KU LEUVEN

STADIUS
Center for Dynamical Systems, Signal Processing and Data Analytics

Citation/Reference	Van den Eynde J., Verdyck J., Blondia C., Moonen M. (2021), **Minimal Delay Violation-Based Cross-Layer Scheduler and Resource Allocation for DSL Networks** IEEE Access, vol. 9, May 2021, 75905-75922
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	10.1109/ACCESS.2021.3081793
Journal homepage	https://ieeeaccess.ieee.org/
Author contact	Jeroen.verdyck@esat.kuleuven.be *+32 (0)16 324723*
Abstract	The quality of service of many modern communication systems depends on the delay performance of the underlying network. In digital subscriber line (DSL) networks, for example, crosstalk introduces competition for data rate among users, which influences the delay distribution. In such a competitive environment, delay performance is largely determined by the manner in which resources are dynamically allocated to the different users. A common approach to this allocation problem is through a cross-layer scheduler. Such a scheduler violates the OSI model by allowing communication between different layers, in order to steer the physical layer towards operating points that maximize some upper layer performance metric. In this paper, we present a new cross-layer scheduler and resource allocation algorithm in the context of DSL networks, referred to as the minimal delay violation (MDV) scheduler, which aims to minimize the number of delay violations and achieve a high

	throughput. Rather than solving a linear network utility maximization problem, as most other schedulers from literature, we consider a problem that is reciprocal with the service rate, allowing the scheduler to allocate the data rates at a finer level, while still maintaining good performance. Through simulations, we show that the MDV scheduler performs better than cross-layer scheduling algorithms from literature with respect to packet loss ratio, delay and throughput performance for various scenarios, and often operates closely to an ideal scheduler.
IR	https://limo.libis.be/primo-explore/fulldisplay?docid=LIRIAS3456534&context=L&vid=Lirias&search_scope=Lirias&tab=default_tab&lang=en_US

(article begins on next page)

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier DOI

Minimal delay violation-based cross-layer scheduler and resource allocation for DSL networks

JEREMY VAN DEN EYNDE[1], JEROEN VERDYCK[2], CHRIS BLONDIA[1], MARC MOONEN[2]
[1]University of Antwerp — imec, IDLab, Department of Mathematics and Computer Science
[2]KU Leuven, Department of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing, and Data Analytics

Part of this research work was carried out in the frame of Research Project FWO nr. G.0B18.18N 'Real-time adaptive cross-layer dynamic spectrum management for fifth generation broadband copper access networks'. The scientific responsibility is assumed by its authors.

ABSTRACT The quality of service of many modern communication systems depends on the delay performance of the underlying network. In digital subscriber line (DSL) networks, for example, crosstalk introduces competition for data rate among users, which influences the delay distribution. In such a competitive environment, delay performance is largely determined by the manner in which resources are dynamically allocated to the different users. A common approach to this allocation problem is through a cross-layer scheduler. Such a scheduler violates the OSI model by allowing communication between different layers, in order to steer the physical layer towards operating points that maximize some upper layer performance metric.

In this paper, we present a new cross-layer scheduler and resource allocation algorithm in the context of DSL networks, referred to as the minimal delay violation (MDV) scheduler, which aims to minimize the number of delay violations and achieve a high throughput. Rather than solving a linear network utility maximization problem, as most other schedulers from literature, we consider a problem that is reciprocal with the service rate, allowing the scheduler to allocate the data rates at a finer level, while still maintaining good performance.

Through simulations, we show that the MDV scheduler performs better than cross-layer scheduling algorithms from literature with respect to packet loss ratio, delay and throughput performance for various scenarios, and often operates closely to an *ideal* scheduler.

INDEX TERMS Cross-layer scheduler, Resource allocation, DSL, Quality of service

I. INTRODUCTION

MAINTAINING a low delay in a communication network is critical to a wide variety of applications such as video conferencing, voice over IP (VoIP), gaming, and live-streaming. If many delay violations occur, quality of experience (QoE) suffers considerably for these applications, and the allocated resources are wasted. The QoE is usually expressed through quality of service (QoS) rules that quantify the desired metrics. Scheduling plays an important role in provisioning QoS, as it chooses how to allocate resources to different applications.

The resources that are available depend on the underlying physical layer. In a digital subscriber line (DSL) network multiple twisted pair lines connect the distribution point unit (DPU) to the customer premises equipment of the users. These lines are bundled inside a cable binder, where the electromagnetic coupling between the different twisted pair lines causes inter-user interference or crosstalk, which is then the major source of competition for data rate among users. Figure 1 shows an example rate region $\mathcal{R}$, the set of all rate vectors that can be provided by the physical layer, for two users. Due to crosstalk there is no allocation that maximizes the service rate for both users at the same time.

The dynamic nature of applications and the physical layer create a competition for data rate. At the upper layer the requirements for the users fluctuate over time, as the serviced applications and their demands change. At the physical layer, meanwhile, there are multiple Pareto-optimal data rate points

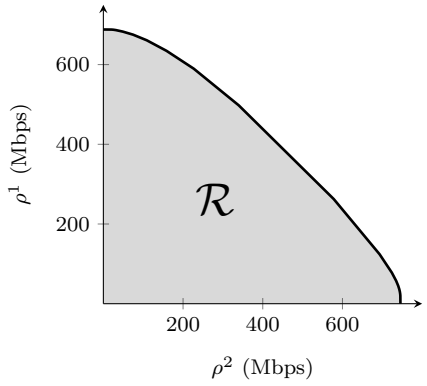


FIGURE 1. A rate region for a two user system

from which to choose (see for example Figure 1). As is dictated by the Open Systems Interconnection (OSI) model the upper and lower layers operate independently of each other. But this can lead to inefficient network usage and degradation of the network performance. For example, if one user is watching a live-stream, and another user is browsing the web and downloading e-mails, then assigning both users a fixed service rate will lead to inefficient usage of the available resources. Additionally, if the live-streaming user experiences a temporary peak in traffic arrivals, it is impossible to indicate to the physical layer that it requires more service, resulting in a reduction of the user's QoS and the performance of the network.

Dynamically adjusting the service rates offered to users can resolve these problems. In the example given above, the upper layers might instruct the physical layer to share service based on the relative queue size of both applications. The physical layer then decides on how to allocate resources, based on the upper layer's preference information, the queue sizes in this case. An approach to share this information between the physical and upper layers is through a utility function. Such function quantifies for each user the usefulness of receiving a certain service rate. Service rates are then set by solving the corresponding network utility maximization (NUM) problem:

$$\boldsymbol{R}^* = \arg\max_{\boldsymbol{R}\in\mathcal{R}} \sum_n u^n(R^n) \tag{1}$$

where $\boldsymbol{R} = [R^1, \ldots, R^n]^T$. A cross-layer scheduler then translates the upper layer preference information into utility functions u^n that express the usefulness to user n of receiving a service rate R^n. There are many available cross-layer schedulers that focus on wireless networks, and optimize in function of different metrics such as delay [1] or power usage [2], or joint optimizing of several metrics [3]. These schedulers assume that the solution to the (1) can be calculated and applied immediately. However, in our DSL setting the solution can take an order of magnitude more time to solve, introducing a delay between obtaining the metrics and application of the new service rates. This can lead to a degradation in performance.

The contribution of this paper is the minimal delay violation (MDV) scheduler, targeted towards DSL G.fast communication networks, that is proven to be throughput optimal for convex rate regions. The MDV scheduler aims to minimize the number of delay violations while also offering a good throughput to best-effort flows. We show through simulations that the MDV scheduler has excellent performance with respect to throughput, delay violations and multiplexing capabilities. This work extends [4] with a stability analysis of the scheduler, and notes on the discretization of the rate region and the intra-user scheduler. It is now being applied to a DSL G.fast system with DSL grouped vectoring. We have simulated additional scenarios (focusing on e.g. multiplexing and heavy-tailed traffic). The simulation results now feature to different intra-user schedulers, and includes more and more recent schedulers from literature.

This paper is structured as follows. We discuss related work in Section II. In Section III we describe the system model. In Section IV we present a formal description of the MDV scheduler, and an analysis of the scheduler together with a discussion of the stability. In Section V we discuss sampling the DSL G.fast physical layer such that we can assess the performance of the schedulers fairly. In Section VI we evaluate the MDV scheduler and compare the performance with other cross-layer schedulers using simulations. We conclude the paper in Section VII.

II. RELATED WORK

Many of the schedulers listed here are used in wireless networks, but due to the general nature of the NUM problem (1) which optimizes weights over a rate region, it can also be applied to the DSL setting. There is a family of cross-layer schedulers where the utility function is linear with R^n. Therefore, (1) is often simplified to

$$\boldsymbol{R}^*(t+1) = \arg\max_{\boldsymbol{R}\in\mathcal{R}} \sum_n \omega^n(t) R^n \tag{2}$$

where t is the time slot and ω^n is called the weight of user n. The seminal work of the max-weight (MW) scheduler [5] introduces one of the first opportunistic schedulers. The MW scheduler has $\omega^n(t) = Q^n(t)$, where $Q^n(t)$ is the length of the queue at time t. It performs very well with respect to throughput, but it lacks any notion of QoS. Many subsequently proposed schedulers focus on optimizing a single QoS metric. For example, the delay-based max-weight (DMW) scheduler of [6] has $\omega^n(t) = \Gamma^n(t)$, where $\Gamma^n(t)$ is called the head-of-line (HOL), the waiting time of the packet at the front of user n's queue. Such an approach is less apt to deal with bursty traffic, as batch arrivals will result in a low initial HOL but at the same time a large queue, causing larger delays for subsequent packets. For the maximal delay utility (MDU) scheduler [7], $\omega^n(t) = \frac{|u'(\overline{w}^n)|}{\overline{\lambda}^n}$, where u is a traffic-class based function, $\overline{w}^n$ the average waiting time, and $\overline{\lambda}^n$ the average arrival rate. The average waiting time is approximated using Little's law. However, as the mean delay is used applications sensitive to real-time requirements can

suffer. The EXP/PF [8] scheduler differentiates between real-time and best-effort applications. The real-time application weights takes the average HOL of all users into account, while the best-effort applications receive only service when the HOL of all real-time applications are below the delay threshold.

In [9] another approach is presented that uses utility functions, where applications with the tightest deadlines receive higher priority. Their approach does not exploit the multi-user diversity, and results in an increased number of packets missing their deadlines. In [10] a joint power allocation and transmit scheduling method is introduced for orthogonal frequency-division multiplexing (OFDM) wireless networks with mixed real-time and non-real time users. It aims to reduce the delay variance and tries to satisfy the delay requirements of the real-time users, though at the expense of throughput. In [11] a scheduler framework is introduced for real- and non-real-time applications in orthogonal frequency-division multiple access (OFDMA) wireless networks. Their framework can approximate other common schedulers such as earliest deadline first (EDF) and modified largest weighted delay first (M-LWDF). Some approaches incorporate a neural network (NN). For example, in [12] the "AdaptSch" framework is presented, built on two NN blocks, the first one of which predicts network traffic, while the second block predicts the performance for a set of predefined schedulers and chooses the best one. This can improve the delay performance, but at the cost of overall throughput. In [13] another allocation algorithm is presented, but the tuning parameters require a priori knowledge of the applications, such as the required throughput. In [2] a cross-layer algorithm is developed in the context of ultra reliable and low-latency communications. It aims to minimize the number of packets that exceed the delay bound for a given power constraint. Rather than the queue size, it bases its decision on the delay of the individual packets.

In [1] a cross-layer scheduling algorithm is developed that minimizes the delay in vehicular networks. It adds a parameter V that allows for a trade-off between throughput and latency.

The authors of [3] introduce a probabilistic cross-layer scheduler. Each packet is transmitted with a certain probability that is determined by the queue length. They aim to minimize the average queuing delay under average power constraints. The model only allows for sending at most one packet per slot. This approach is not suitable for use in our DSL system as the slots are much larger compared to the wireless channel setting in the paper.

Other cross-layer schedulers such as FLS [14] and Opt.-Fair [15] implement a mechanism similar to priority queue (PQ) in every time slot in order to achieve a low packet loss ratio (PLR) and high throughput. They first select and serve some of the real-time flows, and left-over resource blocks (RBs) are then assigned to best-effort flows.

In [16] different scheduling strategies are discussed together with a survey of the schedulers. A taxonomy of cross-layer schedulers can be found in [17].

Some of the schedulers listed above are also used in the simulations in Section VI and are listed in Table 3 on Page 11 together with the expression used by each scheduler to calculate the user weights ω in the weighted sum rate maximization problem (1).

III. SYSTEM MODEL

In this section we give a high level overview of the system and describe the common symbols used throughout the paper.

In the system we consider, time is divided in slots of length $\tau = 50$ ms. There are N users, where each user $n \in [1, N]$ has ϕ^n flows (or equivalently applications). Flows are indexed by subscript i. The total number of flows in the system is $\phi = \sum_{n=1}^{N} \phi^n$.

In each slot an operating point for the physical layer has to be chosen. The physical layer assigns to every user n a service rate $R^n(t) \in \left[0, \hat{R}^n\right]$ where $\hat{R}^n$ is the maximal service rate possible for user n. Furthermore, $R^n \in \mathcal{R}$, where the rate region $\mathcal{R}$ is the set of all Pareto-optimal rate vectors that the physical layer can accommodate. The capacity region is defined as $\mathcal{C} = \text{conv} \cup_{\boldsymbol{r} \in \mathcal{R}} \left((\{\boldsymbol{r}\} - \mathbb{R}_+^N) \cap \mathbb{R}_+^N \right)$, where conv$\mathcal{A}$ denotes the convex hull of the set $\mathcal{A}$.

