Neely, "Delay analysis for maximal scheduling in wireless networks with bursty traffic," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE.* IEEE, 2008.

[3] G. Song, Y. Li, and L. J. Cimini Jr, "Joint channel-and queue-aware scheduling for multiuser diversity in wireless ofdma networks," *Communications, IEEE Transactions on*, vol. 57, no. 7, pp. 2109–2121, 2009.

[4] P. Tsiaflakis, M. Diehl, and M. Moonen, "Distributed Spectrum Management Algorithms for Multiuser DSL Networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4825–4843, oct 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4547456

[5] M. Jaggi, "Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, S. Dasgupta and D. Mcallester, Eds., vol. 28. JMLR Workshop and Conference Proceedings, 2013, pp. 427–435. [Online]. Available: http://jmlr.csail.mit.edu/proceedings/papers/v28/jaggi13.pdf

[6] P. Seeling and M. Reisslein, "Video transport evaluation with H.264 video traces," *IEEE Communications Surveys and Tutorials, in print*, vol. 14, no. 4, pp. 1142–1165, 2012, Traces available at `trace.eas.asu.edu`.