KU LEUVEN

STADIUS
Center for Dynamical Systems, Signal Processing and Data Analytics

Citation/Reference	Van den Eynde J., Verdyck J., Moonen M., Blondia C. (2017), **A delay-based cross-layer scheduler for adaptive DSL** Proc. of the IEEE International Conference on Communications (ICC), Paris, France, May 2017, pp. 1-6.
Archived version	Author manuscript: the content is identical to the content of the published paper, but without the final typesetting by the publisher
Published version	http://dx.doi.org/10.1109/ICC.2017.7997156
Journal homepage	http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7985734
Author contact	jeroen.verdyck@esat.kuleuven.be *+ 32 (0)16 324723*
Abstract	The quality of experience of many modern network services depends on the delay performance of the underlying communications network. In DSL networks, crosstalk 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 a queue's 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. Through simulations, it is shown that the new scheduler outperforms the state of the art in cross-layer scheduling algorithms.
IR	https://lirias.kuleuven.be/handle/123456789/539743

(article begins on next page)

A delay-based cross-layer scheduler for adaptive DSL

Jeremy Van den Eynde* Jeroen Verdyck† Marc Moonen† and Chris Blondia*
*University of Antwerp
IDLab - Department of Mathematics and Computer Science
{jeremy.vandeneynde,chris.blondia}@uantwerpen.be
†KU Leuven, Department of Electrical Engineering (ESAT)
STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics
{jeroen.verdyck,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, crosstalk 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 a queue's 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. Through simulations, it is shown that the new scheduler outperforms the state of the art in cross-layer scheduling algorithms.**

***Index Terms*—Scheduling, Quality of Service, DSL, Cross-layer**

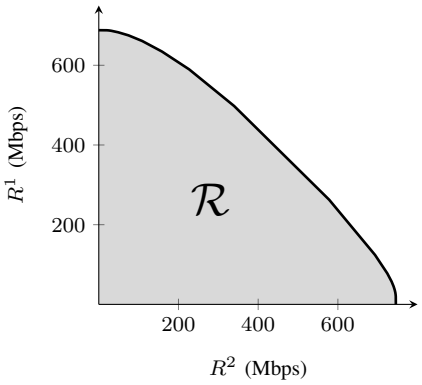


Fig. 1: Rate region of a 2-user G.Fast DSL system

I. Introduction

Maintaining a low delay in communication networks is critical to a large number of 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, such as DSL, competition for bandwidth among users motivates the need for a scheduler that assigns bandwidth appropriately to the users.

In a typical DSL network, multiple twisted pair lines connect the optical network terminal (ONT) to the customer premises equipment of the users. At the ONT the lines are bundled inside a cable binder. The electromagnetic coupling between the different twisted pair lines inside this binder causes inter-user interference or crosstalk, which is the major source of performance degradation in DSL systems.

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) 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, 2) Research Project FWO nr. G.0912.13 'Cross-layer optimization with real-time adaptive dynamic spectrum management for fourth generation broadband access networks ', 3) VLAIO O&O Project nr. HBC.2016.0055 '5GBB Fifth generation broadband access'. The scientific responsibility is assumed by its authors.

The ensemble of approaches that deal with the crosstalk problem is referred to as dynamic spectrum management (DSM) techniques, and can be divided into two major categories: signal coordination and spectrum coordination. This paper considers spectrum coordination, consisting of optimally designing the transmit power spectrum of each user in order to cause minimal crosstalk to other users. Each set of transmit power spectra results in a single tuple of achievable rates. Enumerating all possible rate tuples results in the rate region, an example of which is seen in Fig. 1.

From this figure, it can be seen that no single power allocation simultaneously maximizes the rate for all users. Instead, there are multiple Pareto optimal transmit power spectra which achieve a data rate tuple on the edge of the rate region.

A DSL system typically employs a static operating point. It is chosen in function of a metric such as fairness, maximum sum rate or minimum rate constraints. However, a user's requirements are dynamic, and thus static operating points underutilize the DSL network, hence the need for a dynamical system in which resources are allocated adaptively. Assume time is divided into slots, each slot of which can have a different resource allocation. To the physical layer, no one Pareto optimal resource allocation is inherently better than any other. To the upper layers, however, not all Pareto optimal

resource allocations are equal: those allocations that satisfy the QoS requirements are preferable.

