Evaluation of a strict priority scheduler, and cross-layer resource allocation

Proefschrift voorgelegd tot het behalen van de graad van doctor in de wetenschappen aan de Universiteit Antwerpen te verdedigen door

Jeremy Van den Eynde

University of Antwerp - imec
IDLab - Department of Computer Science

Promotoren
Prof. dr. Chris Blondia
Prof. dr. Marc Moonen

Antwerpen, 2022

Universiteit Antwerpen
Faculteit Wetenschappen

Universiteit Antwerpen
Faculteit Wetenschappen

Faculteit Wetenschappen

Evaluation of a strict priority scheduler, and cross-layer resource allocation

Proefschrift voorgelegd tot het behalen van de graad van
doctor in de wetenschappen
aan de Universiteit Antwerpen te verdedigen door

Jeremy Van den Eynde

Promotoren
Prof. dr. Chris Blondia
Prof. dr. Marc Moonen

Antwerpen, 2022

Contact

Jeremy Van den Eynde
University of Antwerp - imec
IDLab - Department of Computer Science
Sint-Pietersvliet 7, 2020 Antwerpen, België
M: jeremy.vandeneynde@uantwerpen.be

Acknowledgements

Here, I would like to thank those who have helped and supported me directly and indirectly in finishing the long endeavor that's resulted in this thesis. The list of people to thank is bound to be incomplete, as so many have guided me during this process.

First and foremost, I am deeply grateful to my supervisor Prof. dr. Chris Blondia who gave me the opportunity and the time to start and – more importantly – finish my PhD, and has advised me in this journey. Whenever we met, his insightful comments gave me new energy and ideas to continue. I would also like to thank my external co-supervisor Prof. dr. Marc Moonen for the collaboration, insights and interesting projects.

In addition to my supervisors, I would like to thank the honourable members of the jury: Prof. dr. Benny Van Houdt, Prof. dr. Steven Latré and Prof. dr. Dieter Fiems for taking the time to carefully read and discuss my thesis in order to improve it.

Furthermore, I'm grateful for the guidance and support of dr. Kathleen Spaey during my initial years. She and Prof. dr. Chris Blondia teamed up to introduce me into the world of analysis of scheduling algorithms. I also want to thank dr. Jeroen Verdyck for our collaboration and valuable feedback over the years.

A big thank you to Prof. dr. Jeroen Famaey for taking me into his team as one of his own, even though I was the weird duckling, working on DSL projects while all others were working on wireless and IoT networks and mmWave communications.

Life at university would have been dull without the company of the many colleagues with whom I had interesting discussions and laughs during the tea/coffee breaks, and card, kubb or ping pong games during our lunches. Thank you Bart, Daniel, Johan, Nico, Niko, Serena, Jakob, Filip, and many others. I also fondly look back to the bike rides home with Bart Sas!

Many people kept their curiosity about my research, despite my often abstract and unclear answers, and helped me unload my mind in various ways. In particular, I'd like to mention my brother and sister, Jixop, Moorkens, Wouter!, Bobby, Frady, Sven, Julia, Esteban, and many others who provided me with the often necessary distractions, whether it was climbing, music, movies, drinks, games, travels or foods.

Finally, I would like to thank my parents for providing me with an always welcoming, supportive and warm place.

Abstract

The aim of this thesis is twofold: (I) the analysis of the end-to-end delay and delay variation of a strict priority scheduler for a particular combination of traffic inputs, and (II) cross-layer allocation of resources in shared systems.

In industries like railroad and power companies, old and proprietary computer networks are being replaced with common and open standards. The critical traffic that runs on these old networks has very strict quality of service (QoS) requirements for critical traffic. These requirements should also be respected in the upgraded networks, which now is also used for other traffic, such as closed-circuit television (CCTV), voice over IP (VoIP) and information systems.

In part (I) of this thesis, we develop expressions for the end-to-end (E2E) delay and delay variation distributions, for the different classes of aggregate traffic that are served by a strict priority scheduler. This can help to dimension the network and ensure the QoS is not violated.

We characterize the busy period, taking low priority traffic into account, of all priorities in order to calculate the additional delay each traffic class encounters. In particular, we characterize the busy period of any aggregate of constant bit-rate (CBR) sources. We use those distributions to obtain the E2E delay bound, taking the through-traffic and cross-traffic (CT) into account. Methods used in literature are usually limited to two priority classes or do not account for the through-traffic.

We evaluated our approach using simulation of a network, and found that our expressions are able to upper bound the E2E delay and delay variation for all the traffic priorities that we have considered here.

In part (II) we look at cross-layer resource allocation. The Open Systems Interconnection (OSI) model is based on layers that provide an abstraction of a certain networking function. These abstractions allow very limited interaction between the layers, which results in missed opportunities to increase the performance of the network. Cross-layer algorithms make use of information contained in layers it can normally not access, in order to achieve a better performance.

The research presented here applies a novel cross-layer scheduler, called the minimal delay violation (MDV) scheduler, that can be used when the achievable data rate of a user depends on the data rates the other users receive. The MDV scheduler takes the state of the users into account, such as the delay and number of dropped packets, to calculate a weight that expresses the rate requirements for each user. First we consider a digital subscriber line (DSL) context, where cross-talk between the copper cables of users reduces the maximum simultaneously achievable data rates. The MDV scheduler is then applied to an long-term evolution (LTE) and 5G (5G) setting.