The upper layer determines at the start of slot t the system state $\mathscr{S}(t)$, which can include historical data up to time t, such as arrival rates, or immediate data such as queue lengths. Based on $\mathscr{S}(t)$ the scheduler then constructs the utility functions $u_i^n(\cdot)$ for each flow. These utility functions are then passed to the processing unit of the physical layer, where NUM problem (1) is solved to determine the optimal operating point. At the start of slot $t+1$, the reply of the physical layer, i.e. service rates $\boldsymbol{R}(t+1)$, is applied. These service rates are in effect in the interval $[t+1, t+2[$. There is thus a delay of one slot between the request and application of service rates.

Each flow $i \in [1, \ldots, \phi^n]$ of user n has its QoS defined by $P\{D_i^n(t) > \hat{T}_i^n\} \leq \varepsilon_i^n$, where $D_i^n(t)$ are the delays of the flow's packets up to slot t, $\hat{T}_i^n$ a delay upper bound, and ε_i^n the allowed violation probability. Traffic arrives in a buffer large enough to hold all packets. However, if a packet's delay exceeds $\hat{T}_i^n$, the packet is useless to the flow, and will be dropped. If $P\{D_i^n(t) > \hat{T}_i^n\} > \varepsilon_i^n$ in a reasonable interval, the QoE of the user will suffer.

The number of arrivals and departures in bits for flow i and user n during the interval $[t, t+1[$ are denoted by $A_i^n(t)$ and $E_i^n(t)$. The number of arrivals and departures in packets in an interval $[s, t]$ are written as $A_{p,i}^n(s,t)$ and $E_{p,i}^n(s,t)$ The queue size (in bits) at the start of slot t is denoted by $Q_i^n(t)$. The HOL is the time spent in the system by the packet at the head of the queue, and is denoted by $\Gamma_i^n(t)$.

Every flow has a utility function $u_i^n(R_i^n, \mathscr{S}_i^n(t))$, which quantifies the usefulness to the flow of receiving a service rate R_i^n, given state $\mathscr{S}_i^n(t)$[1]. At the start of slot t the cross-

[1] We will often omit $\mathscr{S}_i^n(t)$ if is clear from context

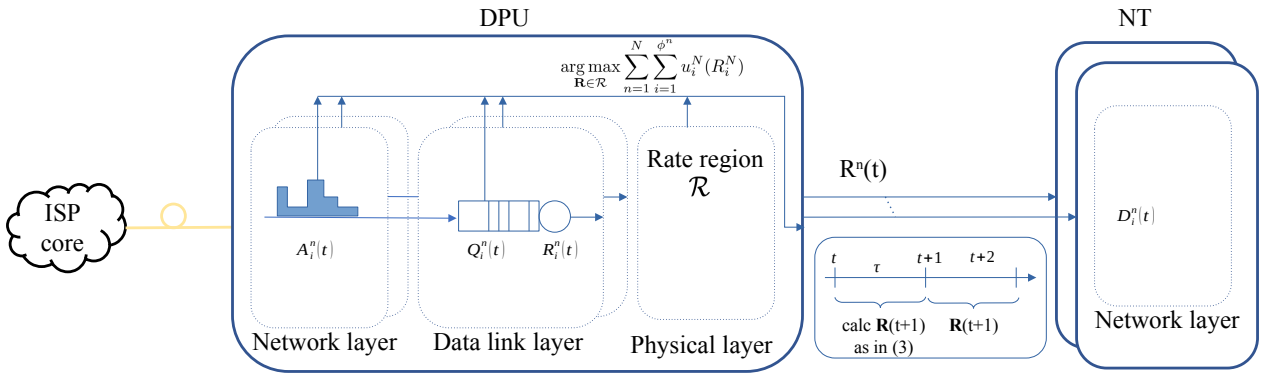


FIGURE 2. The system model. The ISP core network is connected to the distribution point unit (DPU) through an optical fiber cable. The DPU is connected to N network terminations (NTs). In the DPU layers 1 to 3 contribute here to solving the NUM problem. The physical and data link layers at the NT are omitted.

layer scheduler selects the rate assignment $\boldsymbol{R}(t+1) \in \mathcal{R}$ that maximizes the system's performance, i.e.:

$$\boldsymbol{R}(t+1) = \arg\max_{\boldsymbol{R}\in\mathcal{R}} \sum_{n=1}^{N} \sum_{i=1}^{\phi^n} u_i^n(R_i^n, \mathscr{S}_i^n(t)). \quad (3)$$

A large family of scheduling algorithms is linear in $\boldsymbol{R}$ [7], [18]–[21], i.e.

$$u(R, \mathscr{S}(t)) = R \cdot \omega(\mathscr{S}(t)). \quad (4)$$

For the MDV scheduler we present in this paper, we have

$$u(R, \mathscr{S}(t)) = -\frac{\omega(\mathscr{S}(t))}{R+\zeta} \quad (5)$$

where ζ is a small constant to avoid division by 0. We refer to (4) and (5) as MW-style and MD-style schedulers respectively. The reasoning for using an MD-style scheduler is explained in the introduction of the next section.

IV. THE MDV SCHEDULER

In this section, we introduce the MDV scheduler and its equations, and discuss some properties.

Schedulers from literature usually calculate ω based on immediate QoS-related metrics like the queue state and the HOL delay (and possibly other metrics like power). As the resources for these schedulers are typically assigned immediately every 1 ms, metrics like the queue and HOL delay can provide accurate guidance for the duration of the next 1 ms-sized slot. However, in our DSL system slot sizes are an order of magnitude larger, and, together with the fact resources are only allocated one slot later, the queue and HOL might be out-of-date the moment that the resources are assigned. For example, assume $Q(t) = 0$, and at $t + 0.01$ packets arrive, then if the weight depends on the queue (e.g. [1], [22]) or HOL (e.g. [23], [24]), it will get assigned a low or possibly even zero service rate for the coming 50 ms slot.

Furthermore, most cross-layer schedulers are developed in the context of wireless networks, where the achievable data rate can change drastically over short time spans. Hence, these use the utility function (4) which favors servicing users experiencing a better signal-to-noise ratio (SNR), at the cost of data rate fairness. In the DSL setting the rate region is static, hence opportunistic scheduling is not as important.

Therefore, in the MDV scheduler we solve the first problem, the inherent delay of one slot between calculating ω and application of the service rates, by making use of the expected arrival rate and the number of recently dropped packets, in addition to the queue size, to determine a flow's weight. The arrival rate and PLR are not as volatile and hence can steer the weights better over multiple slots. This ensures that a flow will receive sufficient service rate, even though it is not backlogged. The inclusion of the queue ensures that sudden bursts of traffic will increase the service rate. The second issue, the opportunistic character of cross-layer schedulers, is resolved by using a utility function of the form (5), which is more suitable for fair sharing [25].

The weight for the MDV scheduler is shown in (6), and is composed of three components. The factor $\tilde{\lambda}_i^n(t+1)$ is an estimate of the number of bits that will arrive in slot $t+1$. The function $c_i^n(\cdot)$ is dependent on the traffic class (e.g. streaming, or best-effort), and operates on its argument which acts as a measure for how close a flow is to violating its QoS delay requirement.

In Section IV-A we will first discuss the components that comprise the scheduler. Then in Section IV-B we highlight a difference with MW-style schedulers. In Section IV-C we discuss the stability of the MDV scheduler. In Section IV-D we add some important notes on the discretization of the rate region.

A. THE COMPONENTS

In this subsection we will have a detailed look at the components that make up the MDV scheduler, as described in (6).

1) Factor (a)

Factor (a), $\tilde{\lambda}_i^n(t+1)$, constitutes an estimate of the required service rate to support the flow in slot $t+1$, the slot that we

$$\omega_i^n(\mathscr{S}_i^n(t)) = \underbrace{\tilde{\lambda}_i^n(t+1)}_{\text{(a)}} \cdot \underbrace{c_i^n}_{\text{(c)}} \left(\underbrace{\delta_c \frac{Q_i^n(t)}{(R_i^n+\zeta)\cdot \hat{T}_i^n} + (1-\delta_c)\frac{P\{D_i^n(t) < \hat{T}_i^n\}}{\varepsilon_i^n}}_{\text{(b)}} \right) \tag{6}$$

are now finding the suitable weights for. We calculate it as $\tilde{\lambda}_i^n(t+1) = \frac{\tilde{A}_i^n(t+1)}{\tau}$, where $\tilde{A}_i^n(t+1)$ is a prediction of the number of bits that will arrive during the slot $t+1$.[2] We use the low-complexity normalized least mean square (NLMS) predictor [26]. Each slot the state of the algorithm is updated with the vector $\boldsymbol{a}_i^n(t) = [A_i^n(t-1), \ldots, A_i^n(t-p)]^T$, i.e. the arrivals in the past p slots, which results in the prediction $\tilde{A}_i^n(t+1)$. In the simulations we used $p = 20$.

2) Argument (b)

The MDV scheduler aims to ensure that less than $100\varepsilon_i^n$ percent of the packets experience a delay more than $\hat{T}_i^n$. We use a metric b_i^n here, given by

$$b_i^n = \delta_c \underbrace{\frac{Q_i^n(t)}{(R_i^n(t)+\zeta)\cdot \hat{T}_i^n}}_{\text{(b1)}} + (1-\delta_c) \underbrace{\frac{P\{D_i^n(t) < \hat{T}_i^n\}}{\varepsilon_i^n}}_{\text{(b2)}}. \tag{7}$$

It expresses the closeness of a flow to violating its QoS delay requirement $P\{D_i^n(t) < \hat{T}_i^n\} \leq \varepsilon_i^n$, which is then used as the argument of c_i^n in (6). The closer b_i^n is to 1, the more likely there are delay violations, and the more service rate should be given to this flow in order to reduce the delay violations. When b_i^n exceeds 1 there will be violations in the next slot.

We estimate the b_i^n metric based on the weighted average of the current queue size (b1) and the past delay violations (b2), which approximate the predicted delay of the most recently arrived packet and the delay percentile respectively. We now look at (b1), (b2) and δ_c in more detail.

(b1): The factor $\frac{Q_i^n(t)}{R_i^n(t)+\zeta}$ in (7) can be seen as an approximation of the delay of the most recently arrived packet, if the service rate were to be kept at $R_i^n(t)$. Thus $(b1) = \frac{Q_i^n(t)}{(R_i^n(t)+\zeta)\hat{T}_i^n}$ indicates the proximity of the delay of the queue's last packet to the delay upper bound $\hat{T}_i^n$, given a constant service rate $R_i^n(t)$. A value larger than 1 means that the packet will violate the QoS delay requirement.

In [27] the authors show that queue-independent schedulers incur a delay that grows at least linearly with the number of flows. Hence, the queue should be incorporated in order to achieve good performance.

(b2): The term (b2) represents an estimate of the number of delay violations so far, relative to the QoS delay bound. We calculate the delay distribution $P\{D_i^n(t) < \hat{T}_i^n\}$ as $1 - E_{p,i}^n/A_{p,i}^n$, where $E_{p,i}^n$ is the number of packets that have been sent with a HOL less than $\hat{T}_i^n$ and $A_{p,i}^n$ are the total number of packets for the flow. These can be easily tracked using a simple counter.

δ_c: The parameter δ_c is class-dependent and shifts the focus to future delay violations (a large δ_c) or to past delay violations (a small δ_c). For the streaming traffic class, we use $\delta_{stream} = 0.5$, while best-effort traffic class has $\delta_{BE} = 0.2$. By choosing a smaller $\delta_{BE} < \delta_{stream}$, we shift the weight from the volatile queue metric to the less volatile delay metric for the best-effort traffic class. As best-effort traffic is less sensitive to variation in delay than streaming traffic, it can also cope better with temporarily larger queues. These values were chosen empirically through simulations.

3) Function (c)

The traffic class-dependent function c is applied to the argument (b) and indicates the elasticity of the flow. Following two classes are defined:

- $c_{stream}(x) = \beta(x, 1, 0.6, 0.2, 1.0)$
- $c_{BE}(x) = \beta(x, 0.15, 0.4, 0.2, 0.1)$

with

$$\beta(x,\gamma,\mu,\sigma,\rho) = \begin{cases} \beta_2(x,\gamma,\mu,\sigma) & \text{if } x \leq 1 \\ \beta_2(1,\gamma,\mu,\sigma) + (x-1)\rho & \text{if } x > 1 \end{cases} \tag{8}$$

where β_2 is the sigmoid function

$$\beta_2(x,\gamma,\mu,\sigma) = \frac{\gamma}{1+\exp(-(x-\mu)/\sigma)}$$

The functions c_{stream} and c_{BE} behave like a regular sigmoid when $b_i^n < 1$. When $b_i^n \geq 1$, however, c_{stream} and c_{BE} will switch to linear mode. The slope in this case is much larger for $c_{stream}(b_i^n)$ than for $c_{BE}(b_i^n)$. If the system is overloaded, b_i^n quickly becomes more than 1 for all flows as the queue sizes (and subsequently delays) will increase. But as the streaming class's weight increases faster, the streaming traffic class flows will be prioritized over best-effort traffic flows.

The values for the c_{stream} and c_{BE} functions are chosen empirically through simulations. It is clear that when a flow from the streaming traffic class is far from violating its requirements, its weight is low, thus giving more weight to other flows. Comparing the traffic class functions c_{stream} and c_{BE} in Figure 3, we can see that for small b_i^n the function c_{BE} is relatively large. This results in the best-effort traffic class receiving a larger share. However, as the system load increases, and thus also b_i^n increases, c_{stream} will quickly receive a larger weight and hence a larger service rate.

[2]For stability reasons (see Section IV-C), we assume $\tilde{A}_i^n(t) > 0$, and use $\tilde{A}_i^n(t)$ if $\tilde{A}_i^n(t+1) \leq 0$.