This preference information is communicated to the physical layer through an interface, the utility function. Such function quantifies the usefulness of receiving a data rate. Data rates are then chosen as a solution to the corresponding network utility maximization (NUM) problem:

$$\boldsymbol{\rho}^* = \arg\max_{\boldsymbol{\rho}\in\mathcal{R}} \sum_i u_i(\rho^i) \tag{1}$$

The scheduler provides utility function u_i to the physical layer, which it uses to compute an optimal power allocation along the edge of $\mathcal{R}$. Many existing cross-layer schedulers optimize a simple metric, such as queue length [1], head-of-line delay [2], or average waiting time [3], 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 the system in function of the delay percentile, a measure that closely resembles the true QoS requirements of delay sensitive traffic. The MDV scheduler employs the NUM-DSB algorithm [4], a novel fast physical layer resource allocation algorithm to solve the corresponding NUM problem. The MDV scheduler has excellent performance with respect to throughput and delay violations, especially for high loads.

This paper is structured as follows. Section II describes the system model of both the physical and upper layers. Section III presents a formal description of the MDV scheduler. In Section IV we discuss the simulation and accompanying results. We conclude the paper in Section V.

II. System model

A. 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 spectrum coordination is considered, each of these tones k can be modeled as an interference channel.

$$\boldsymbol{y}_k = H_k\boldsymbol{x}_k + \boldsymbol{z}_k \tag{2}$$

In (2), $\boldsymbol{x}_k = \left[x_k^1,\ldots,x_k^N\right]^T$ 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

$$b_k^n(\boldsymbol{s}_k) = \log_2\left(1+\frac{1}{\Gamma}\frac{|h_k^{n,n}|^2 s_k^n}{\sum_{n\neq m}|h_k^{n,m}|^2 s_k^m + \sigma_k^n}\right), \tag{3}$$

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 consumption of user n, are given as

$$R^n(\boldsymbol{b}^n) = f_s\sum_{k=1}^K b_k^n \qquad P^n(\boldsymbol{s}^n) = \sum_{k=1}^K s_k^n, \tag{4}$$

where f_s is the symbol rate.

The total transmit power of each user is limited to P^{tot}. The transmit spectrum of each user additionally has to satisfy the spectral mask constraint $\boldsymbol{s}^n \leq \boldsymbol{s}^{\text{mask}}$. The set of all possible power loadings of user n can thus be described as

$$\mathcal{S}^n = \left\{\boldsymbol{s}^n\in\mathbb{R}_+^K \mid \ P^n(\boldsymbol{s}^n)\leq P^{\text{tot}} \text{ and } \boldsymbol{s}^n\leq\boldsymbol{s}^{\text{mask}}\right\}. \tag{5}$$

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

$$\mathcal{B} = \boldsymbol{b}(\mathcal{S}) \tag{6}$$

Finally, we define the rate region as

$$\mathcal{R} = \left\{\boldsymbol{r}\in\mathbb{R}_+^N \mid \exists\,\boldsymbol{r}'\in\boldsymbol{R}(\mathcal{B}) : \boldsymbol{r}\leq\boldsymbol{r}'\right\}. \tag{7}$$

As an example, the rate region of a 2-user G.Fast system that employs spectrum coordination is depicted in Fig. 1. Generally, there is no power allocation that simultaneously maximizes the data rate of all users.

B. Upper layer & scheduling

Time is divided in slots of length τ seconds. Each time slot, an operating point for the physical layer has to be chosen. The scheduling occurs in the upper layer, since it has the information that can help deciding the optimal point of operation.

At slot $t\in\mathbb{N}$ the upper layer requests new rates from the physical layer, based on system state $\mathcal{S}(t)$, which can include queue lengths, arrival rates and historical data up to time t. At the start of slot $t+1$, the rates $\boldsymbol{\rho}(t+1)$ are applied in the interval $[t+1,t+2[$. There is thus a delay of 1 slot between the request and application of rates.