Through simulations, we show that the MDV scheduler outperforms other schedulers from literature in a multitude of scenarios, with respect to the delay and throughput. We additionally discuss some properties of the scheduler, such as its stability. To constrain misbehaving flows, we design a throughput constraining algorithm that can be applied to multiple cross-layers schedulers from literature, with minimal changes.

We also implement one other cross-layer allocation algorithm that can be used when service rates are assigned dynamically, but there is a delay between requesting and receiving the data rate, such as for example in satellite communication networks. Here we modify an admission control algorithm to provide a prediction on a traffic aggregate. We evaluate the algorithm using simulation for different traffic mixes, and protocols, and find that the offered trade-off between efficiency and adherence to the QoS is reasonable, compared to an ideal situation.

Nederlandse samenvatting

Deze thesis omvat twee delen: (I) de analyse van de E2E vertraging en variatie van de vertraging van pakketten in een stricte prioriteitscheduler voor een bepaalde combinatie van verkeer, en (II) de cross-layer allocatie in gedeelde netwerken.

In industriën zoals de spoorwegen en het electriciteitsnet worden oude en gepatenteerde computer netwerken vervangen door netwerken gebaseerd op open standaarden. Het belangrijke verkeer dat over deze oude netwerken loopt heeft hele strikte QoS vereisten. Deze vereisten moeten ook gerespecteerd worden in de geüpgrade netwerken, die nu ook voor ander verkeer gebruikt kan worden, zoals CCTV, VoIP en informatie systemen.

In deel (I) van deze thesis ontwikkelen we uitdrukkingen voor de kansverdelingsfunctie van de E2E vertraging en variatie van de vertraging voor de verschillende klasses van geaggregeerd verkeer dat door een strikte prioriteitsscheduler wordt verwerkt. Dit kan helpen om het netwerk te dimensioneren en er voor te zorgen dat de QoS gerespecteerd wordt.

We karakteriseren de *busy period*, de tijd dat bepaalde prioriteiten verstuurd worden, van alle prioriteiten om zo de extra vertraging te berekenen dat elke verkeersklasse ondervindt. Bij het berekenen van de busy period wordt steeds ook het lage prioriteitsverkeer in rekening gebracht. Hier merken we op dat we de busy period voor een aggregaat van verschillende CBR bronnen een nauwkeurige verdelingsfunctie hebben bepaald. Deze verdelingsfuncties worden vervolgens gebruikt om een grens op E2E vertraging en variatie van de vertraging te bepalen. Belangrijk hierbij is op te merken dat zowel het tijdelijk verkeer als het verkeer dat over verschillende nodes wordt verstuurd (het through-traffic), in rekening is gebracht. De methodes die gebruikt worden in de literatuur zijn gewoonlijk

beperkt tot twee verkeersklassen, of houden geen rekening met through-traffic.

We hebben onze uitdrukkingen geëvalueerd aan de hand van simulaties van een netwerk. Daaruit bleek dat we voor alle verkeersklassen die we hier beschouwen, een bovengrens op de E2E vertraging en variatie van de vertraging konden bepalen.

In deel (II) kijken we naar de cross-layer toekenning van hulpbronnen. Het OSI model is gebaseerd op lagen die een abstractie voorzien voor bepaalde netwerk functionaliteit. Deze abstracties laten slechts een zeer beperkte wisselwerking toe tussen de verschillende lagen, wat er voor zorgt dat er vele kansen tot optimalisatie verloren gaan. Cross-layer algoritmes maken gebruik van de informatie in de andere lagen die normaal niet beschikaar is, om zo tot een betere prestatie te komen.

Het onderzoek dat we hier presenteren toont een nieuwe cross-layer scheduler, genaamd de MDV scheduler, die kan gebruikt worden wanneer de data rate van een gebruiker afhangt van hoeveel data rate andere gebruikers hebben gekregen. De MDV scheduler neemt de staat van al de gebruikers in overweging, zoals de vertraging en de hoeveelheid verloren pakketten, om een gewicht te berekenen dat de data rate vereisten uitdrukt voor elke gebruiker. Eerst kijken we naar een DSL context, waar cross-talk tussen de koperkabels van de gebruikers ervoor kan zorgen dat niet alle gebruikers hun maximale data rate kunnen behalen. De MDV scheduler is vervolgens ook nog toegepast op een LTE en 5G netwerk.

Met behulp van simulaties tonen we dat de MDV scheduler betere resultaten voorlegt dan andere gelijkaardige schedulers in verschillende scenarios. We hebben hier voornamelijk naar vertraging en throughput gekeken. Daarnaast hebben we ook enkele eigenschappen van de scheduler besproken, zoals de stabiliteit. Om te voorkomen dat applicaties zich misdragen, hebben we een algoritme ontwikkeld dat de throughput van applicaties in een bepaalde range brengt. Het algoritme is toepasbaar op verschillende cross-layer schedulers uit de literatuur met minimale aanpassingen.