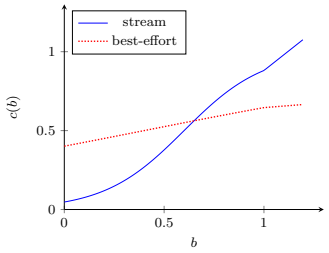


FIGURE 3. $c_{stream}(b)$ and $c_{BE}(b)$

B. INTRA-USER SCHEDULING

In the previous subsection, we discussed the MDV scheduler and how it calculates the weights ω_i^n for the corresponding NUM problem. In this section, we look at the scheduling of flows for a single user, contrasting the MW-style and MD-style schedulers.

The channel of a single user can be fed with the output of a traditional regular packet scheduler (e.g. [28]–[30]), operating on the aggregate of the user's flows. In some circumstances however, it may be useful to consider each flow being allocated a separate channel. Consider for example a best-effort flow. Generally, it can deal with packet loss, and thus such a flow might request a higher service rate at the cost of a higher bit-error rate. Likewise, VoIP calls require reliable transfer, and as such can request for a low bit-error rate. For example [31] discusses a DSL scenario in which each flow can have different properties.

In such case, it is interesting to use the scheduler to also allocate the resources for the intra-user flows. Solving the NUM problem for all users can then conceptually be split into two steps. First a service point $\boldsymbol{R}$ in the rate region is picked. Then, for each user n the service rate R^n must be divided over user n's flows. This intra-user rate region can be considered a simplex rate region (see Figure 4 for an example of a three flow 2-simplex). In such a scenario, increasing one flow's rate by δ will decrease the sum of the other flows' rates by exactly δ.

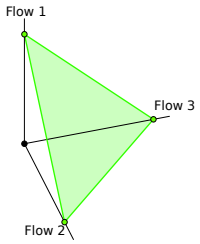


FIGURE 4. Intra-user rate region for a user with three flows

It is in this setting that our cross-layer scheduler excels. Consider a user n receiving a rate R^n, to be distributed over ϕ^n flows. This rate region is the $\phi^n - 1$-simplex. If we use an MW-style scheduler for intra-user rates $\boldsymbol{r}^n = [r_1^n, \ldots, r_{\phi^n}^n]$ then the NUM problem can be written as

$$\underset{[r_1^n,\ldots,r_{\phi^n}^n]^T \succcurlyeq \mathbf{0}}{\arg\max} \sum_{i=1}^{\phi^n} \omega_i^n r_i^n, \text{ subject to } \sum_{i=1}^{\phi^n} r_i^n \leq R^n. \quad (9)$$

Here, (9) results in an assignment that gives only a non-zero rate to the flow that has the largest weight ω_i^n.

If we now consider an MD-style scheduler the corresponding NUM problem is formulated as:

$$\underset{[r_1^n,\ldots,r_{\phi^n}^n]^T \succcurlyeq \mathbf{0}}{\arg\max} \sum_{i=1}^{\phi^n} -\frac{\omega_i^n}{r_i^n}, \text{ subject to } \sum_{i=1}^{\phi^n} r_i^n \leq R^n. \quad (10)$$

For this problem the closed form solution is

$$\boldsymbol{r^n} = \left(\sum_{i=1}^{\phi^n} \sqrt{\omega_i^n}\right)^{-1} \cdot \begin{bmatrix} \sqrt{\omega_1^n} \\ \vdots \\ \sqrt{\omega_{\phi^n}^n} \end{bmatrix} \cdot R^n. \quad (11)$$

Here the rate is distributed proportionally to all flows, rather than only to the flow that has the largest weight. This allows the MDV scheduler to be readily used for intra-user scheduling, whereas MW-style schedulers may require additional intra-user scheduler mechanisms.

C. STABILITY

In this subsection, we discuss the concepts of stability and throughput optimality, and how these apply to the MDV scheduler.

We define a system with scheduling policy ψ to be (queue) stable if for an arrival rate vector $\boldsymbol{\lambda}$ the expected lengths of all queues in the system remain bounded. The stability region is the set of all arrival rate vectors $\boldsymbol{\lambda}$ for which scheduling policy ψ results in a stable system. Thus, an arrival rate vector outside the stability region could lead to a system in which one or more queues are not bounded, and grow to infinity.

Stability also leads to the concept of throughput optimality. Assume we have an optimal scheduling policy ψ^* whose stability region is maximal, i.e. queue stable for the largest set of arrival rate vectors, then this policy ψ^* is throughput optimal. Schedulers such as MW [5] are proven to be throughput optimal in some scenarios [32].

We show here that the MDV scheduler is throughput optimal for convex capacity regions. For non-convex rate regions it is possible to find an arrival rate vector such that for example the MW scheduler can stabilize the queues, while the MDV scheduler cannot. In practice this only occurs for a limited set of arrival rate vectors.

The proof we present here is based on the Lyapunov drift in a fluid system, and is similar to the proof of (Ω, α)-fairness in the context of bandwidth sharing [25], [33]. We first consider a general scheduler whose weights depend only on some constants and the queue sizes. If for such a system the arrival rate vector lies within the scheduler's stability region, then we show that the sum of the queue sizes will decrease, if we start from arbitrarily large queues. Next, we show that this also is true when we allow the constants to change at slot boundaries.

Finally, we show that the MDV scheduler must be stable for arrival rate vectors within the scheduler's stability region, by constraining it between two other, stable schedulers.

For the proof we make use of a general scheduler of the form

$$\boldsymbol{R}^* = \arg\max_{\boldsymbol{R}\in\mathcal{R}^\alpha} \sum_i \left(A_i Q_i(t) + B_i\right)^{\beta-1} \frac{(R_i+\zeta)^{1-\alpha}}{1-\alpha} \quad (12)$$

with constants $\zeta > 0$, $A_i > 0$, $B_i \geq 0$, $\beta > 1$ and $\alpha \in \mathbb{R}_+^0 \setminus \{1\}$. This scheduler belongs to the family of α-fair utility maximization functions [25]. To avoid division by zero when $R_i = 0$ and $\alpha > 1$, we have introduced ζ, which should be small compared to typical values of R_i.

In (12), $\mathcal{R}^\alpha \subseteq \mathcal{R}$ is the set of operating points the scheduler can select from a rate region $\mathcal{R}$ (we use $\mathcal{C}^\alpha$ for the corresponding capacity region). This set $\mathcal{R}^\alpha$ depends only on the parameter α. In [34] it is shown that α determines how much the rate region $\mathcal{R}$ is *convexified*. As α grows, operating points that are more interior to conv $\mathcal{R}$ are included in $\mathcal{R}^\alpha$, until $\mathcal{R}^\alpha = \mathcal{R}$. In Figure 5 we show an example rate region $\mathcal{R}$ and $\mathcal{R}^\alpha$ for $\alpha \in \{0, 0.5, 2\}$, where we can observe more points being selected from $\mathcal{R}$ as α increases. Note in particular that Figure 5a forms the convex hull of $\mathcal{R}$.

When the rate region $\mathcal{R}$ is continuous, we can approximate the rate region with a finite number of operating points [35]. In Section IV-D we have some remarks about this sampling process.

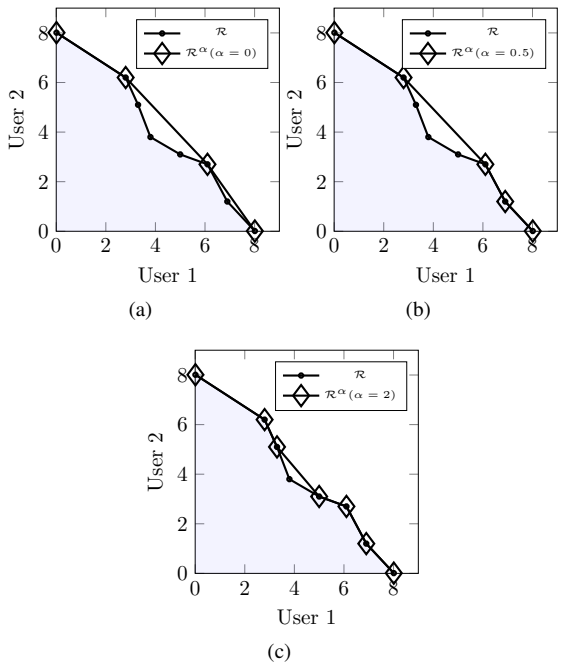


FIGURE 5. $\mathcal{R}$ and $\mathcal{R}^\alpha$ for a two users system ($\alpha \in \{0, 0.5, 2\}$)

Theorem 1. *The scheduler described by* $\boldsymbol{R}^* = \arg\max_{\boldsymbol{R}\in\mathcal{R}^\alpha} \sum_i \left(A_i Q_i(t) + B_i\right)^{\beta-1} \frac{(R_i+\zeta)^{1-\alpha}}{1-\alpha}$ *is stable if* $\boldsymbol{\lambda} \in \mathcal{C}^\alpha$.

The proof can be found in appendix A. There we also show that the constants A_i and B_i do not impact the stability region, but do reduce the rate at which the queues decrease.

We use this result and show that the system is still stable when $A_i(t)$ and $B_i(t)$ can change at the start of a slot, and remain constant for the duration of the slot.

Corollary 1.1. *In a slotted system*

$\boldsymbol{R}(t)^* = \arg\max_{\boldsymbol{R}\in\mathcal{R}^\alpha} \sum_i \left(A_i(t) Q_i(t) + B_i(t)\right)^{\beta-1} \frac{(R_i+\zeta)^{1-\alpha}}{1-\alpha}$

is stable for $\boldsymbol{\lambda} \in \mathcal{C}^\alpha$ *for any function* $A_i(t) > 0$, $B_i(t) \geq 0$. *During slot* t $A_i(t)$ *and* $B_i(t)$ *are constant.*

The proof is presented in appendix B, and follows the same steps as the previous proof. In the proof we make use of the fact that the weights remain constant, except at slot times, at which the derivatives are undefined. However at these time instants the queues themselves do not change. This corollary then leads to the stability of the MDV scheduler by bounding it between two schedulers with variable $A_i(t)$ and $B_i(t)$.

Corollary 1.2. *The MDV scheduler is stable if* $\boldsymbol{\lambda} \in \mathcal{C}^2$.

Proof. We can find constants $g_l, g_u > 0$, such that for any t we can upper and lower bound the class-dependent multiplier $c(\cdot)$ in (6) (as depicted by the dashed lines in Figure 6):

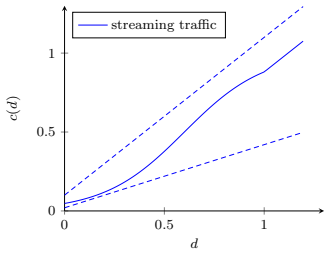


FIGURE 6. Multiplier for the streaming traffic class and bounds

$$\begin{aligned} & g_l \delta_c \frac{\tilde{\lambda}_i^n(t+1)}{R_i^{n\prime}(t)\hat{T}_i^n} Q_i^n(t) \\ <& \tilde{\lambda}_i^n(t+1) c_i^n \left(\delta_c \frac{Q_i^n(t)}{R_i^{n\prime}(t)\hat{T}_i^n} + (1-\delta_c) \frac{P\{D_i^n(t) < \hat{T}_i^n\}}{\varepsilon_i^n} \right) \\ <& g_u \delta_c \frac{\tilde{\lambda}_i^n(t+1)}{R_i^{n\prime}(t)\hat{T}_i^n} Q_i^n(t) + g_u(1-\delta_c) \frac{\tilde{\lambda}_i^n(t+1)}{\varepsilon_i^n} \end{aligned} \quad (13)$$

where $R_i^{n\prime}(t) = R_i^n(t) + \zeta$. Equation (13) is the same as the weight ω_i^n from the MDV scheduler, first described in Equation (6). As mentioned in Section IV, we assume that $\tilde{\lambda}_i^n(t+1) > 0$, thus our MDV scheduler function is bounded between functions of the form $A_L(t)Q(t)$ and $A_U(t)Q(t) + B_U(t)$ with $A_{\{L,U\}}(t) > 0$ and $B_U(t) > 0$. Hence using Corollary 1.1 we can conclude that the MDV scheduler also is stable. □

It is clear from (7) that using only the number of delay violations will result in a unstable MDV scheduler, as the

delay component is bounded by $1/\varepsilon_i^n$. In such a case it is not possible to find a lower bound with $A_L(t) > 0$. This shows that it is necessary to incorporate the queue length to keep the scheduler stable.

Other schedulers (e.g. [11]) bound the utility of best-effort traffic, to ensure that best-effort traffic will be low priority if the system load is high. In the MDV scheduler this could be also accomplished by setting $\rho = 0$ in (8). However, when $\rho = 0$ we encounter the same problem as for using only the delay distribution, i.e. we cannot find a lower bound with $A_L(t) > 0$, as now the traffic class c_{BE} is bounded. Hence in such case, the stability region is reduced, meaning that in some scenarios the best-effort queues can be unbounded, even if the arrival rate vector is within the capacity region.

We have shown here that the scheduler is guaranteed to be stable when the average arrival rate vector $\boldsymbol{\lambda}$ is within $\mathcal{C}^2$. An arrival rate vector outside of $\mathcal{C}^2$ does not necessarily result in unstable queues, as this depends on the arrival patterns of the users. It is usually very challenging to derive the exact stability region for such cases [36].

The rate region for our DSL setting (see Section V) is not convex, and there are thus arrival patterns for which the MDV scheduler cannot keep the queues stable. To test this we ran 10 000 simulations with the rate regions used in the simulations. Each simulation had a random average arrival vector within $\mathcal{R}$ (without considering $\mathcal{R}^2$). For constant bit-rate (CBR) traffic we found that in about 2% of the scenarios the MDV scheduler was not able to bound the queues whereas the MW scheduler was. Changing the fixed packet lengths of the CBR traffic to exponentially distributed lengths (but keeping the average arrival rate identical), dropped this number to about 0.3%. Thus in real-life scenarios the loss in stability region is very small as the traffic is much more diverse.