Every user $n\in[1,N]$ has ϕ^n traffic streams or flows, each flow i having a delay upper bound $\hat{T}_i^n$, and allowed violation probability ϵ_i^n, or, equivalently, conformance probability $\eta_i^n = 1-\epsilon_i^n$.

Traffic arrives in an infinite buffer. We denote by $a_{i,l}^n(t)$ and $Q_{i,l}^n(t)$ respectively the arrival time and length in bit of user n's l-th queued packet of stream i at the start of slot t. We furthermore define $Q_i^n(t) = \sum_{l=0}^{L_i^n(t)-1} Q_{i,l}^n$, with $L_i^n(t)$ the number of packets in the queue.

QoS requirements are expressed as $P\{D_i^n > \hat{T}_i^n\}\leq\epsilon_i^n$, where D_i^n is the flow's delay distribution, $\hat{T}_i^n$ the delay upper

bound, and ϵ_i^n the flow's violation probability. If $D_i^n > \hat{T}_i^n$, the packet is useless to the application. If more than ϵ_i^n of the packets in a reasonable interval violate this upper bound, the QoE of the user will suffer.

Every stream has a utility function $u_i^n(\rho_i^n, \mathcal{S}(t))$, which quantifies the usefulness to the stream of receiving a rate ρ_i^n, given state $\mathcal{S}(t)$. At the start of every slot t the cross-layer scheduler selects the rate assignment $\boldsymbol{\rho}(t+1) \in \mathcal{R}$ that maximizes the system's performance with respect to a metric:

$$\boldsymbol{\rho}(t+1) = \arg\max_{\boldsymbol{\rho}\in\mathcal{R}} \sum_{n=1}^{N} \sum_{i=1}^{\phi^n} u_i^n(\rho_i^n, \mathcal{S}(t)) \tag{8}$$

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

$$u_i^n(\rho_i^n, \mathcal{S}(t)) = \omega_i^n(\mathcal{S}(t)) \cdot \rho_i^n \tag{9}$$

For example, the Max-Weight scheduler (MW) [5] has $\omega_i^n(\mathcal{S}(t)) = Q_i^n(t)$. The Proportional Fair Queuing scheduler [6] has weights $\omega_i^n(\mathcal{S}(t)) = 1/\bar{\rho}_i^n(t)$, with $\bar{\rho}_i^n(t)$ the average channel capacity up to slot t. For the Max-Delay Utility (MDU) scheduler [3], the authors give $\omega^n(\mathcal{S}(t)) = \frac{|u'^n(\bar{W}^n)|}{\bar{\lambda}^n}$, 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 [7].

III. Minimal Delay Violation Scheduler

Schedulers that want to achieve a delay-optimal result, must incorporate the queue. In [8] the authors show that queue-independent schedulers incur a delay that grows at least linearly with N. In general, schedulers that use the queue size aim to minimize the average delay. This allows one to use Little's law, $\bar{Q} = \bar{\lambda}\bar{W}$, which states that the long-term average number of customers in a stable system $\bar{Q}$ equals the long-term average arrival rate $\bar{\lambda}$ multiplied by the average waiting time $\bar{W}$. Thus, by targeting $\bar{Q}$ the average delay can be manipulated without the need for managing complex information.

Furthermore, [9] shows that for $GI/GI/1$ queues $P\{T > \hat{T}_i^n\} \leq \epsilon$ can be approximated by $P\{T > \hat{T}_i^n\} \approx \exp\{-\frac{\hat{T}_i^n}{\bar{W}}\}$.

The problem is that there can be a significant difference between queue-length and delay. The authors of [10] and [11] mention that keeping the queues small does not guarantee small delays. Another problem is that using only the queue length discards a lot of information contained in the queue, that is important for the delay distribution. For example, imagine a queue with a packet of 10Mbit at the HOL, followed by two smaller 1Mbit packets. For a rate of 10Mbps, the average delay of this queue would be 1.1s. Reversing this queue, does not modify the size. The average delay, however, drops to 0.5s.