Verder hebben we ook een cross-layer algoritme geïmplementeerd dat kan gebruikt worden wanneer de data rates dynamisch worden toegekend, maar er een vertraging is tussen de aanvraag en het toekennen van de data rate, zoals bijvoorbeeld kan optreden in satelliet communicatie netwerken. We gebruiken een admissie controle algoritme en wijzigen dit, zodoende een voorspelling op een verkeersaggregaat te krijgen. We evalueren dit algoritme met behulp van simulaties voor verschillende combinaties van verkeer. Vergeleken met een optimaal algoritme functioneert het algoritme redelijk, als we kijken naar de trade-off tussen QoS en efficiëntie van het kanaal.

Contents

List of Figures

List of Tables

Acronyms

3g2u 3 groups, 2 users per group.
3GPP 3rd Generation Partnership Project.
4GBB fourth generation broadband access.
5G 5G.
5G NR 5G New Radio.
5GBB fifth generation broadband access.
AC admission control.
AC arrival curve.
ADSL asymmetric digital subscriber line.
AFDX avionics full duplex switched ethernet.
BE best-effort.
BP busy period.
CBR constant bit-rate.
ccdf complementary cumulative distribution function.
CCTV closed-circuit television.
cdf cumulative distribution function.
CO central office.
CQI channel quality indicator.
CT cross-traffic.
DBA dynamic bandwidth allocation.
DC departure curve.
DiffServ differentiated services.
DMT discrete multitone.
DMW delay-based max-weight.
DNC deterministic network calculus.
DoD Department of Defense.
DPU distribution point unit.
DS differentiated services.
DSCP differentiated services code point.
DSL digital subscriber line.
DSLAM Digital subscriber Line Access Multiplexer.
DSM dynamic spectrum management.
eNB eNodeB.
EPC evolved packet core.
E-UTRAN evolved UMTS terrestrial radio access network.
E2E end-to-end.
EBB exponentially bounded burstiness.
EDF earliest deadline first.

EMA exponentially moving average.
EXP/PF EXP/PF.
FIFO first in first out.
FLS FLS.
FR1 Frequency Range 1.
FR2 Frequency Range 2.
FTTF fiber to the frontage.
FTTH fiber to the home.
G.fast G series fast access to subscriber terminals.
GEO geosynchronous earth orbit.
gNB gNodeB.
GOOSE generic object oriented substation events.
GV grouped vectoring.
HOL head-of-line.
HP high priority.
i.i.d. independent and identically distributed.
IA inter-arrival.
IETF Internet Engineering Task Force.
IntServ integrated services.
IP Internet Protocol.
IPDV instantaneous packet delay variation.
IPv4 internet protocol version 4.
IPv6 internet protocol version 6.
ISDN Integrated Services Digital Network.
ISO International Organization for Standardization.
ISRR inter-site rapid response.
ITU International Telecommunication Union.
JFI Jain's fairness index.
LLC logical link control.
LP low priority.
LTE long-term evolution.
MBAC measurement based admission control.
M-LWDF modified largest weighted delay first.
MAC media access control.
MD min-delay.
MDU maximal delay utility.
MDV minimal delay violation.
mgf moment-generating function.
MIMO multiple input/multiple output.
mmWave millimeter wave.
MP medium priority.
MSS maximum segment size.
MT max throughput.
MTU maximum transmission unit.
MW max-weight.
NC network calculus.
NLMS normalized least mean square.

NN neural network.
NRT non-real time.
NT network termination.
NUM network utility maximization.
OFDM orthogonal frequency-division multiplexing.
OFDMA orthogonal frequency-division multiple access.
ONT optical network terminal.
OSI Open Systems Interconnection.
PBOO pay bursts only once.
PDV packet delay variation.
PF proportionally fair.
pgf probability generating function.
PHB per-hop behaviour.
PLR packet loss ratio.
pmf probability mass function.
PPM packet prediction mechanism.
PPP point-to-point protocol.
PRB physical resource block.
QHMLWDF queue-HoL-M-LWDF.
QoE quality of experience.
QoS quality of service.
RAN radio access network.
RB resource block.
RFC Request for Comments.
RIPE NCC Réseaux IP Européens Network Coordination Centre.
RR round robin.
RSVP resource reservation protocol.
RT real time.
SC service curve.
SDH synchronous digital hierarchy.
SNA System Network Architecture.
SNC stochastic network calculus.
SNR signal-to-noise ratio.
SONET synchronous optical networking.
SP strict priority.
SSC stochastic service curve.
STE stochastic traffic envelope.
TB token bucket.
TBRM token bucket rate modifier.
TCP transmission control protocol.
TDM time-division multiplexing.
TE traffic envelope.
TT through-traffic.
TTI transmission time interval.
UDP user datagram protocol.
UE user equipment.
UMTS universal mobile telecommunications system.

VoIP voice over IP.

Symbols

Common symbols

$\mathbf{X}$	a **bold**-faced letter indicates a vector
$\overline{\mathbf{X}}$	average over $\mathbf{X}$
$\mathbf{X}^T$	transpose of a vector or matrix $\mathbf{X}$
$\lfloor x \rfloor$	integer part of x
$\lceil x \rceil$	smallest integer exceeding x
$(x)^+$	$\max(0, x)$
$\mathrm{Bin}(N, n, p)$	$\binom{N}{n} (p)^n (1-p)^{N-n}$
δ_x	Kronecker delta is 1 if $x = 0$ and 0 otherwise
δ_s	Dirac delta function: $\delta_s(s) = \infty$, 0 everywhere else, and $\int_{-\infty}^{\infty} \delta_s(x)dx = 1$
$\hat{X}$	upper bound on a metric X
$\check{X}$	lower bound on a metric X
$\hat{T}$	QoS packet delay upper bound
ε	QoS packet delay violation upper bound
$q[t]$	queue size (bits) at time t
$\Gamma[t]$	HOL: waiting time of the packet at the front of the queue
$A\,[s,t]$	arrivals (bits) during in the interval $[s,t]$
$a\,[s,t]$	arrivals (packets) during in the interval $[s,t]$
$E\,[s,t]$	departures (bit) during in the interval $[s,t]$
$e\,[s,t]$	departures (packets) during in the interval $[s,t]$
$TB(\rho,\sigma)$	a token bucket with rate ρ and burst size σ