D. QUEUE PERFORMANCE FOR A DISCRETE RATE REGION

In the previous section we discretized a fluid rate region to discuss the stability region. This section provides an alternative view on the behavior of an MD-style scheduler when discretizing the rate region, as the sampling not only impacts $\mathcal{R}^\alpha$, but also the behavior for service rates close to zero.

MD-style schedulers can exhibit large queues when the rate region is not distributed well, as shown in the following example. Consider the utility function $u_{MD}(\rho) = Q/\rho$ in a system with a rate region $\mathcal{R} = \{\rho^a, \rho^e\}$ from Figure 7, i.e. having just two operating points. As this rate region is convex, the scheduler achieves the maximal stability region. Let user 1 and user 2 have an average arrival rate such that $\boldsymbol{\lambda} = [0.1, 0.1]$ packets/time unit. It is clear that the arrival rate vector is well within the capacity region $\mathcal{C}$.

Figure 8 shows the queue evolution $Q(t+1) = \max(0, Q(t) - \rho(t)) + A(t, t+1)$ for the two users of this system. Q_1 remains very close to $0.1 = \lambda_1 \cdot 1$, while Q_2 increases until it exceeds about 10 (or equivalently, when $\frac{Q_2}{Q_1} > \frac{(\rho_1^a)^{-1} - (\rho_1^e)^{-1}}{(\rho_2^e)^{-1} - (\rho_2^a)^{-1}} \approx 100$), after which Q_2 remains hovering around 10. We informally name the region in which the queues grow relatively large, despite low arrival rate vectors, the *pseudo-unstable* region. We name the region in which this does not occur the *pseudo-stable* region.

Increasing the number of operating points near the extremals reduces the arrival rate vector region in which this behavior occurs. For example, extending the rate region to $\mathcal{R} = \{\rho^a, \rho^c, \rho^e\}$ will reduce the aforementioned pseudo-unstable region to average arrival rate vectors in the horizontally shaded areas, giving us a pseudo-stable arrival rate region in the white square.

Intuitively, if $\lambda_2 > \rho_2^c = 0.5$, then only choosing operating point ρ^a can reduce user 2's queue. However, ρ^a is only chosen when $\frac{Q_2}{Q_1} > \max_{x \in c,e} \frac{(\rho_1^a)^{-1} - (\rho_1^x)^{-1}}{(\rho_2^x)^{-1} - (\rho_2^a)^{-1}} \approx 9998$, rendering it more difficult to chose ρ^a. Extending the region again to $\mathcal{R} = \{\rho^a, \rho^b, \rho^c, \rho^d, \rho^e\}$ reduces the pseudo-unstable to the square-shaded areas.

As can be induced, increasing the number of operating points near the service rates close to zero reduces this pseudo-unstable region.

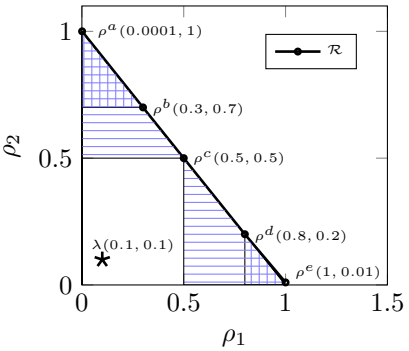


FIGURE 7. Rate regions for the example of Section IV-D

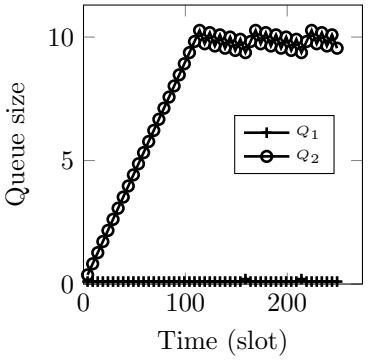


FIGURE 8. Queue evolution of a MD scheduler with two rate points

V. PHYSICAL LAYER MODEL

So far we have discussed the upper layers only. In this section, we will look at the physical layer and the discretization of the rate region to avoid problems when comparing the performance of the different schedulers.

In the simulations in Section VI we assume a downstream DSL G.fast physical layer. Ideally, such a system applies precoding or vectoring across all users in the network. All users may then be able to communicate free of interference. In some cases however, full vectoring is not available [37], [38] and one should resort to grouped vectoring (GV) instead. In a grouped vectoring system, two or more vectoring groups exist. Vectoring is then possible among users that are in the same group, but not among users that are in a different group. As a result of the uncanceled interference, competition for bandwidth among users between the vectoring groups is typically strong.

The G.fast implementation for this paper uses zero-forcing grouped vectoring as in [37]. In such a system vectoring matrices are fixed and different rate trade-offs can be made by varying the transmit power allocation $\boldsymbol{s}$. The power allocation $\boldsymbol{s}$ that is to be employed in each time slot can be determined by solving the following NUM problem:

$$\max_{\boldsymbol{s}\in\mathcal{S}} \sum_{n=1}^{N} u^n(R^n(\boldsymbol{s})). \tag{14}$$

In (14) we have the utility function u^n from (1). $R^n(\boldsymbol{s})$ expresses the rate of user n as a function of $\boldsymbol{s}$, the transmit power allocation of all users, which is chosen from the set of feasible power allocations $\mathcal{S}$. The NUM problem in (14) contains the spectrum coordination problem from [39] as a special case, and is therefore NP-hard [39]. As such, the locally optimal solution to (14) can be far away from the global optimum. Moreover, locally optimal power allocation algorithms may yield different results when different utility functions are considered, even when the utility functions are chosen in such a way that one would expect the same solution to be found. Using locally optimal solutions to the NUM problem as in (14) is therefore not ideal when the objective is to compare the performance of different schedulers, as one cannot exclude that the observed differences in performance are to be attributed to the behavior of the algorithm that is used to solve the non-convex NUM problem.

In order to obtain a reliable comparison of the different schedulers, we propose to compute a discrete set of power allocation settings $\hat{\mathcal{S}} \subset \mathcal{S}$ from which the scheduler can choose a single $\boldsymbol{s} \in \hat{\mathcal{S}}$. The achievable set of rate vectors will be defined as

$$\hat{\mathcal{R}} = \{\boldsymbol{r} \in \mathbb{R}_+^N | \exists \boldsymbol{s} \in \hat{\mathcal{S}} : \boldsymbol{r} = [R^1(\boldsymbol{s}), \ldots, R^n(\boldsymbol{s})]^T\}.$$

Each time-slot, the scheduler then chooses a power allocation $\boldsymbol{s} \in \hat{\mathcal{S}}$ by evaluating the objective function of (14) for each $\boldsymbol{r} \in \hat{\mathcal{R}}$ and selecting the rate vector that achieves the highest value. The considered set of power allocation settings $\hat{\mathcal{S}}$ will still be obtained by solving a set of NUM problems as in (14). However, the question of whether or not these power allocations correspond to global optimums of the NUM problem from which they are obtained is now irrelevant with respect to the scheduler's performance.

Some literature is available on selecting a representative set of DSL resource allocations in order to achieve good performance: [40] considers full-duplex DSL and constructs a set $\hat{\mathcal{S}}$ containing two elements to obtain a performance gain over time-division duplexing, and [41]–[43] employ multi-objective evolutionary algorithms to obtain a larger set $\hat{\mathcal{S}}$ containing resource allocations that are – in some sense – diverse. We will however take a more heuristic approach towards compiling $\hat{\mathcal{S}}$ by solving the following weighted sum rate maximization (WSRM) problem for a predetermined set of weight vectors $\hat{\mathcal{W}}$:

$$\max_{\boldsymbol{s}\in\mathcal{S}} \sum_{n=1}^{N} \omega^n \cdot R^n(\boldsymbol{s}). \tag{15}$$

The set $\hat{\mathcal{W}}$ is chosen such that the convex hull of the resulting set $\hat{\mathcal{R}}$ covers the rate region well, as this yields a large set of arrival rates that can be stabilized.

Before going into the details about how $\hat{\mathcal{W}}$ was compiled, we define a metric that quantifies how well a set $\hat{\mathcal{R}}$ covers its corresponding rate region. The coverage metric will be based on an inner and an outer approximation of the true rate region, which are respectively defined as

$$\hat{\mathcal{R}}_{in} = \text{conv} \bigcup_{\boldsymbol{r}\in\hat{\mathcal{R}}} \{(\{\boldsymbol{r}\} - \mathbb{R}_+^N) \cap \mathbb{R}_+^N\} \tag{16}$$

and

$$\hat{\mathcal{R}}_{out} = \bigcap_{\boldsymbol{r}\in\hat{\mathcal{R}}} \{\boldsymbol{q} \in \mathbb{R}_+^N | \boldsymbol{\omega}^T \cdot \boldsymbol{q} \leq \boldsymbol{\omega}^T \cdot \boldsymbol{r}\} \tag{17}$$

where conv$\mathcal{A}$ denotes the convex hull of the set $\mathcal{A}$ and the vector $\boldsymbol{\omega}$ in (17) is the weight vector $\boldsymbol{\omega} \in \hat{\mathcal{W}}$ that was employed in (15) to obtain the considered rate vector $\boldsymbol{r}$.

Note that $\hat{\mathcal{R}}_{in}$ also corresponds to the set of arrival rates that can be stabilized by a throughput optimal scheduler operating in $\hat{\mathcal{R}}$. Moreover, the approximation $\hat{\mathcal{R}}_{out}$ is based on the observation that each solved WSRM problem identifies a half-space that fully contains the original set of achievable rates $\mathcal{R} = \{\boldsymbol{r} \in \mathbb{R}_+^N | \exists \boldsymbol{s} \in \mathcal{S} : \boldsymbol{r} = [R^1(\boldsymbol{s}), \ldots, R^n(\boldsymbol{s})]^T$[3]. It can be readily seen that $\hat{\mathcal{R}}_{in} \subset \mathcal{R} \subset \hat{\mathcal{R}}_{out}$. The proposed coverage metric is then defined as

$$\text{Cover } \hat{\mathcal{R}} = \sqrt[N]{\frac{\text{Vol } \hat{\mathcal{R}}_{in}}{\text{Vol } \hat{\mathcal{R}}_{out}}} \tag{18}$$

where Vol $\mathcal{A}$ denotes the volume of the set $\mathcal{A}$. The N-th root is applied to the ratio of volumes to give a sense of "relative distance" between the two rate region approximations. If for instance $\hat{\mathcal{R}}_{out}$ is a scaled version of $\hat{\mathcal{R}}_{in}$, then the proposed measure will yield the value by which $\hat{\mathcal{R}}_{in}$ should be scaled to obtain $\hat{\mathcal{R}}_{out}$. Hence, the closer (18) is to 1, the better the match.

Using this metric, we can now compile and assess the set of weights $\hat{\mathcal{W}}$. First we construct $\hat{\mathcal{W}}_1$, the set of group weights

[3]Note that this outer approximation can be inaccurate, as the employed power allocation algorithm attains a locally optimal solution to the WSRM problem as in Equation (15), not the globally optimal solution.

that is obtained by uniformly sampling the unit $(N_{grp}-1)$-simplex, where N_{grp} is the number of vectoring groups:

$$\hat{\mathcal{W}}_1 = \{a\boldsymbol{\omega}_{grp} \in \mathbb{N}^{N_{grp}} | \mathbf{1}^T \boldsymbol{w}_{grp} = 1\} \quad (19)$$

where $a \in \mathbb{N}$ determines the sampling density. We have chosen $a = 20$.

This set $\hat{\mathcal{W}}_1$ is now modified to include the user weights. For every group $i \in 1..N_{grp}$ the weights are expanded to a vector of length M_i with each component having value ω_i, where M_i is the number of users in group i. Finally, we again iterate over each group i, now setting its weights to $\mathbf{1}^T\omega_i \cdot \boldsymbol{b}$ where $\boldsymbol{b}$ is any combination of 0 and 1 (excluding $\mathbf{0}$ as that is already covered by $\omega_i = 0$), resulting in $2^{M_i}-1$ combinations for group i. The resulting set of weights is denoted by $\hat{\mathcal{W}}_2$.

Before showing the coverage results, we first describe the G.fast networks that are used in the simulations of Section VI, and then apply the algorithm to obtain the sampled rate regions. Three G.fast networks are considered: one with two vectoring groups each containing two users (2g2u), a network with two vectoring groups each containing three users (2g3u), and finally a network with three vectoring groups each containing two users (3g2u, see Figure 9 for this rate region, where for each group all user rates are summed to reduce dimensionality). The channel matrices are based on lab measurements of a 104m long cable [44]. The considered cable type is representative for access cables that are widely used by KPN in the Netherlands. The employed G.fast parameter settings are summarized in Table 1 (we refer to [37] for further details).

Parameter	Value
P_{tot}	4 dBm
$\sigma_{k,i,n}$	-140 dBm
f_s	48 kHz
Δ_f	51.75 kHz
Γ	10 dB
s_{mask}	n/a
K	2047

TABLE 1. Summary of G.fast parameter settings

In Table 2 we show the coverage results for the three different networks. It can be seen that this simple sampling algorithm approximates the rate region quite well, as the numbers in the cover $\hat{\mathcal{R}}$ column are all very close to 1.

	$\lvert\hat{\mathcal{R}}\rvert$	Vol $\hat{\mathcal{R}}_{in}$	Vol $\hat{\mathcal{R}}_{out}$	Cover $\hat{\mathcal{R}}$
2 users, 2 groups	189	1.9311e+36	2.0964e+36	0.9797
3 users, 2 groups	1029	2.0361e+54	2.5787e+54	0.9614
2 users, 3 groups	6237	1.8691e+54	2.2279e+54	0.9712

TABLE 2. Different networks and their coverage for $\hat{\mathcal{W}}_2$

VI. EVALUATION

In this section, we evaluate the MDV scheduler using simulations and by comparing it to other schedulers from literature, as well as to an ideal scheduler. In Section VI-A we describe the setup, including the runtime settings, the metrics investigated and the plot layout. Then we look at the simulation results themselves in Section VI-B.