The contribution of this paper is the novel Minimal Delay Violation (MDV) scheduler. It aims to minimize the delay violations by looking at the individual packets, rather than the average queue size. It accomplishes this by first estimating the delay metric $\tilde{D}_i^n(t)$ for the coming slot, based on the packets in the queue and observed past delays. Then, depending on the average arrival rate and the proximity of $\tilde{D}_i^n(t)$ to $\hat{T}_i^n$, the flow's weight is defined.

The MDV scheduler uses the utility function $u_i^n(\rho_i^n, \mathcal{S}(t)) = -\frac{\omega_i^n(\mathcal{S}(t))}{\rho_i^n}$, which is increasing, concave and differentiable on $]0, +\infty[$. The following NUM problem is then solved in every time slot:

$$\arg\max_{\boldsymbol{\rho}\in\mathcal{R}} \sum_{n=1}^{N} \sum_{i=1}^{\phi^n} -\frac{\omega_i^n(\mathcal{S}(t))}{\rho_i^n} = \arg\min_{\boldsymbol{\rho}\in\mathcal{R}} \sum_{n=1}^{N} \sum_{i=1}^{\phi^n} \frac{\omega_i^n(\mathcal{S}(t))}{\rho_i^n} \tag{10}$$

in which $\boldsymbol{\omega}$ is defined as

$$\omega_i^n(\mathcal{S}(t)) = \underbrace{\tilde{\lambda}_i^n(t)}_{a)} \cdot \underbrace{\beta(\tilde{D}_i^n - \hat{T}_i^n)}_{b)} \cdot \underbrace{f_i^n\left(\frac{\tilde{D}_i^n}{\hat{T}_i^n}\right)}_{c)} \tag{11}$$

This weight, divided by $\rho_i^n(t+1)$, gives an approximation to the flow's η_i^n-percentile of the delay in the next slot. If we minimize this delay over all users, we minimize the amount of delay violations. Factor $a)$ is an approximation of the future assigned rate, which is based on the arrival rate. A large value implies a possible large delay, and hence should increase the weight. Factor $b)$ increases the weight if the predicted delay approaches the delay upper bound. For an equal $\tilde{D}_i^n$ holds that the larger $\hat{T}_i^n$ becomes, the smaller the weight will be. Factor $c)$ implements the QoS for the flow's traffic class and returns the importance of getting close to the delay upper bound. For example, best effort traffic violating a delay is not as bad as a video frame violating a delay.

We now discuss these three factors in more detail.

a) $\tilde{\lambda}_i^n(t) = \frac{1}{4}\left(\bar{\lambda}_i^n(t) + \sum_{s=t-2}^{t} \rho_i^n(s)\right)$ gives an estimate of the future $\rho_i^n(t+1)$. It is the average of the exponentially averaged arrival rate $\bar{\lambda}_i^n(t)$ and the past assigned data rates. Ideally, $\tilde{\lambda}_i^n(t) = \rho_i^n(t+1)$ as that would allow us to accurately predict the queue evolution.

b) A flow that has a large delay upper bound naturally is more tolerant for delay violations. This factor in (11) will assign more weight if the delay is approaching the delay upper bound, and very few weight if it is far below the absolute delay upper bound. The sigmoid β is defined as

$$\beta(x, \gamma = 1, \mu = 0, \sigma = 1) = \frac{\gamma}{1 + e^{-\frac{x-\mu}{\sigma}}}$$

The approximation of the η_i^n-percentile of the future delay is approximated as

$$\tilde{D}_i^n = \alpha_i^n \frac{\bar{q}_i^n(t)}{\tilde{\lambda}_i^n(t)} + (1 - \alpha_i^n)\bar{d}_i^n(t)$$

It is based on a weighted average of the predicted delay and past observed delays. Here, $\alpha_i^n \in [0, 1]$ indicates the importance of the queue. A small value means that mainly past behavior, i.e. $\bar{d}_i^n(t)$, the η_i^n-percentile of past delays, will influence the weight. This is useful for flows that prefer a long-term average data rate, such as background jobs. A large α_i^n on the other hand will place more importance on the η_i^n-percentile of the predicted delay $\frac{\bar{q}_i^n(t)}{\tilde{\lambda}_i^n(t)}$. Streaming traffic benefits a large α_i^n, as it usually is bursty.