Chapter 1

m	number of CBR aggregate classes
A_j	refers to the j-th CBR aggregate class ($j \in [1, m]$), specified as (N_j, L_j, D_j)
N_j	number of active sources in aggregate A_j
L_j	length of a packet in aggregate A_j
D_j	period of a source from in aggregate A_j
$\text{lcm}(\mathbf{x})$	the least common multiple of $\mathbf{x}$
G	$\text{lcm}(\mathbf{D})$
E_j	number of packets a source from A_j can send in G slots
$\rho_{x,y}$	load in an interval $[x, y]$
W	waiting time distribution
M	maximum amount of work that can arrive from all CBR sources in the first slot
Q	queue distribution
Q^v	queue distribution given a vacation of length v
Z	BP distribution
Z^v	BP given a vacation of length v
D	delay distribution
D^v	delay distribution given a vacation of length v
I	IPDV distribution
I^v	IPDV distribution given a vacation of length v
U	maximum transmission unit (MTU): largest packet that can be transmitted
p_{ON}	probability that a VoIP source is in the ON state
$\mathcal{U}_{[a,b]}$	the uniform distribution over the interval $[a, b]$
$\mathcal{L}(X)$	$\sum_{x \in X} P\{X = x\} \mathcal{U}_{[0,x]}$, where X is a distribution
$\rho^{HP,TT}$	load of the through-traffic packets that are of equal or higher priority
$\rho^{HP,CT}$	load of the cross-traffic packets that are of equal or higher priority
ρ^{HP}	$\rho^{HP,CT} + \rho^{HP,TT}$
$\rho^{LP,TT}$	load of the through-traffic packets that are of lower priority
$\rho^{LP,CT}$	load of the cross-traffic packets that are of lower priority
ρ^{LP}	$\rho^{LP,CT} + \rho^{LP,TT}$
V^{HP}	the vacation due to high priority (HP) traffic

$V^{v,HP}$	the vacation due to HP traffic, including the low priority (LP) vacation
V^{LP}	the vacation due to LP traffic
$V^{LP,TT}$	the vacation due to through-traffic (TT) LP traffic

Chapter 2

x^n	value for user n of a vector $\mathbf{x}$
x_i^n	value for a flow i of user n of a matrix $\mathbf{x}$
$\check{R}$	guaranteed service rate
$\hat{R}$	maximum possible service rate
τ	slot size
$x[t]$	value at time slot t
$\mathcal{R}$	rate region
$\mathcal{C}$	capacity region
$\mathbf{X}^*$	optimal solution of $\mathbf{X}$ among all possible options
ζ	small constant added to a rate to avoid division by zero
conv $\mathcal{A}$	convex hull of $\mathcal{A}$
$\mathscr{S}[t]$	state of the system, including its history
$D[t]$	packet delay distribution up to time t
$u(\cdot)$	utility function
N	number of users
ϕ^n	number of flows of user n
ϕ	total number of flows in the network
$R[t]$	service rate at time t
$d_p\,[s,t]$	dropped packets during in the interval $[s,t]$
p	short-term PLR
ω	weight
$\tilde{x}[t+1]$	prediction of value x at time $t+1$
$c(\cdot)$	class-dependent function
Ω	constant weight for the SAT flow

Publications

[1] Jeremy Van den Eynde and Chris Blondia. Cross-layer optimization with real-time adaptive dynamic spectrum management for fourth generation broadband access networks. In *IFIP International Conference on Autonomous Infrastructure, Management and Security*, pages 184–188. Springer, 2014.

[2] Jeremy Van den Eynde, Jeroen Verdyck, Marc Moonen, and Chris Blondia. Delay performance enhancement for 4th generation dsl networks through cross-layer optimization. In *Proc. 6th joint WIC/IEEE SP Symposium on Information Theory and Signal Processing in the Benelux*, pages 2–9, 2016.

[3] Jeremy Van den Eynde, Jeroen Verdyck, Marc Moonen, and Chris Blondia. A delay-based cross-layer scheduler for adaptive dsl. In *Communications (ICC), 2017 IEEE International Conference on*, pages 1–6. IEEE, 2017.

[4] Jeremy Van den Eynde and Chris Blondia. Measurement-based dynamic resource allocation on traffic aggregates. In *2019 IEEE International Conference on Communication, Networks and Satellite (Comnetsat) (IEEE COMNETSAT 2019)*, Makassar, Indonesia, July 2019.

[5] Jeremy Van den Eynde and Chris Blondia. Token bucket-based throughput constraining in cross-layer schedulers. In David C. Wyld Natarajan Meghanathan, editor, *6th International Conference on Computer Science, Engineering and Information*, volume 9, pages 209–219, November 23 24, 2019, Zurich, Switzerland, November 2019.