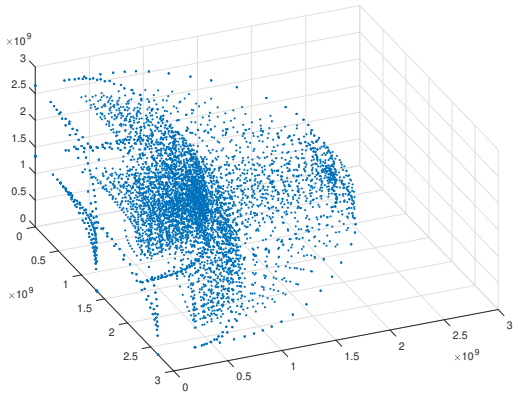


FIGURE 9. Rate region for 3 groups with 2 users (in bit/second)

A. SETUP

In this section, we describe how the evaluation was performed. First we give an overview of the settings and schedulers used. In Section VI-A1 we briefly show the intra-user scheduler settings used in the simulations. We continue with enumerating the metrics that we have used in Section VI-A2 and introducing the plot layouts in Section VI-A3.

The simulations were run in OMNeT++ using the INET framework. Each of the N users has a number of flows that send traffic to a sink over a channel using a fluid model. There is a warm-up time of 5 s, during which no results are recorded. Every $\tau = 50$ ms the weights are computed by a scheduler. Prior to this computation, packets whose HOL exceeds $\hat{T}$ are removed from the queue. Packets that are late and in transit will still be delivered, however.

The schedulers are summarized in Table 3 on Page 11. We have also included an approximation to an *ideal* scheduler, called the Oracle scheduler. This algorithm has access to future arrivals and can select the optimal rate from the rate region in order to minimize the number of delay violations, while at the same time maximizing the system throughput. To reduce the simulation time of the Oracle scheduler, some shortcuts were taken. The first shortcut is that the Oracle scheduler only looks at the next $M = 2$ future slots. Increasing M would allow for better handling of bursts and increased throughput. The second shortcut is that we approximate the runtime delay distribution used by the scheduler, resulting sometimes in a temporarily suboptimal rate selection. Even though these simplifications result in a slightly suboptimal result, mainly when the load is high, the results offer a valuable benchmark that can be used to compare the other schedulers to.

The simulations cover different scenarios. Each scenario is

Scheduler	Weight	Notes
EXP-MLWDF (2018) [45]	$\frac{\alpha}{\overline{\rho}} \exp\left(\frac{\hat{T}}{\hat{T}-\Gamma}\right)$	
EXP/PF (2003) [46]	$\exp\left(\frac{\alpha\Gamma-\overline{\Gamma}}{1+\sqrt{\overline{\Gamma}}}\right)\frac{1}{\overline{\rho}}$	$\overline{\Gamma}$ is the average HOL of all real-time flows
Lei (2007) [47]	$\frac{a\exp(-a\cdot(\Gamma-\hat{T}))}{(1+\exp(a\cdot(\Gamma-\hat{T})))^2}$	$a = 1.5$
JUPS (2018) [48]	$\frac{1}{\overline{\rho}}\frac{-\log(\varepsilon)\Gamma}{\hat{T}}$	
JSM (2018) [13]	$-\frac{\mu}{\sigma}\frac{v}{(1-v)^2}\frac{1}{1+e(1/(1+v),1,\mu,\lambda)}$	$\mu = 1, \lambda = -0.1088$, $v = e(\Gamma, \hat{T}, \mu, \sigma)$, $\sigma = \frac{\mu\hat{T}}{2\ln(0.99^{-1}-1)}$, $e(x, x_r, a, b) = \exp(\frac{a}{b}(x - x_r)$
MQS (2015) [49]	$\frac{\exp(a)}{\exp(a)+\exp(-a)}$	$a = \frac{\Gamma-\hat{T}}{\hat{T}}$
MPT (2018) [23]	$\Gamma \cdot \frac{\hat{T}-\Gamma}{\exp(\sqrt{1+\hat{T}-\Gamma})}$	
Queue-HoL-M-LWDF (QHMLWDF) (2013) [22]	$\frac{\alpha\Gamma Q}{\overline{\rho}}$	
Wu (2020) [1]	Q	Uses utility function $u(R) = Q\cdot\log(1+R)$
eEXPRule (2020) [24]	$\exp\left(\frac{5\Gamma}{\hat{T}(1+\sqrt{\overline{\Gamma}})}\right)$	$\overline{\Gamma}$ is the average HOL of all real-time flows
Oracle		
MDV	$-\tilde{\lambda}c\left(\delta_c\frac{Q}{R\hat{T}}+(1-\delta_c)\frac{P\{D(t)<\hat{T}\}}{\varepsilon}\right)$	Reciprocal scheduler. $\tilde{\lambda}$ is a prediction of future arrival, $\delta_c \in [0, 1]$

TABLE 3. Summary of the considered schedulers (in no particular order) and their settings. Common symbols: $\overline{\rho}$ (averaged service rate), $\overline{\lambda}$ (averaged arrival rate) and $\alpha = -\ln(\varepsilon)/\hat{T}$.

repeated 20 times, with a random seed based on the repetition index. The traffic generated is the same for all (scenario, repetition)-tuples, i.e. each scheduler will receive the same traffic. The traffic can be categorized by traces (videos), regular generated traffic such as traffic with packet arrivals according to a Poisson process and exponentially distributed packet sizes (which we refer to as M/M/1), VoIP, CBR, heavy-tailed (self-similar traffic) and a traffic source called SAT that keeps the output line saturated by keeping the queue always backlogged. These traffic flows are considered to be of the streaming traffic class by default. In some scenarios we set the traffic class of M/M/1 to best-effort.

1) Intra-user scheduling

We mainly consider the scenarios in which all flows have their own channel. For some scenarios we also consider using a regular scheduler, as the one flow, one channel regime is very disadvantageous to the linear schedulers. In such case, each of the users' flows are inputs to the scheduler, which is then directed to the users' only output channel. We employ the parameterless EDF scheduler, which serves the flow whose HOL packet has the most stringent deadline. The NUM problem is then reduced to

$$\arg\max_{\boldsymbol{R}\in\mathcal{R}} \sum_{n=1}^{N} \omega^n R^n$$

with $\omega^n = \sum_{i=1}^{\phi^n} \omega_i^n$.

2) Metrics

We assess the schedulers' performance using the following metrics:

- **Packet loss ratio** $\frac{A_p(0,T_{sim})-E_p(0,T_{sim})}{A_p(0,T_{sim})}$: The number of packets dropped due to their delay being too large, with A_p and E_p the number of respectively arrivals and departures. Here, T_{sim} is the length of the simulation. Note that we show the plain PLR, without taking ε into account, in order to not complicate this metric. Lower is better.
- **Average throughput** $\frac{E(0,T_{sim})}{T_{sim}}$: The total number of bits that have been sent successfully. This is influenced by both the packet loss ratio, and the SAT traffic type, which keeps the queue backlogged. Higher is better.
- **Average delay** $\mathbb{E}[D]$: The delay is calculated as the time difference between creation and arrival of a packet. Delays can occasionally exceed $\hat{T}$ if service is lowered while a packet is being transmitted. Lower is better.

3) Plot layout

We make use of two types of plots in the results section:

- **Violin plot**: Violin plots consist of a rotated and mirrored histogram that is smoothed using a kernel density estimator. The data that are used to compose the histogram come from the results of all the scenarios. The schedulers are arranged horizontally. The wider a violin plot is, the higher its frequency at that point. Inside the shaded area, the quartiles are displayed. For easier comparison, a line connects the mean values. We present the data like this as it gives a good summary over the many data points.

- **Line plot**: Each individual line represents a different scheduler. The y-axis is the average of all selected scenarios, while the x-axis is the independent parameter, such as the number of multiplexed flows per user.

B. RESULTS

In this section, we discuss the results of the simulations. First, in Section VI-B1, we consider scenarios in which each user has exactly one flow. Then we continue with scenarios representing typical use cases in Section VI-B2, scenarios where the system load is close to 1 in VI-B3, scenarios that focus on heavy-tailed traffic in VI-B4 and statistical multiplexing in VI-B5.

As the schedulers' performances are similar among the different rate regions, we show the averaged results, unless otherwise noted.

1) Single flow scenarios

In this first section, we consider scenarios in which each user has exactly 1 flow. The flows are a mix of different kinds of traffic. This set of scenarios highlights that the excellent performance of the MDV scheduler does not solely depend on its intra-user scheduler.

Figure 10a shows the PLR distribution for the flows of the scenarios. There we can clearly see that the MDV scheduler performs very close to the Oracle scheduler. All other schedulers have outliers of at least 20%, while for the MDV scheduler the PLR for most flows stays below 5%.

The throughput plot shown in Figure 10b shows that the MDV scheduler has the best throughput performance. The Oracle scheduler's throughput is less as it trades service to achieve a lower PLR .

2) Regular scenarios

The regular scenarios comprise a mix of different sources where each user has about 4 flows.

Figure 11a shows the PLR distribution for the flows for the regular 4- and 6-user scenarios, as the curves for the different schedulers are shaped very similarly. There we can see that most schedulers cannot cope well with the traffic offered. Within a scenario, usually some applications have a low PLR, at the cost of other flows. These flows are mainly video flows and high volumes of M/M/1 generated traffic.

For the MDV scheduler, however, the PLR remains very close to 0% for all scenarios. Only VoIP sometimes suffers losses, occasionally up to 10%. This is due to its low volume, but highly bursty nature. The low volume (compared to e.g. video) makes it more difficult to assign a large enough weight, while the burstiness, coupled with the delay of 1 slot until rates are in effect, cause a relatively large PLR . Hence, for such flows it might be more effective to consider them to be CBR streams and assign a fixed service rate in the physical layer. The performance is very similar to that of the Oracle scheduler. Due to the low PLR, we also have a higher throughput, as can be observed in Figure 11b.

Figure 12 shows the same scenarios, but now allocating resources per user, and scheduling flows with the EDF scheduler, as described in Section VI-A1. It should not surprise us that the PLR is lower for the linear schedulers, especially for the Lei scheduler, as flows for a single user are multiplexed. This is also visible in the increased throughput. Despite this, the MDV scheduler still has a better performance for both the PLR and throughput. Also here, there are MDV outliers caused by VoIP traffic. The six left-most schedulers perform significantly better in these scenarios with EDF scheduler, as can be observed by the decreased mean and the wider part being at the bottom of the plot.

3) High load

The scenarios from this subsection deal with traffic that is close to the rate region boundary, resulting in a high system load. Each user sends either one flow of type M/M/1 or CBR that averages to $0.99 \cdot \boldsymbol{R}^*$, where $\boldsymbol{R}^*$ is the solution to the NUM problem for weights $\mathbf{1}$. For each of the flows, we impose the same QoS restriction $P\{D(t) > 100 \text{ ms}\} \leq 0.01$.

Figure 13a shows the best PLR performance for the MQS, Wu and MDV schedulers the latter two of which perform close to the ideal. Especially the MPT and eEXPRule schedulers have difficulty to deal with the offered load. The EXP-MLWDF scheduler features three distinct blobs. This is due to the scheduler giving either all, some or close to none service to the same flows during the simulation.

In a high load scenario flows sometimes get a lot of service, but as the packet has almost finished transmitting, other flows have considerably more weight. Due to the used fluid model, this can result in some packets have large delays, as can be observed for the EXP-MLWDF and MPT schedulers in the delay plot Figure 13b. All other schedulers have a delay upper bound close to 100 ms, as would be expected. The MDV and Oracle schedulers' delay distributions are very close to to 100 ms, indicating that the delay fairness of the users is quite good: all users experience similar delays.

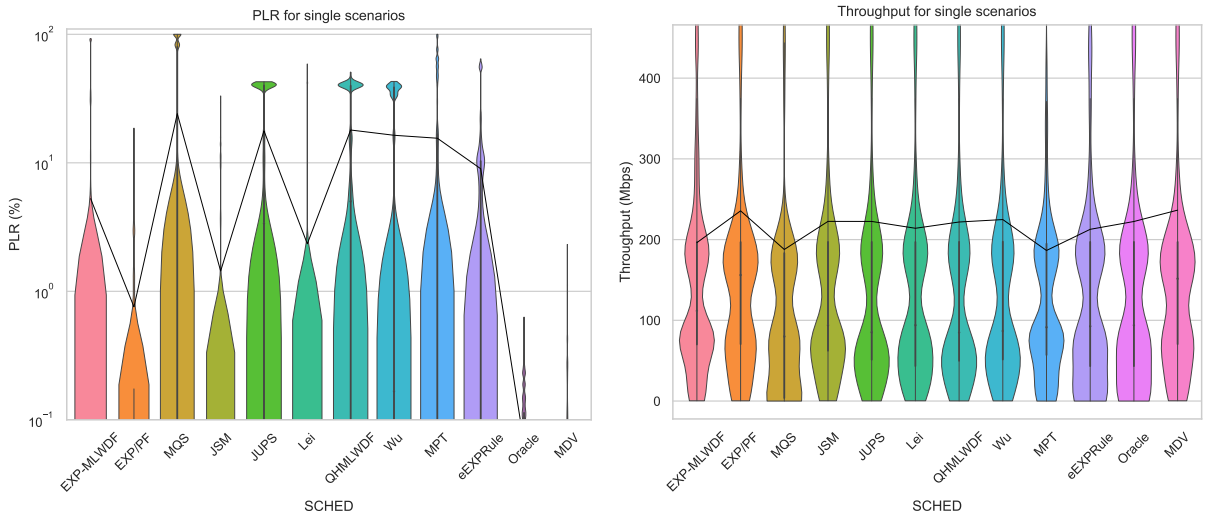
4) Heavy-tailed traffic

In the heavy-tailed scenarios, half of the users are sending one flow with regular M/M/1 traffic, while the other users have one flow that comprises a superposition of Pareto-distributed traffic with ON- and OFF-periods, resulting in heavy-tailed traffic [50].