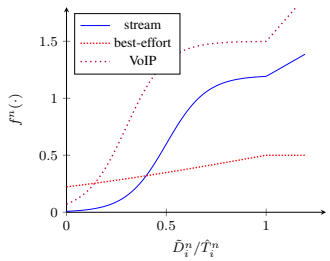


Fig. 2: Multiplier for stream, best-effort and VoIP applications

To calculate $\bar{q}_i^n(t)$ we need to construct the flow's cumulative queue size $\check{Q}_{i,l}^n(t)$, $l \in [1, L_i^n(t)]$:

$$\check{Q}_{i,l}^n(t) = (t - a_{i,0}^n)\rho_i^n(t) + \sum_{m=1}^{l'} Q_{i,m}^n(t) \frac{\tilde{\lambda}_i^n(t)}{\rho_i^n(t)} + \sum_{m=l'+1}^{l} Q_{i,m}^n(t) \tag{12}$$

In this equation, the first term accounts for the head-of-line delay. The second term accounts for the l' packets that will be sent in the interval $[t, t+1]$, and for which we already know the rate. The final term accounts for the packets that depart in the subsequent slots, at a yet unknown rate. Using (12) we can now calculate $\frac{\bar{q}_i^n(t)}{\tilde{\lambda}_i^n(t)}$, and thus also $\tilde{D}_i^n$.

c) The last factor $f_i^n\left(\frac{\tilde{D}_i^n}{\hat{T}_i^n}\right)$ applies the function f_i^n to the normalized predicted delay. This function transforms its argument, the proximity of the predicted delay to $\hat{T}_i^n$, into a weight that reflects its importance with respect to its QoS requirements.

For example, if video v's normalized delay $\frac{\tilde{D}_i^n}{\hat{T}_i^n}$ is small, then v is not important, as its delay requirements will probably not be violated in the coming slot, and hence can have a lower rate assigned. If, on the other hand, the normalized delay approaches 1 then its weight should be much larger, to express delays will probably be violated. For best effort, violating a delay is not as bad, and thus it can be assigned a lower weight.

The following functions have been defined, and can be seen depicted in Fig. 2:

- $f_{stream}(d) = s(1.2, 0.5, 0.1, 1.0, d)$
- $f_{BE}(d) = s(1.0, 1.0, 0.8, 0.0, d)$
- $f_{VoIP}(d) = s(2.5, 0.3, 0.08, 2.0, d)$

with

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

The function s behaves like a regular sigmoid. When the delay reaches its delay upper bound, however, it will go into a linear mode. Best efforts utility must be bounded [10] [12].

A. Complexity

The algorithm is more complex with respect to runtime and memory requirements than MW or MDU, due to the calculation of percentiles. Each flow requires computation of the delay, and the percentiles of the delay and queue. Both computing the percentile [13] and the delays can be performed with an average runtime complexity of $\mathcal{O}(M)$, where M is the set size over which to calculate the percentiles. Hence, the additional system runtime complexity of the scheduler is in the order of $\mathcal{O}(KM)$, where $K = \sum_{n=1}^{N} \phi^n$ is the total number of flows in the system. In practice M can be set to a small number such as 100. Observed delays also have to be tracked, and require storage for KM packets. However, the computational complexity of the cross-layer scheduler is dominated by the resource allocation algorithm (NUM-DSB), justifying the use of a more complex scheduler.

B. Practical constraints

There are some discrepancies between real-world and simulation. In the latter each packet contains the length and delay requirement. However, in real networks this might not be the case without stateful packet inspection. For example, TCP is stream-based and has no notion of packets as delivered by the upper layers: a packet can be split in two TCP segments, or multiple packets might be sent in one TCP segment.

Another difficulty is obtaining the delay of a packet. In simulation, a callback can provide the exact arrival time of complete packet. However, in hardware obtaining the delay requires extra communication. Given that we know the remaining bits in a packet, the delay can be approximated using the data rate.