[6] Jeremy Van den Eynde and Chris Blondia. The busy period distribution of the superposition of periodic arrivals with vacation time. In *Proceedings of the 13th EAI International Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '20, page 204–207, New York, NY, USA, 2020. Association for Computing Machinery.

[7] Jeremy Van den Eynde, Jeroen Verdyck, Marc Moonen, and Chris Blondia. Minimal delay violation-based cross-layer scheduler and resource allocation for DSL networks. *IEEE Access*, 9:75905–75922, 2021.

[8] Jeremy Van den Eynde and Chris Blondia. A minimal delay violation downlink LTE scheduler. In *2021 IEEE 46th Conference on Local Computer Networks (LCN) (LCN 2021)*, Edmonton, Canada, October 2021.

[9] Jeremy Van den Eynde and Chris Blondia. Delay analysis of a multi-hop strict priority scheduler. In *2022 20th Mediterranean Communication and Computer Networking Conference (MedComNet)*, pages 94–102, Paphos, Cyprus, 2022. IEEE.

[10] Jeremy Van den Eynde and Chris Blondia. Delay analysis of a multi-hop strict priority scheduler. In *2022 20th Mediterranean Communication and Computer Networking Conference (MedComNet)*, pages 94–102. IEEE, 2022.

Chapter 1
Introduction

1.1 Context

In industries like railroad and power companies, computer networks are an integral part to the correct and safe functioning of the system. The network traffic for these critical applications on these networks need to adhere to very strict QoS requirements to ensure the safety of the system. The hardware and software for these systems, however, are increasingly more expensive to maintain, as the hardware is getting older, and the proprietary nature makes it difficult or impossible to interconnect them with other networks and use the network for additional purposes. As these systems are being upgraded to new hardware and software that uses open standards, it is also used for additional, less-critical tasks, such as monitoring (CCTV), VoIP, information systems, management etc. However, the QoS requirements of the critical traffic must still be respected. In part (I) of the thesis, we consider a SP scheduler, where the highest priorities are represented by an aggregate of CBR sources. For the next priority we consider an aggregate of on-off sources. The next priority comprises a video flow, comprising very large datagrams which become fragmented once they enter the network. The lowest priority is reserved for background traffic, whose sole purpose, in this thesis, is to disturb the packets of the higher priorities.

We derive expressions for the single node queue size, delay and delay variation, and BP distributions. In particular, we develop a closed-form expression for the BP of an aggregate of CBR sources, where the period and/or packet size can be different, and an optional vacation is taken into account. The BP distribution is used to calculate the additional delay traffic from the lower priority can encounter. We extend the single node delay and delay variation to an E2E delay and delay variation distribution. Literature usually works with only two priorities, or only considers the delay of one priority over the different nodes. The expression we developed provides an upper bound on the delay distributions for all the different priorities we considered here, and can be extended to include more priorities. Our expressions provide good bounds on both the E2E delay and delay variation when compared to the multiple scenarios we simulated in an Internet Protocol (IP) network.

For part (II), we delve into the OSI model proposed by the International Organization for Standardization (ISO). A long time ago, they redesigned computer networks by composing black boxes, called layers. These layers are building blocks that provide limited but specific functionality. Through well-defined interfaces between these layers a network stack can be easily constructed that can be tailored to specific contexts. To ensure that these layers are interchangeable, layers have a limited interface, and layers can only communicate through these interfaces with the layer above or below. However, this limitation can lead to inefficiencies as some layers are missing important information to guide functionality.

For example, the channel quality in LTE, and hence the data rate at which a device can transmit, can change drastically from one moment to the next. Therefore, a seemingly ideal solution would be to opportunistically give service to the user that experiences the best channel quality. However, not all applications are equal, and do not really need the highest service rate. For example, a user that is fetching e-mails has very different data rate and delay requirements than a user that is holding a video call. In this case, we might try to give priority to the video call. For a two user system, the solution might seem easy, however, as the number of users increases, also the complexity grows in assigning data rates “optimally”.

Likewise, in recent DSL technologies a similar problem occurs. Due to interference between different users’ copper cables, data rates negatively impact other users’ data rates. There is a need to fairly distribute the available service rate among the users. Currently, these service rates are precalculated, based on a limited number of fixed scenarios. However, the behavior of the traffic and users is dynamic, and performance of the network can be increased by taking this varying behavior into account.

In part (II), we focus on the cross-layer aspect in some contexts. We first develop a scheduler, called the MDV scheduler, that takes the user requirements into account to steer the physical layer. The scheduler takes multiple QoS parameters into account, such as the delay upper bound, the queue size, the HOL, arrival rate and loss. These data come from the different layers, breaking the boundaries between neighboring layers. We compare our algorithm using simulations in several networking contexts to other schedulers from literature, where we can observe that our scheduler outperforms those other schedulers in various scenarios. We also discuss some properties of our scheduler.

Later, we implement another cross-layer algorithm that can be used for satellite communications. Transmitting data over such a channel is expensive, hence to reduce costs, traffic is aggregated and capacity is reserved for the aggregate on a slot-by-slot basis using a request-grant mechanism. However, the large latency makes a predictive cross-layer approach necessary. We evaluate some algorithms against an ideal algorithm in a couple of different settings, and for different traffic types, and for the user datagram protocol (UDP) and transmission control protocol (TCP) protocols.