The PLR in Figure 14a shows that the EXP/PF, MQS, Wu and MDV schedulers have a similarly good performance. The EXP-MLWDF and MPT schedulers have two visible parts: the M/M/1 traffic occupies the lower part, while the self-similar traffic has a large PLR .

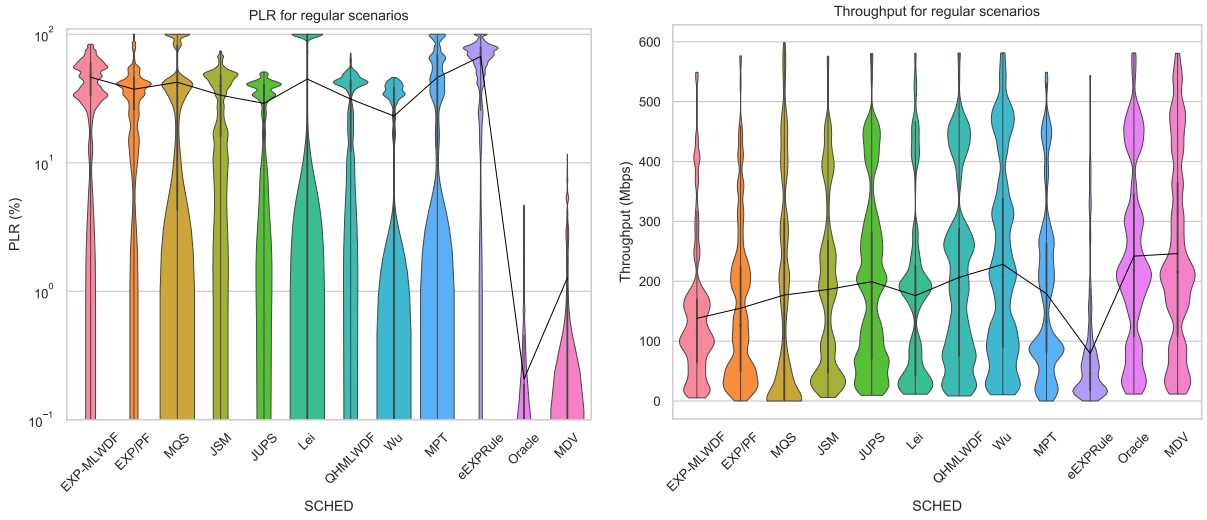
5) Statistical multiplexing

The raison d'être of packet switched networks is multiplexing. It is well known that the peak rate of an aggregate of bursty traffic is significantly lower than the sum of the peak rates. Hence, instead of reserving rates for the peak rate of every flow, squandering service rate, flows can be multiplexed. This



(a) PLR (b) throughput

FIGURE 10. The PLR and throughput for scenarios with one flow per user



(a) PLR (b) throughput

FIGURE 11. The PLR and throughput for regular scenarios

subsection looks at the schedulers' performances when each user sends a number of Starwars videos [51]. The x-axis in the plots in Figure 15 represent the number of videos each user is transmitting. On the y-axis, we can see the average of the metric.

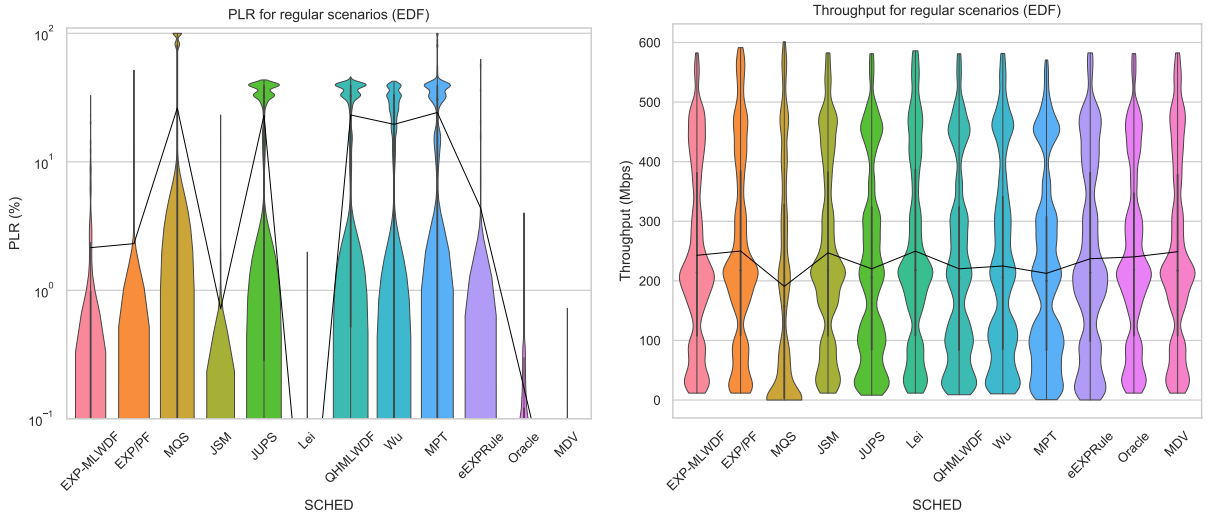
Figure 15a shows the PLR for the 2g2u rate region, which more clearly shows the behavior of the schedulers. The other rate regions behave similarly, but are shifted. We can clearly discern two groups: the MDV and Oracle schedulers, that increase linearly with N, and all other schedulers that peak quickly and then decrease, decrease a little to increase again around $N = 10$.

The performance of the MDV scheduler is relatively close to the Oracle scheduler. It is able to support three video flows per user. When $N > 4$, the violation rate increases linearly.

If we use the EDF scheduler, then we can observe in Figure 15b that all schedulers perform significantly better. Again, this does not surprise, as the flows are now multiplexed per user. However, there are still plenty of schedulers that have difficulty coping with more than four flows. In this plot we can also notice that, in addition to MDV, the MPT, EXP/PF and MQS schedulers all are also able to multiplex 8 flows.

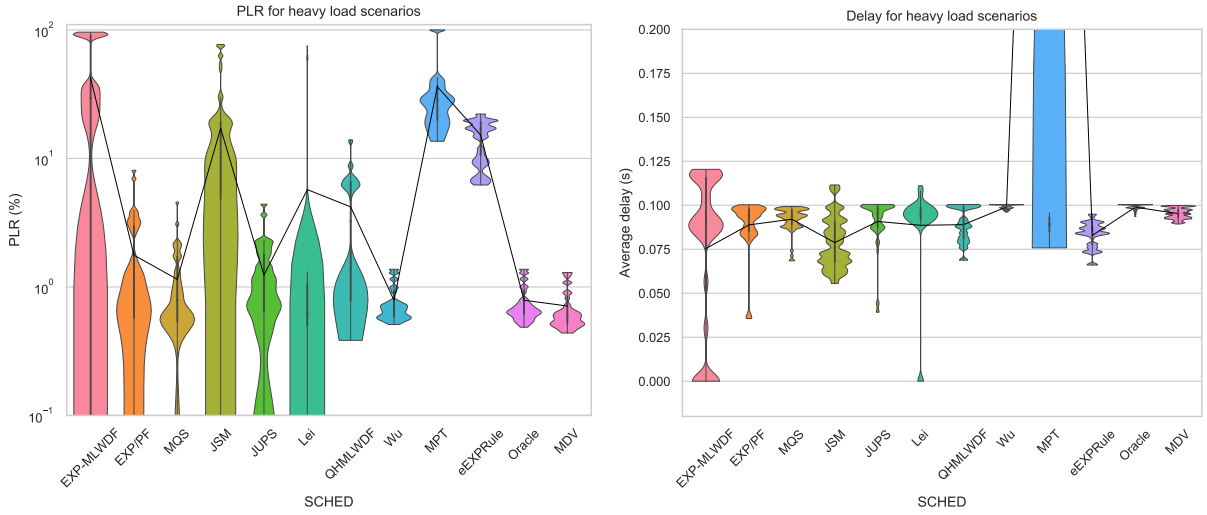
VII. CONCLUSION

In this paper, we have presented a new cross-layer scheduler for DSL networks. The scheduler combines the arrival rates,



(a) PLR (b) throughput

FIGURE 12. The PLR and throughput for regular scenarios for the EDF scheduler



(a) PLR (b) average delay

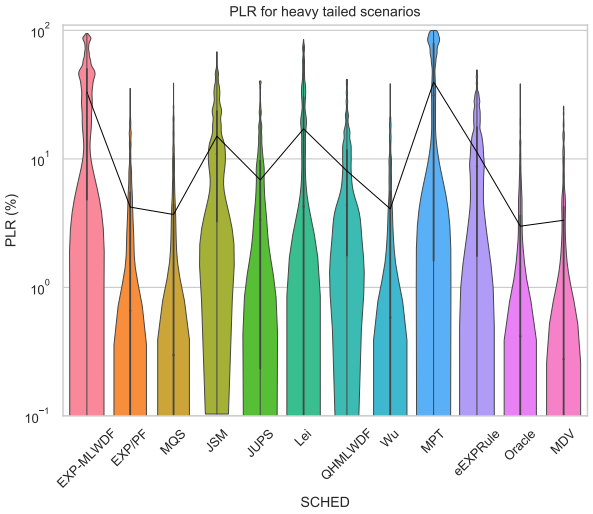
FIGURE 13. The PLR and average delay for high load scenarios

current queue sizes and the past observed delays to generate a weight that, in combination with a utility function of the form $u(R, \omega) = \frac{\omega}{R+\zeta}$ minimizes the number of delay violations. We have discussed the stability region of the MDV scheduler, and showed that it is throughput optimal for convex rate regions. We have compared the PLR, throughput and delay performance to other cross-layer schedulers from literature, together with an Oracle scheduler, and demonstrated that it performs at least as good, and usually substantially better, than the other schedulers in the tested scenarios, which included multiplexing and heavy-tailed traffic. The scheduler works excellent in the case where each flow has its own "channel", but also performs equally or better than the other cross-layer schedulers when using the EDF scheduler as the intra-user scheduler. The MDV scheduler also often performs close to the Oracle scheduler (with respect to QoS requirements that are specified as a percentile of the delay distribution).

.

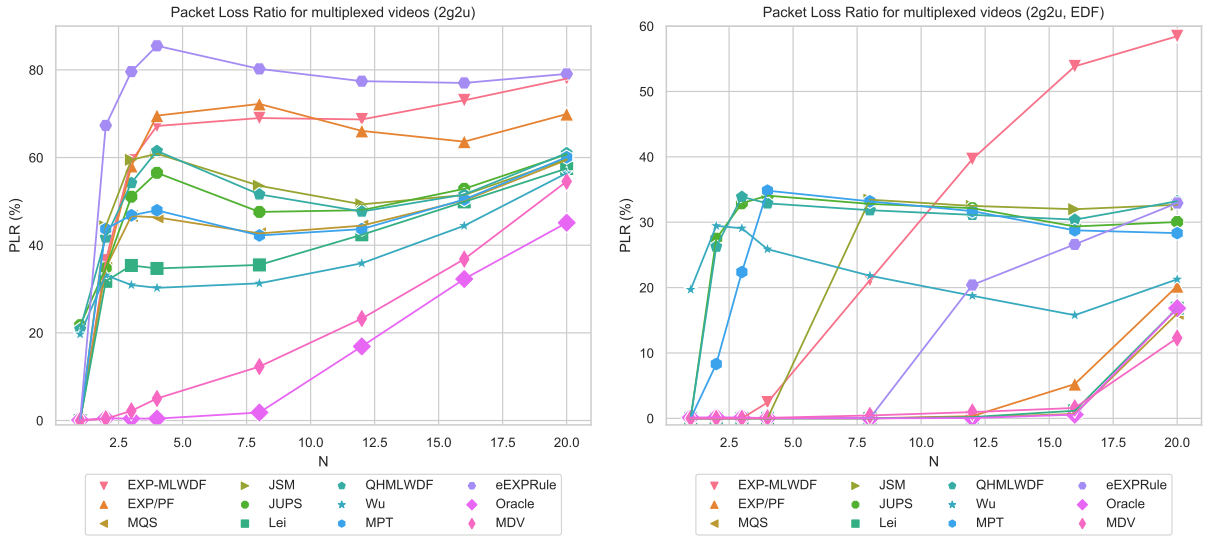
APPENDIX A PROOF OF STABILITY FOR CONSTANT A AND B

For this proof we model the queue using a Markov chain. We assume a flow i has packet arrivals according to a Poisson process with parameter ν_i. The packet sizes are exponentially distributed with mean μ_i^{-1}. Define the average arrival rate vector $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_n] = [\frac{\nu_1}{\mu_1}, \ldots, \frac{\nu_n}{\mu_n}]$. By including the residual inter-arrival and service times we can extend this



(a) PLR

FIGURE 14. The PLR for the heavy-tailed traffic scenarios



(a) PLR

(b) PLR using the EDF scheduler

FIGURE 15. The PLR for multiplexing Starwars videos for the 2g2u rate region using the regular intra-user scheduler and EDF scheduler

result to renewal arrival processes and generally distributed packet sizes [52], and obtain results for general distributions.

The transition rates of the queues that describe the system are given by

$$\begin{aligned} Q_i &\to Q_i + 1 && \text{at rate } \nu_i \\ Q_i &\to Q_i - 1 && \text{at rate } \mu_i R_i^* \end{aligned}$$

where R_i^* is given by Equation (12).