IV. Performance

A. Simulation setup

We evaluate our scheduler using simulations and compare it to the Max-Weight and Max-Delay Utility schedulers. The simulation consists of two parts, namely the physical layer and networking layer. The former consists of finding the solution to the NUM problem using the NUM-DSB algorithm in MATLAB, while the latter is executed in C++ using the OMNeT++ and INET frameworks. Every $\tau = 50$ms, OMNeT++ computes the utility function parameter $\boldsymbol{\omega}$ and sends it to MATLAB using the *MATLAB Engine API for C*. In the next slot, the rates $\boldsymbol{\rho}$ are read from MATLAB, and applied to the simulated channels as follows. User n's channel capacity $C^n(t)$ is set to $\sum_{i=1}^{\phi^n} \rho_i^n(t)$. Each of its flows i is then fed into a WDRR scheduler with weights $\frac{\rho_i^n(t)}{C^n(t)}$. Packets are sent when the channel is free. At arrival at the next hop, the statistics of reassembled packets are tracked.

The DSL network under consideration is a G.Fast network which connects N users to a distribution point. For $N = 2$, the distances to the ONT are 120m and 110m for respectively users 1 and 2. For $N = 5$, this ranges from 110m for user 1 up to 150m for user 5, increasing with 10m for each consecutive user. Parameter settings for the DSL system are summarized in TABLE I. Spectral mask constraints were not included.

Each of the N users are assigned one or more traffic streams in a scenario. Each scenario is repeated 5 times. We have defined the following kind of traffic streams: video, Poisson with fixed-length packets, VoIP, CBR and SAT. Video includes *Starwars*, *Alice in Wonderland* [14] and a 4k video entitled

TABLE I: G.Fast parameter settings

Parameter	Value	Parameter	Value
$P^{n,\text{tot}}$	4 dBm	K	2047
f_s	48 kHz	Δ_f	51.75 kHz
Γ	12.6 dB		

The Beauty of Taiwan[2]. Each video has its packet lengths multiplied by a constant such that the average rate is closer to its maximal rate, and interesting patterns occur. VoIP traffic generates bursty 64kbps traffic. The SAT type will keep the user's queue backlogged at all times, effectively saturating the line. Each flow has QoS requirements assigned, which can be seen in TABLE II

TABLE II: QoS requirements

Traffic type	QoS class	T	pct
Video	Stream	$\frac{1}{\text{FPS}}$	1%
SAT	BE	1s	50%
Poisson1	Stream	0.1s	10%
Poisson2	BE	0.1s	50%
CBR	Stream	1s	1%
CBR	BE	0.1s	10%

B. Results

The figures Fig. 3 and 4 below show the percentage of excess QoS delay violations. The scenario is shown on the x-axis. It groups the three tested schedulers: MW, MDU and MDV. The large symbols show the average among all traffic flows for a scenario and scheduler. The smaller symbols show the individual flows. The legend shows for every scheduler the average among all scenarios in the plot.

a) Test 1: In this series of tests several scenarios have been evaluated, each of which has a different traffic mix, and a system load well below 1. Each scenario has been run for a simulated time of 30 minutes.

Looking at the delay violations in Fig. 3 one can immediately notice the big discrepancy between MW and the QoS-aware schedulers. This should not be surprising since MW has no notion of what packets are important. Another observation is that for the 5-user scenarios, the QoS schedulers encounter fewer violations, when compared to the MW scheduler. Due to the multiple opportunities, it is now easier for a QoS scheduler to pick a better solution than MW.

If we compare the MDU and MDV schedulers, we can see that for most scenarios MDV performs better than MDU. Also, the variation in delay violations is lower for MDV than MDU, meaning that delay violations are spread more fairly among traffic flows.

Not shown in the plots, but the maximal encountered delay for the two user scenarios is 460s for MW, 11s for MDU and 2.8s for MDV. For five users this is respectively 1768s, 94s and 0.6s. So also here, MDV proves to be a better scheduler in limiting the delay upper bound.