1.2 Structure of this thesis and contributions

Part I is devoted to calculating analytical single node upper bounds on the delay, delay variation, the queue size and busy period distribution for traffic aggregates in a strict priority scheduler. We start the chapter with an introduction in Chapter 2 In Chapter 3 we derive closed-form expressions for the upper bound on an aggregate of CBR flows, where the flows might encounter a server that is busy serving a lower priority packet. A CBR source sends fixed size packets with fixed inter-arrival times. The expressions can be applied to a mix of aggregates with different packet sizes and inter-arrival times, allowing one to analyze different priorities of CBR aggregates. In Chapter 4 we provide bounds on an aggregate of on-off sources, an approximation to an aggregate of VoIP sources. We look at the bound using network calculus (NC) and an approximation using a Poisson model. Chapter 5 takes a look at fragmentation, and how it has an impact on the delay of the reassembled packet, using a video traffic source as an example. In Section 6.1 we look at the construction of the E2E delay when a flow crosses multiple nodes. In each node, a packet can encounter additional delays due to cross-traffic or traffic that is flowing in the same direction, and we must take this into account. We look at a simple method, an approximation that is valid in some scenarios, and a more computationally complex algorithm that can handle more scenarios. In Chapter 6 we simulate an IP network, and compare the E2E delay and E2E IPDV of the simulations with the calculated metrics. We then end the chapter with conclusions in Chapter 7.

Part II concerns cross-layer algorithms. We first start with an introduction to cross-layering in Chapter 8. In Chapter 9 we introduce the MDV scheduler, a cross-layer scheduler that optimizes in function of system throughput and the QoS requirements. This cross-layer scheduler is shown through simulations to perform well in both the DSL (Chapter 9), and LTE and 5G (Chapter 10) settings. In Chapter 11 we introduce a cross-layer resource allocation mechanism for use in satellite communication networks. Chapter 12 closes the chapter with a generic algorithm for cross-layer schedulers that limits misbehaving flows from impacting well-behaving flows, which can be used to enforce traffic arrival bounds, specified upon admission control.

We close this thesis in Chapter 13.

The contributions of this thesis are as follows:

- in Chapter 3 we give an accurate characterization of the BP of an aggregate of CBR sources, taking an idle period of the scheduler into account. This is then used to provide tight bounds on the delays and delay variations of the traffic; This is used to characterize the delay of the different priorities in the SP scheduler;

- in Section 6.1 we give an algorithm to calculate bounds on the E2E delay, taking both the low priority and high priority through-traffic into account;
- in Part II we introduce novel cross-layer algorithms: in Chapter 8 and Chapter 9 we discuss the MDV cross-layer scheduler for a DSL, LTE and 5G context, which outperforms cross-layer schedulers from literature;
- in Chapter 11 we introduce a cross-layer algorithm for use in satellite context;
- in Chapter 12 we develop an algorithm that constrains assigned data rate in a cross-layer scheduler.

Part I

Analysis of a strict priority scheduler

Chapter 2 Introduction

Utility companies have traditionally used time-division multiplexing (TDM) networks like synchronous digital hierarchy (SDH) and synchronous optical networking (SONET). These networks provide the high reliability and strict QoS requirements that are necessary to ensure that important messages arrive in a timely and correct manner.

For example, in a railway environment it is crucial that the messages that control the power distribution of the overhead lines (signaling) or the railroad switches arrive within a very small delay of being issued. The TDM network can guarantee the small delays. However, messaging, telephony, access control and video surveillance are also increasingly used in networks. In the past years, railway companies have moved from their decades old systems to IP backbones for voice, video and data traffic [5], for CCTV [3], or for remote signaling, traffic control and passenger information systems [1].

Likewise, in the electrical power industry there has been a transformation going on towards an adaptive smart grid that allows system operators to closely monitor, diagnose and mitigate issues. The backbones are often still based on TDM networks. However, increasingly more applications use IP over Ethernet, making the traffic more bursty. This makes the TDM network less cost-effective as the network has to be provisioned for these bursts, and makes an IP over Ethernet network attractive. For example, New Zealand installed switches to support both TDM and IP networks [6], and Beijing Electric Power Corporation increased its efficiency by installing Ethernet equipment [7].

Even though the underlying technology has changed, the arrival patterns of the stringent traffic types have remained the same. In this section, we look at the behavior of these important packets for a particular scheduler.

In IP networks there are two approaches to QoS. The first approach is integrated services (IntServ) offering fine-grained QoS: each application specifies its requirements using a resource reservation protocol (RSVP). A path is set up along all the routers on the path from the source to the destination. If this setup succeeds, then the application is guaranteed its resources. If the setup fails, there are not enough resources along the path to fulfill the requirements, and the state of the network remains the same. This has the advantage that it is easy to calculate tight bounds on metrics such as end-to-end delay and buffer size requirements. However, this requires IntServ support, not only from the routers to set-up and maintain the state for each application that requested a specific QoS, but also from each application that needs to provide the flow specifications. Furthermore, for each application state must be stored in each router, making it less scalable. Another disadvantage is that if the traffic does not stay within its spec, delays and packets might be dropped, and if the specification is too loose, applications might be disallowed that could have entered the IntServ network.