We will now look at the ergodicity of this Markov process Q_i. A Markov process is called ergodic if all the states in the Markov chain are aperiodic and positive recurrent, i.e. all states can be visited from any of the states with non-zero probability within a finite period. Thus, if the process is ergodic, then the queues are stable.

Proof. We first look at the fluid system, when the initial queues grow to infinity:

$$X_i(t) = \lim_{\omega \to \infty} \frac{Q_i(\omega t)}{\omega} \text{ with } Q_i(0) = \omega, \forall i.$$

Define $\boldsymbol{X}(t) = [X_1(t), \ldots, X_n(t)]$. Given the initial distribution of $\boldsymbol{X}(0) = \mathbf{1}$, it follows from the strong law of large numbers that the evolution of the fluid $\boldsymbol{X}(t)$ is defined by

$$\dot{X}_i(t) = \nu_i - \mu_i R_i(t)$$

for all t such that $X_i(t) > 0$. Here, $\dot{X}_i(t)$ is written using Newton's notation, i.e. $\dot{X}_i(t) = \frac{d}{dt} X_i(t)$. $\boldsymbol{R}(t)$ is the solution to Equation (12). If the traffic conditions

$$\boldsymbol{\lambda} \in \mathcal{C}^{\alpha} \tag{20}$$

are satisfied, then we show that there exists a constant $T > 0$, such that $\boldsymbol{X}(t) = \boldsymbol{0}, \forall t \geq T$. For this, we define Lyapunov function F and scheduler G as

$$F(\boldsymbol{u}) = \sum_i \frac{(A_i u_i + B_i)^\beta}{\mu_i (\lambda_i + \zeta)^\alpha A_i \beta},$$

$$G(\boldsymbol{u}) = \sum_i (A_i X_i(t) + B_i)^{\beta-1} \frac{(u_i + \zeta)^{1-\alpha}}{1-\alpha}.$$

The Lyapunov function represents a scalar measure of the queue sizes in the system and will be large if at least one of the queues is large. Differentiating $F(\boldsymbol{X}(t))$ with respect to t we get the Lyapunov drift:

$$\dot{F}(\boldsymbol{X}(t)) = \sum_i \frac{(A_i X_i(t) + B_i)^{\beta-1}}{(\lambda_i + \zeta)^\alpha} (\lambda_i - R_i(t)), \quad (21)$$

using the fact that $\lambda_i = \frac{\nu_i}{\mu_i}$.

Let $\boldsymbol{R}(t) = \arg\max_{\boldsymbol{u} \in \mathcal{R}} G(\boldsymbol{u}) = \arg\max_{\boldsymbol{u} \in \mathcal{R}^\alpha} G(\boldsymbol{u})$. Thus $\boldsymbol{R}(t)$ attains the maximum over $\mathcal{R}^\alpha$, and we have that for any $\boldsymbol{u}$ the gradient of G satisfies

$$\nabla G(\boldsymbol{R}(t)) \cdot (\boldsymbol{u} - \boldsymbol{R}(t)) \leq 0,$$

where $\cdot$ is the dot-product. By concavity of G, we obtain that

$$\nabla G(\boldsymbol{u}) \cdot (\boldsymbol{u} - \boldsymbol{R}(t)) \leq 0. \quad (22)$$

Under the stability condition (20), we can find an $\epsilon > 0$ such that $\boldsymbol{u} = (1+\epsilon)\boldsymbol{\lambda} \in \mathcal{C}^\alpha$. Applying $\boldsymbol{u}$ to (22) results in

$$\textstyle\sum_i (A_i X_i(t) + B_i)^{\beta-1} (\lambda_i(1+\epsilon) + \zeta)^{-\alpha} (\lambda_i(1+\epsilon) - R_i(t)) \leq 0$$

which can be reduced to

$$\sum_i (A_i X_i(t) + B_i)^{\beta-1} (\lambda_i + \zeta)^{-\alpha} (\lambda_i(1+\epsilon) - R_i(t)) \leq 0.$$

This can be rewritten using (21) to obtain

$$\begin{aligned}
\dot{F}(\boldsymbol{X}(t)) &\leq -\epsilon \sum_i \mu_i^{-1} (\lambda_i + \zeta)^{-\alpha} (A_i X_i(t) + B_i)^{\beta-1} \\
&\leq -\epsilon \sqrt[\beta]{\min_i (\mu_i^{-1} (\lambda_i + \zeta)^{-\alpha})} \cdot \\
&\quad \left(\sum \mu_i^{-1} (\lambda_i + \zeta)^{-\alpha} (A_i X_i(t) + B_i)^\beta \right)^{\frac{\beta-1}{\beta}} \quad (23) \\
&\leq -\epsilon \sqrt[\beta]{\min_i (\mu_i^{-1} (\lambda_i + \zeta)^{-\alpha})} (\beta \min(\boldsymbol{A}))^{\frac{\beta-1}{\beta}} \cdot \\
&\quad \left(\sum \frac{\mu_i^{-1} (\lambda_i + \zeta)^{-\alpha}}{\beta A_i} (A_i X_i(t) + B_i)^\beta \right)^{\frac{\beta-1}{\beta}} \quad (24) \\
&= -\theta F(\boldsymbol{X}(t))^{\frac{\beta-1}{\beta}}
\end{aligned}$$

In step (23) we employed the well-known inequality $||\boldsymbol{a}||_q \leq ||\boldsymbol{a}||_p \leq n^{1/p-1/q} ||\boldsymbol{a}||_q$, where $||\boldsymbol{a}||_p$ is the p-norm of a vector a, $0 < p < q$, and n the number of elements in the vector. Let in the following $a_i = \sqrt[p]{w_i} x_i$, $p = \beta - 1$, $q = \beta$ and $\boldsymbol{A} = [A_1, \ldots, A_n]$ then

$$\begin{aligned}
&||\boldsymbol{a}||_\beta \leq ||\boldsymbol{a}||_{\beta-1} \\
&\iff \left(\sum w_i^{\frac{\beta}{\beta-1}} x_i^\beta \right)^{1/\beta} \leq \left(\sum w_i x_i^{\beta-1} \right)^{1/(\beta-1)} \\
&\iff \min(\boldsymbol{w})^{\frac{1}{\beta}} \left(\sum w_i x_i^\beta \right)^{\frac{\beta-1}{\beta}} \leq \sum w_i x_i^{\beta-1} \\
&\iff -\sum w_i x_i^{\beta-1} \leq -\min(\boldsymbol{w})^{\frac{1}{\beta}} \left(\sum w_i x_i^\beta \right)^{\frac{\beta-1}{\beta}}
\end{aligned}$$

Step (24) uses Abel's inequality.

Now, if there exists $T > 0$ for which $F(\boldsymbol{X}(T)) = 0$, then it is clear that $F(\boldsymbol{X}(t))$ will always be able return to 0, $\forall t \geq T$. As $F(\boldsymbol{X}(t)) = 0$ implies that $\boldsymbol{X}(t) = \boldsymbol{0}$, and thus also implies that the queues are bounded, this concludes the proof. □

Furthermore,

$$\begin{aligned}
&\dot{F}(\boldsymbol{X}(t)) \leq -\theta F(\boldsymbol{X}(t))^{\frac{\beta-1}{\beta}} \\
\iff &\dot{\ln}(F(\boldsymbol{X}(t))) F(\boldsymbol{X}(t))^{\frac{1}{\beta}} \leq -\theta.
\end{aligned}$$

Integrating both sides results in

$$\begin{aligned}
&\int_0^t \dot{\ln}(F(\boldsymbol{X}(s))) F(\boldsymbol{X}(s))^{\frac{1}{\beta}} ds \leq \int_0^t -\theta ds \\
\iff &\beta F(\boldsymbol{X}(s))^{\frac{1}{\beta}} |_0^t \leq -\theta t \\
\iff &F(\boldsymbol{X}(t)) \leq \left(F(\boldsymbol{X}(0))^{\frac{1}{\beta}} - \frac{\theta}{\beta} t \right)^\beta.
\end{aligned}$$

This implies that $F(\boldsymbol{X}(t)) = 0$ for all $t \geq T$, with

$$T = \frac{1}{\epsilon} \sqrt[\beta]{\sum_i \frac{\min_j (\mu_j (\lambda_j + \zeta)^\alpha) \frac{(A_i + B_i)^\beta}{A_i}}{\mu_i (\lambda_i + \zeta)^\alpha \min(\boldsymbol{A})^{\beta-1}}}.$$

The inclusion of a constant B_i does not impact the stability region (but does increase T). Also, multiplying $\boldsymbol{A}$ by a constant c will cancel out, and have no effect on T. Modifying A_i does influence T, as then more service is allocated to flow i, leaving less service for other flows. The smallest T is reached when all A_i are equal. The ζ parameter adds a constant to the arrival rates λ_i. As a typical ζ is small, its influence on T is limited.

We can obtain the MD scheduler for $\beta = 2$, $\alpha = 2$, $\boldsymbol{A} = \boldsymbol{1}$ and $\boldsymbol{B} = \boldsymbol{0}$, resulting in

$$T_{MD} = \frac{1}{\epsilon} \sqrt{\sum_i \frac{\min_j (\mu_j (\lambda_j + \zeta)^2)}{\mu_i (\lambda_i + \zeta)^2}}.$$

For the MW scheduler ($\beta = 2$ and $\alpha = 0$) we get

$$T_{MW} = \frac{1}{\epsilon} \sqrt{\frac{\min_j (\mu_j)}{\sum_i \mu_i}}.$$

Both schedulers are thus clearly stable when $\boldsymbol{\lambda} \in \mathcal{C}^\alpha$ as the upper bound exists. Additionally, for the MW scheduler,

we also have throughput optimality (i.e. stability region is maximal) as $\mathcal{R}^0$ forms a convex hull of the rate region. In [34] it is shown that there exists a $\gamma > 0$, such that the scheduler with $\alpha < \gamma$ also is throughput optimal. This γ depends on the shape of the rate region. For a convex rate region $\gamma = \infty$, and thus all α-fair schedulers are throughput optimal.

Finally, we can also observe that $\lim_{\zeta\to\infty} T = T_{MW}$, i.e. we can make any of the schedulers approach the MW scheduler by increasing ζ (and thus making it throughput optimal).

APPENDIX B PROOF OF STABILITY FOR TIME-DEPENDENT A AND B

Proof. Analogous to the proof of Theorem 1, we define

$$F(\boldsymbol{u}) = \sum_i \frac{\mu_i^{-1}(\lambda_i+\zeta)^{-\alpha}}{\beta} \frac{(A_i(t)u_i + B_i(t))^\beta}{A_i(t)},$$

$$G(\boldsymbol{u}) = \sum_i (A_i(t)X_i(t) + B_i(t))^{\beta-1} \frac{(u_i+\zeta)^{1-\alpha}}{1-\alpha}.$$

Differentiating $F(\boldsymbol{X}(t))$ with respect to t results in

$$\dot{F}(\boldsymbol{X}(t)) = \sum_i (\lambda_i+\zeta)^{-\alpha}(A_i(t)X_i(t)+B_i(t))^{\beta-1} \cdot (\lambda_i - R_i(t)) + a(t) + b(t)$$

where

$$a(t) = \sum_i (\lambda_i+\zeta)^{-\alpha}\mu_i^{-1}\dot{A}_i(t)\cdot(A_i(t)X_i(t)+B_i(t))^{\beta-1} \cdot \left(\frac{X_i(t)}{A_i(t)} - \frac{A_i(t)X_i(t)+B_i(t)}{A_i^2(t)\beta}\right)$$

and

$$b(t) = \sum_i (\lambda_i+\zeta)^{-\alpha}\mu_i^{-1}\dot{B}_i(t)\cdot\frac{(A_i(t)X_i(t)+B_i(t))^{\beta-1}}{A_i(t)}.$$

Repeating the same inequality steps as in the previous proof, we arrive at

$$\dot{F}(\boldsymbol{X}(t)) \le -\theta(t)F(\boldsymbol{X}(t))^{\frac{\beta-1}{\beta}} + a(t) + b(t).$$

In the fluid system considered here, $a(t)$ and $b(t)$ depend on $\dot{A}_i(t)$ and $\dot{B}_i(t)$ respectively, which are both undefined for all $t \in \mathbb{N}$ (where they change value) and 0 for all other t. Thus, we can reduce the system to one in which the weights are constant for the duration of a slot. At the slot boundaries $t \in \mathbb{N}$ the function $F(\boldsymbol{X}(t))$ possibly makes a jump, due to the weights changing. However this does not impact the queue sizes themselves. We have thus that $\dot{F}(\boldsymbol{X}(t)) \le 0$, making the system stable. □

As before, we can rewrite the equation to obtain

$$F(\boldsymbol{X}(t)) \le \left(F(\boldsymbol{X}(0))^{\frac{1}{\beta}} - \eta\sum_{s=0}^{t}\min(\boldsymbol{A}(s))^{\frac{\beta-1}{\beta}}\right)^\beta$$

where $\eta = \epsilon\sqrt[\beta]{\frac{\min_i(\mu_i^{-1}(\lambda_i+\zeta)^{-\alpha})}{\beta}}$. This implies that the smaller $\min(\boldsymbol{A}(s))$ is with respect to $\max(\boldsymbol{A}(s))$, the longer it can take to reduce all queues. If $\min(\boldsymbol{A}(s)) = \max(\boldsymbol{A}(s)), \forall s$, then the result is reduced to the previous theorem, as A is scaling-independent.

REFERENCES

[1] Q. Wu, X. Fan, W. Wei, and M. Wozniak, "Dynamic scheduling algorithm for delay-sensitive vehicular safety applications in cellular network," Information Technology and Control, vol. 49, no. 1, pp. 161–178, 2020.

[2] Y. Wang and W. Chen, "Minimizing Delay Violation Probability in URLLC over Fading Channels: A Cross-Layer Approach," in GLOBECOM 2020-2020 IEEE Global Communications Conference. IEEE, 2020, pp. 1–6.