[2]http://tempestvideos.skyfire.com/Sales_Optimization_Demo/beauty_taiwan_4k_final-ed.mp4

Sometimes, bursty flows cause MDV to perform a little worse than MDU. E.g. in scenarios 9 and 10, the bursty Taiwan video has a little more delay violations for MDV. It is not yet exactly clear what causes this, as for example in scenarios 11 and 16, which also consists of the bursty Taiwan video, MDV performs much better than MDU. For scenarios 2 and 14, both of which consist of all Starwars flows, MDV also performs a little worse, as it chooses to keep delay violations more closely together. MDU assigns in these cases more bandwidth to users who have a larger maximal capacity, and hence also lower their delay violations.

TABLE III shows the throughputs of the various scenarios that have SAT flows. One can see that on average MDV has a higher throughput than both MW and MDU schedulers. Scenarios 1 and 12 consist of only SAT traffic. In both, MDV outperforms MDU. For scenarios 9, 17 and 21, which are mixes comprising a lot of video and SAT flows we can see that MDV performs much better than the other schedulers. Since the scheduler knows more accurately the impending violations, it can assign more intelligent bandwidth to the SAT flows.

TABLE III: Average troughput (Mbps)

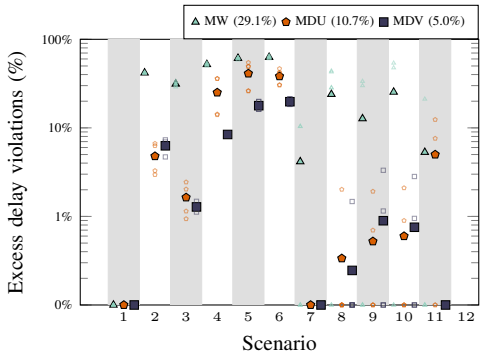
	MW	MDU	MDV
Scenario 1	171	170	172
Scenario 9	127	136	138
Scenario 11	151	78	151
Scenario 12	94	87	89
Scenario 17	15	25	29
Scenario 19	59	58	58
Scenario 21	69	75	80
Average	98	90	102

b) Test 2: For these tests we look at the system's behavior in case traffic arrives at a rate close to a Pareto optimal point. We pick the rate vectors $\boldsymbol{\rho_1} = \arg\max_{\boldsymbol{\rho}\in\mathcal{R}} \sum_{n=1}^{N} 1\cdot\rho^n$ and $\boldsymbol{\rho_2} = \arg\min_{\boldsymbol{\rho}\in\mathcal{R}} \sum_{n=1}^{N} \frac{1}{\rho^n}$. In the first and second scenario, user n sends respectively CBR and Poisson (Psn1) traffic at $0.99\rho^n$. For the third scenario (Psn2), she sends at ρ^n. Fig. 4 below shows the results for $\boldsymbol{\rho_1}$ and $\boldsymbol{\rho_2}$ for two and five users.

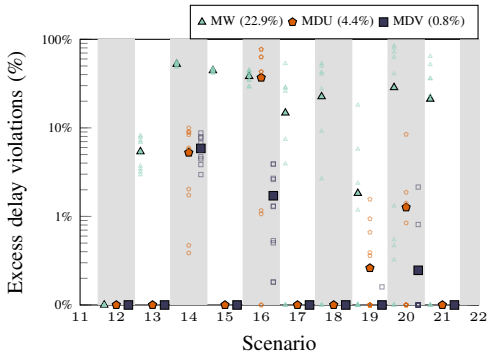
For the two user scenarios (Fig. 4a), we can observe that MDV does not violate any QoS requirements for any of the scenarios. The MW scheduler consistently violates delays. The different traffic flows however all have similar violation probabilities. The MDU scheduler performs well for $\boldsymbol{\rho_1}$, however for $\boldsymbol{\rho_2}$ one traffic flow suffers considerable delay violations, in favor of the other flow.

For five users (Fig. 4b), MDV also outperforms the other schedulers considerably. Here, however, MW performs much better for $\boldsymbol{\rho_1}$ than MDU, which consistently has a very high violation probability. It has to be noted that also here some flows violate delays with high probability, while others never suffer large delays.