The second approach to QoS is differentiated services (DiffServ) and is much more coarse-grained. In a DiffServ network applications are aggregated in traffic classes at the edge of the DiffServ domain. Inside the DiffServ network traffic is classified and serviced based on its class only, i.e. there is no notion of an individual application, only packets belonging to a traffic class. Compared to IntServ this reduces the complexity of the applications and routers inside the DiffServ network as there is no set-up and reservation to perform along the routes nor any state to maintain. The cost, however, is that analysis of end-to-end delay of individual applications is impossible, and we have to resort to the characteristics of the aggregate itself. For example, if there are 2 VoIP calls belonging to the same traffic class going over a single DiffServ node it might be that one VoIP call has a very high E2E as it has traversed a high number of links already, while the other VoIP call has just entered the network. Ideally the first call receives preferential treatment. However, as it belongs to the same traffic class, it might be that the packet of the second call is transmitted first, for example because it arrived earlier in the queue.

In this chapter we provide expressions for the single node queue size, delay, IPDV and BP for particular inputs to a strict priority scheduler inside a DiffServ network. Then we extend the delay and IPDV to the E2E delay and IPDV. So we now take a closer look at a single node inside DiffServ network.

A packet that enters a DiffServ domain is subjected to a marker and conditioner. The marker establishes the QoS a packet should get from the network, and marked accordingly. This marker typically marks a packet based on the IP source and destination address, and the source and destination port, but any of the packet's properties can be used. The marker is set in IP's differentiated services (DS) field, occupying the 6-bit differentiated services code point (DSCP). The conditioner then can reduce the incoming traffic by applying policers or shapers, according to some traffic profile. A traffic meter counts packets or bits over an interval and determines whether a new packet exceeds the traffic profile. A policer will typically drop an out-of-profile packet (deterministic or probabilistic), or mark it out-of-profile. A policer can use multiple traffic meters to provide actions (e.g. mark out-of-profile for small violations of the traffic profile, and drop immediately for the severe violations). A shaper on the other hand will delay out-of-profile packets and send them once the traffic profile allows it.

Table 2.1: IETF RFC 4594 recommendations

Service class	DSCP	Conditioning at DS edge	PHB	Queuing	AQM
Network control	CS6 (48)	See section 3.1	RFC 2474	Rate	Yes
Telephony	EF (46)	Police using sr+bs	RFC 3246	Priority	No
Signaling	CS5 (40)	Police using sr+bs	RFC 2474	Rate	No
Multimedia conferencing	AF4x (3x)	Marker (RFC 2698)	RFC 2597	Rate	Yes per DSCP
Real-time interactive	CS4 (32)	Police using sr+bs	RFC 2474	Rate	No
Multimedia streaming	AF3x (2x)	Marker (RFC 2698)	RFC 2597	Rate	Yes per DSCP
Broadcast video	CS3 (24)	Police using sr+bs	RFC 2474	Rate	No
Low-latency data	AF2x (1x)	Marker (RFC 2698)	RFC 2597	Rate	Yes per DSCP
OAM	CS2 (16)	Police using sr+bs	RFC 2474	Rate	Yes
High-throughput data	AF1x (1x)	Marker (RFC 2698)	RFC 2597	Rate	Yes per DSCP
Standard	DF (0)	Not applicable	RFC 2474	Rate	Yes
Low-priority data	CS1 (8)	Not applicable	RFC 3662	Rate	Yes

Once a packet is inside the DiffServ domain packets are not marked or policed anymore. The service a packet receives depends only on the DSCP field and the appropriate behavior for the traffic class in the static per-hop behaviour (PHB) table. In Table 2.1 we can see an example of DSCP assignments to traffic classes, and corresponding PHB table. This table is a recommendation from the IETF, specified in RFC 4594. The first column is the traffic class. The second column contains the DSCP name and associated value. In the third column we can see the recommended action to take when a packet enters the DiffServ network. The fourth column refers to the RFC that specifies the recommended behavior for the DSCP class. The fifth column indicates the scheduler to use. A priority queue schedules packets with higher priority first, while a rate queue will try to empty the queue at a specified rate. The sixth and final column describes how queues are managed when it reaches a certain threshold.

2.1 Related work

Priority scheduling remains one of the main scheduling types in networks when a mix of delay-sensitive high priority and best-effort effort packets are to be supported (see for example [44, 77, 31]).

In [176] the probability generating function (pgf) is derived for the waiting time for an individual class in a geometric batch input queue, for a preemptive scheduler.

The authors of [193] study a single node, non-preemptive priority scheduler for a general number of priority classes. They assume a slotted system, and use a traffic model that describes the number of arrivals in a slot using the pgf, and are assumed to be independent and identically distributed (i.i.d.). A Markov Chain is constructed to derive the number of packets in the system, which is then used to obtain the pgf of the packet delays.

Lee [112] present an analytic method for the delay performance in a strict priority scheduler with three traffic classes. They model each traffic class as a non-preemptive M/G/1 queue, and obtain the average waiting time for the three different traffic classes. In [168] they analyzed an M/M/1 priority queue with an arbitrary number of customer classes, with preemption. In [98] the waiting times of a non-preemptive M/M/c priority queue has been derived, when all the different classes have the same mean service time. In [191] a priority queue is analyzed for an arbitrary number of traffic classes. The packets are assumed to arrive in trains, i.e. when a session is in the ON state, one packet arrives per slot, back to back, after which the session ends, a new one to start later. The session lengths follow a general distribution. Packet delays are derived for all traffic classes, both the moments and the probability tails, by using pgfs.