[3] M. Wang, J. Liu, W. Chen, and A. Ephremides, "Joint queue-aware and channel-aware delay optimal scheduling of arbitrarily bursty traffic over multi-state time-varying channels," IEEE Transactions on Communications, vol. 67, no. 1, pp. 503–517, 2018.

[4] J. Van den Eynde, J. Verdyck, M. Moonen, and C. Blondia, "A delay-based cross-layer scheduler for adaptive DSL," in Communications (ICC), 2017 IEEE International Conference on. IEEE, 2017, pp. 1–6.

[5] 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.

[6] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting, "Scheduling in a queuing system with asynchronously varying service rates," Probability in the Engineering and Informational Sciences, vol. 18, no. 02, p. 191—217, 2004.

[7] 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.

[8] R. Basukala, H. M. Ramli, and K. Sandrasegaran, "Performance analysis of EXP/PF and M-LWDF in downlink 3GPP LTE system," in 2009 First Asian Himalayas International Conference on Internet. IEEE, 2009, pp. 1–5.

[9] S. Ryu, B. Ryu, H. Seo, and M. Shin, "Urgency and efficiency based packet scheduling algorithm for OFDMA wireless system," in IEEE International Conference on Communications, 2005. ICC 2005. 2005, vol. 4. IEEE, 2005, pp. 2779–2785.

[10] M. Katoozian, K. Navaie, and H. Yanikomeroglu, "Utility-based adaptive radio resource allocation in OFDM wireless networks with traffic prioritization," IEEE Transactions on Wireless Communications, vol. 8, no. 1, pp. 66–71, 2009.

[11] A. Sharifian, "Utility-based Packet Scheduling and Resource Allocation Algorithms with Heterogeneous Trac for Wireless OFDMA Networks," Ph.D. dissertation, Carleton University Ottawa, 2014.

[12] P. Semov, P. Koleva, and V. Poulkov, "Adaptive Resource Scheduling based on Neural Network and Mobile Traffic Prediction," in 2019 42nd International Conference on Telecommunications and Signal Processing (TSP). IEEE, 2019, pp. 585–588.

[13] R. P. Antonioli, E. B. Rodrigues, T. F. Maciel, D. A. Sousa, and F. R. Cavalcanti, "Adaptive resource allocation framework for user satisfaction maximization in multi-service wireless networks," Telecommunication Systems, vol. 68, no. 2, pp. 259–275, 2018.

[14] S. Martiradonna, A. Grassi, G. Piro, and G. Boggia, "5G-air-simulator: An open-source tool modeling the 5G air interface," Computer Networks, vol. 173, p. 107151, 2020.

[15] M. I. Elhadad, M. Abd-Elnaby, and E.-S. M. El-Rabaie, "Optimized delay threshold scheduler for multimedia traffic over LTE downlink network," Multimedia Tools and Applications, vol. 78, no. 11, pp. 15 507–15 525, 2019.

[16] M. A. Lawal, I. Saidu, A. Mohammed, and Y. A. Sade, "Downlink scheduling algorithms in LTE Networks: A survey," IOSR J Mob Comput Appl, vol. 4, no. 3, pp. 1–12, 2017.

[17] M. M. Nasralla, N. Khan, and M. G. Martini, "Content-aware downlink scheduling for LTE wireless systems: A survey and performance comparison of key approaches," Computer Communications, vol. 130, pp. 78–100, 2018.

[18] 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.

[19] A. Mekkittikul and N. McKeown, "A starvation-free algorithm for achieving 100% throughput in an input-queued switch," in Proc. of the IEEE International Conference on Communication Networks. Citeseer, 1996.

[20] 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.
[21] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, P. Whiting, and R. Vijayakumar, "Providing quality of service over a shared wireless link," IEEE Communications magazine, vol. 39, no. 2, pp. 150–154, 2001.
[22] M. M. Nasralla and M. G. Martini, "A downlink scheduling approach for balancing QoS in LTE wireless networks," in 2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC). IEEE, 2013, pp. 1571–1575.
[23] S. Xulu and G. Aiyetoro, "Cross-layer design approach based packet scheduling in next generation wireless networks," in 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC). IEEE, 2018, pp. 757–761.
[24] J. Yaqoob, W. L. Pang, S. K. Wong, and K. Y. Chan, "Enhanced exponential rule scheduling algorithm for real-time traffic in LTE network." International Journal of Electrical & Computer Engineering (2088-8708), vol. 10, no. 2, 2020.
[25] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," IEEE/ACM Transactions on networking, no. 5, pp. 556–567, 2000.
[26] G. Goodwin and K. Sin, "Adaptive filtering prediction and control. 1984," Englewood Clifs: Prentice Ha II, 1984.
[27] 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.
[28] A. Kumar, A. Abdelhadi, and T. C. Clancy, "Delay-Efficient Multiclass Packet Scheduler," in Design and Implementation of Practical Schedulers for M2M Uplink Networks. Springer, 2018, pp. 15–80.
[29] A. Aguiar, A. Wolisz, and H. Lederer, "Utility-based Packet Scheduler for Wireless Communications," Proceedings. 2006 31st IEEE Conference on Local Computer Networks, pp. 863–870, 2006.
[30] F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, "Downlink Packet Scheduling in LTE Cellular Networks: Key Design Issues and a Survey," IEEE Communications Surveys & Tutorials, vol. 15, pp. 678–700, 2013.
[31] J. Verdyck and M. Moonen, "Dynamic Spectrum Management in Digital Subscriber Line Networks With Unequal Error Protection Requirements," IEEE Access, vol. 5, pp. 18 107–18 120, 2017.
[32] B. Sadiq and G. De Veciana, "Throughput optimality of delay-driven MaxWeight scheduler for a wireless system with flow dynamics," in 2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton). IEEE, 2009, pp. 1097–1102.
[33] T. Bonald and L. Massoulié, "Impact of fairness on Internet performance," in ACM SIGMETRICS Performance Evaluation Review, vol. 29, no. 1. ACM, 2001, pp. 82–91.
[34] T. Bonald and A. Proutière, "Flow-level stability of utility-based allocations for non-convex rate regions," in 2006 40th Annual Conference on Information Sciences and Systems. IEEE, 2006, pp. 327–332.
[35] J. Liu, A. Proutière, Y. Yi, M. Chiang, and H. V. Poor, "Flow-level stability of data networks with non-convex and time-varying rate regions," in Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, 2007, pp. 239–250.
[36] W. Luo and A. Ephremides, "Stability of N interacting queues in random-access systems," IEEE Transactions on Information Theory, vol. 45, no. 5, pp. 1579–1587, 1999.
[37] J. Verdyck, C. Blondia, and M. Moonen, "Network Utility Maximization for Adaptive Resource Allocation in DSL Systems," in 2018 26th European Signal Processing Conference (EUSIPCO). IEEE, 2018, pp. 787–791.
[38] P. Silverman, "Techniques to mitigate uncancelled crosstalk on vectored vdsl2 lines," in Broadband Forum Technical Report: TR-320, 2014.
[39] Z.-Q. Luo and S. Zhang, "Dynamic spectrum management: Complexity and duality," IEEE journal of selected topics in signal processing, vol. 2, no. 1, pp. 57–73, 2008.
[40] P. Tsiaflakis, Y. Lefevre, W. Coomans, and J. Maes, "Friendly full duplex: A multi-user full duplex method for MGfast in coexistence with G. fast," in 2018 IEEE Global Communications Conference (GLOBECOM). IEEE, 2018, pp. 1–6.
[41] J. Bezerra, A. Klautau, M. Monteiro, E. Pelaes, E. Medeiros, and B. Dortschy, "An evolutionary algorithm for improved diversity in DSL spectrum balancing solutions," EURASIP Journal on Advances in Signal Processing, vol. 2010, no. 1, p. 513610, 2010.
[42] A. Gomes, M. Monteiro, B. Dortschy, and A. Klautau, "An hybrid evolutionary multiobjective algorithm for multiuser margin maximization in DSL," International Journal of Communication Systems, vol. 29, no. 1, pp. 194–209, 2016.
[43] M. Wolkerstorfer and D. Statovci, "Fairness tradeoff analysis during VDSL2 network migration-Simulation and testbed results," in 2014 IEEE Symposium on Computers and Communications (ISCC). IEEE, 2014, pp. 1–6.
[44] TNO, "G.fast: Release of measured transfer characteristics of the 104m KPN access cable," ITU - Telecommunication Standardization Sector, Tech. Rep., Mar. 2013.
[45] I. Angri, M. Mahfoudi, A. Najid, and M. El Bekkali, "Exponential MLWDF (EXP-MLWDF) Downlink Scheduling Algorithm Evaluated in LTE for High Mobility and Dense Area Scenario," International Journal of Electrical and Computer Engineering, vol. 8, no. 3, p. 1618, 2018.
[46] J.-H. Rhee, J. M. Holtzman, and D.-K. Kim, "Scheduling of real/non-real time services: adaptive EXP/PF algorithm," in The 57th IEEE Semiannual Vehicular Technology Conference, 2003. VTC 2003-Spring., vol. 1. IEEE, 2003, pp. 462–466.
[47] H. Lei, L. Zhang, X. Zhang, and D. Yang, "A packet scheduling algorithm using utility function for mixed services in the downlink of OFDMA systems," in 2007 IEEE 66th Vehicular Technology Conference. IEEE, 2007, pp. 1664–1668.
[48] G. Aiyetoro and F. Takawira, "Joint User Scheduling and PRB Mapping Scheme in Satellite LTE Networks," in 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC). IEEE, 2018, pp. 24–29.
[49] O. Bello, H. Zen, A.-K. Othman, and K. A. Hamid, "Efficient and low-complexity scheduling algorithm in a multi-user heterogeneous traffic scenario," in 2015 IEEE 12th Malaysia International Conference on Communications (MICC). IEEE, 2015, pp. 201–206.
[50] M. S. Taqqu, W. Willinger, and R. Sherman, "Proof of a fundamental result in self-similar traffic modeling," ACM SIGCOMM Computer Communication Review, vol. 27, no. 2, pp. 5–23, 1997.
[51] "MPEG traces archive," http://web.archive.org/web/20080916125231/http://www-info3.informatik.uni-wuerzburg.de/mpeg/traces/, accessed: 2020-02-27.
[52] J. G. Dai et al., "On positive Harris recurrence of multiclass queueing networks: a unified approach via fluid limit models," The Annals of Applied Probability, vol. 5, no. 1, pp. 49–77, 1995.



JEREMY VAN DEN EYNDE is a Ph.D. researcher associated with imec and the University of Antwerp, Belgium. He received a M.S. degree in computer science from the University of Antwerp in 2009. He joined the PATS/IDLab research group at the University of Antwerp in 2011 to do research in modeling of network systems under supervision of Prof. C. Blondia. His current research focuses on quality of service, cross-layer optimization and queuing theory.



JEROEN VERDYCK (S'15) received the M.Sc.degree in electrical engineering from KULeuven, Leuven, Belgium, in 2014, where he is currently pursuing the Ph.D. degree with the Electrical Engineering Department, under the supervision of Prof. M. Moonen. He is involved in joint projects with KU Leuven and the University of Antwerp, Antwerp, Belgium. His research interests include signal processing and optimization for digital communication systems with an emphasis on DSL wireline access networks.



CHRIS BLONDIA obtained his Master in Science and Ph.D. in Mathematics, both from the University of Ghent (Belgium) in 1977 and 1982 respectively. In 1983 he joined Philips Belgium, where he was a researcher between 1986 and 1991 in the Philips Research Laboratory Belgium (PRLB) in the group Computer and Communication Systems. Between August 1991 and end 1994 he was an Associate Professor in the Computer Science Department of the University of Nijmegen (The Netherlands). In 1995 he joined the Department of Mathematics and Computer Science of the University of Antwerp, where he is currently a Full Professor in the Internet and Data Lab (IDLab) research group. He has been the Chair of the Department of Mathematics and Computer Science between 2010 and 2016. He is lecturing networking courses. His main research interests are related to the design, analysis, and implementation of algorithms and protocols for communication networks, in particular, wireless networks, focusing on their performance. He has published over 250 papers in international journals and conferences on these research areas. He has been involved in many national and European Research Programs. The IDLab research group is also a core group of the Flemish Strategic Research Center imec (www.imec.be).



MARC MOONEN (M'94–SM'06–F'07) is currently a Full Professor with the Electrical Engineering Department, KU Leuven, where he is also the Head of the research team involved in the area of numerical algorithms and signal processing for digital communications, wireless communications, DSL, and audio signal processing. Dr. Moonen was the Chairman of the IEEEBenelux Signal Processing Chapter from 1998 to 2002, a member of the IEEE Signal Processing Society Technical Committee on Signal Processing for Communications, and the President of EURASIP (2007–2008 and 2011–2012). He is currently a member of the Editorial Board of the *EURASIP Journal on Advances in Signal Processing*. He was a recipient of the 1994 KU Leuven Research Council Award, the 1997 Alcatel Bell (Belgium) Award (with P. Vandaele), and the 2004 Alcatel Bell (Belgium) Award (with R. Cendrillon). He was a 1997 Laureate of the Belgium Royal Academy of Science. He received best paper awards from the IEEE Transactions on Signal Processing (with G. Leus and D. Giacobello) and the *Signal Processing* (with S. Doclo). He has served as an Editor-in-Chief of the *EURASIP Journal on Applied Signal Processing* from 2003 to 2005 and an Area Editor for feature articles in the *IEEE Signal Processing Magazine* from 2012 to 2014. He has been a member of the Editorial Board of the IEEE Transactions on Circuits and Systems II, the *IEEE Signal Processing Magazine*, the *Integration*, the VLSI journal, the *EURASIP Journal on Wireless Communications and Networking*, and the *Signal Processing*.