In high load situations, it is clear that MDV performs substantially better than the MW and MDU schedulers. This is because MDV has an accurate idea of the short-term requirements for all flows. As such, it can temporarily hold

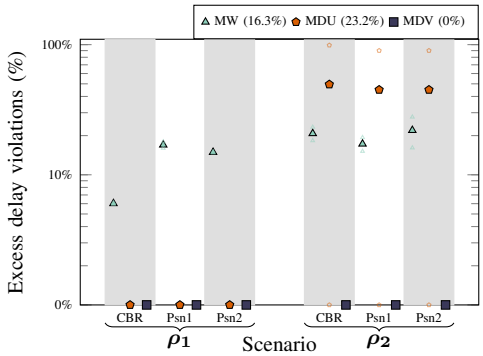


(a) Test 1 - delay violations ($N = 2$)

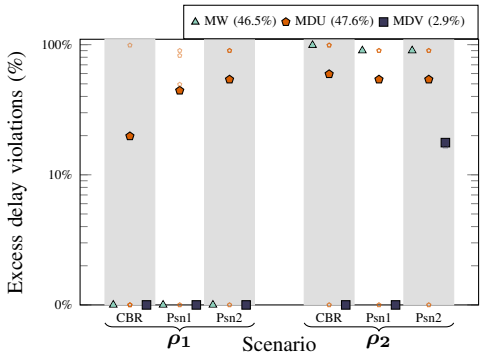


(b) Test 1 - delay violations ($N = 5$)

Fig. 3: Test 1 delay violations



(a) Test 2 - Delay violations ($N = 2$)



(b) Test 2 - Delay violations ($N = 5$)

Fig. 4: Test 2 delay violations

back some flows in a controlled way, giving other flows a higher capacity.

V. Conclusion

In this paper we introduced a DSL system, and argued the need for a dynamic cross-layer scheduler. A network utility maximization (NUM) problem is presented as a clear interface between the upper and lower layers. The weights to this NUM problem are calculated by predicting the delay percentile using the packets in queue, and applying a QoS utility function. Simulations have shown that this approach offers good performance with respect to delay violations and throughput, especially in systems where the load is high.

References

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE transactions on automatic control*, vol. 37, no. 12, pp. 1936–1948, 1992.

[2] C. Wang and R. D. Murch, "Adaptive cross-layer resource allocation for downlink multi-user mimo wireless system," in *2005 IEEE 61st Vehicular Technology Conference*, vol. 3. IEEE, 2005, pp. 1628–1632.

[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] J. Verdyck, C. Blondia, and M. Moonen, "A Low-Complexity Algorithm for Utility Based Spectrum Coordination in DSL Systems," in *Proc. of 42nd International Conference on Acoustics, Speech and Signal Processing (accepted)*, 2017.

[5] L. Georgiadis, M. J. Neely, L. Tassiulas *et al.*, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends® in Networking*, vol. 1, no. 1, pp. 1–144, 2006.

[6] A. Jalali, R. Padovani, and R. Pankaj, "Data throughput of cdma-hdr a high efficiency-high data rate personal communication wireless system," in *Vehicular technology conference proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, vol. 3. IEEE, 2000, pp. 1854–1858.

[7] 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, 2008.

[8] M. J. Neely, "Order optimal delay for opportunistic scheduling in multi-user wireless uplinks and downlinks," *IEEE/ACM Transactions on Networking (TON)*, vol. 16, no. 5, pp. 1188–1199, 2008.

[9] S. Asmussen, *Applied probability and queues*. Springer Science & Business Media, 2008, vol. 51.

[10] A. Sharifian, "Utility-based packet scheduling and resource allocation algorithms with heterogeneous trac for wireless ofdma networks," Ph.D. dissertation, Carleton University Ottawa, 2014.

[11] M. Sharma and X. Lin, "Ofdm downlink scheduling for delay-optimality: Many-channel many-source asymptotics with general arrival processes," in *Information Theory and Applications Workshop (ITA), 2011*. IEEE, 2011, pp. 1–10.

[12] G. Song, "Cross-layer resource allocation and scheduling in wireless multicarrier networks," Ph.D. dissertation, Citeseer, 2005.

[13] B. Vallée, J. Clément, J. A. Fill, and P. Flajolet, "The number of symbol comparisons in quicksort and quickselect," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2009, pp. 750–763.

[14] 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`.