There is a lot of literature studying two different users and a priority queue. For example, the authors of [123] also investigate the non-preemptive priority queue. They assume a discrete Geo/Geo/1 queue system, with two types of customers. They build a discrete time Markov Chain, which includes the priorities, the customers and other state, and use it to obtain the average queue length and average waiting time of the two types of customers, and the average busy period of the system. In [205] they analyzed the performance of an M/M/1 system and obtained the average queue length and average busy period. Madan [124] considers a non-preemptive queue in which a single server serves a high priority M/G/1 queue and a low priority M/D/1 queue. The steady state is obtained for the system queue size.

A two-class priority queue is studied in [59], where the queue is assumed to be finite, and analyzes not only delays but also packet loss.

In [192] the busy period of the different classes of a priority queue is analyzed, for the two-class discrete $M^{GI}/D/1$ queue.

In [186] the authors give explicit expressions for the waiting times for two preemptive priority queues for M/M/1 queues with different service rates.

In [66] an M/G/1 priority scheduler with preemption is analyzed, where the system can be preempted at discrete interruption points. This can occur for example when there is fragmentation of packets. Closed-form expressions are given for the mean waiting time, mean service completion interval and mean response time.

When considering the E2E delay, there are different options. In a Jackson network [90], the joint state probability of a network of queues is expressed in a product form, leading to concise notation and low-complexity solutions. Jackson networks have been extended to include multiple classes of arrival flows, and scheduling algorithms, while keeping the product-form notation [27, 99]. All this simplicity, however, comes at the cost of requiring Poisson arrival and independent service times.

In [52] an E2E analysis is performed on multiplexed exponentially bounded burstiness (EBB) traffic. The departure of the through-traffic flow is characterized, and fed into the next node. The resulting delays in each node can then be summed to yield a probabilistic E2E delay bound. In the paper it is shown that this leads to a delay bound that scales by $\mathcal{O}(H \log H)$, compared to adding per-node delay bounds which scaled by $\mathcal{O}(H^3)$, where H is the number of nodes in the network. This is the stochastic counterpart of pay bursts only once (PBOO), where a burst in some types of networks contribute to the delay only in the first node. The approach in the paper does not take priorities into account.

Another approach is to use the busy period bounds. The BP allows defining a lower bound on the service, called the service curve, a flow might receive [39, 116, 65]. This service curve can then be min-plus convoluted to obtain a service curve formulation, with a bounding function.

In [50] the accuracy of stochastic network calculus bounds is compared to known results from product-form networks. In case of a low load of cross-traffic and independence of arrivals, the bounds for the low priority through-traffic are reasonably accurate for a strict priority scheduler. When the load of cross-traffic is high, the network calculus bounds degrade significantly. They improve results and recover the results for the $M/M/1$ and $M/D/1$ queues, and improve general bounds with increasing load.

In [74] an end-to-end bound is developed for a single through-traffic flow for a general class of schedulers. The through-traffic, however, is assigned the lowest priority. All higher priorities are cross-traffic.

In [141] they have two priorities, the first high priority arrival process is a discrete time renewal process, while the low priority arrival process is a batch process. Rather than performing convolution on the single-node results, they apply the model recursively in order to obtain the E2E delay and the *jitter* distributions.

In [64] the authors make use of moment-generating functions (mgfs) to calculate E2E delay bounds for two aggregates. The advantage of the mgf is that it provides efficient multiplexing while maintaining convolutional formulas. The disadvantage, however, is that statistical independence is required.

The system of [103] comprises two queues, a real-time and backlogged, best-effort queue. The real-time traffic is assumed to be CBR traffic, which is approximated by an $M/D/1$ queue. They apply the Laplace-Stieltjes Transform to obtain bounds.

The papers referenced here (and others, like [51]) assume that the aggregate of through-traffic flows occupies a single priority. Few attention is given to a strict priority scheduler where all the priorities can carry through-traffic. In the scenarios in this chapter, we take through-traffic of higher and lower priorities into account, by considering the dependency that exist between the current node and previous node, with respect to the through-traffic. The through-traffic can introduce long delays if through-traffic packets from different priorities are back-to-back, as we will see later on in the chapter. Furthermore, papers from literature almost never discuss the variation in delay.

In [153] the authors do support multiple priorities in a strict priority scheduler that is used in a avionics full duplex switched ethernet (AFDX) network. They, however, do not take the through-traffic into account. Furthermore, in our work, the first priorities are considered CBR traffic and modeled explicitly, while in the paper they assume a general linearly bounded model. In [53], also in the context of a AFDX network, delays for multiple priorities are determined by computing the competing frames. The authors of [60] use network calculus to calculate deterministic bounds of a strict priority scheduler for a AFDX network.

2.2 Scheduling

A scheduler determines the order in which packets are transmitted. These packets are stored in queues (or come from other schedulers, in case of hierarchical schedulers) that are connected to the scheduler.

We analyze in this chapter a strict priority scheduler, like the one shown in Figure 2.1. The high priority traffic, requiring priority treatment, is connected to the first input of the scheduler. In the figure, the input at the top is the high priority traffic.

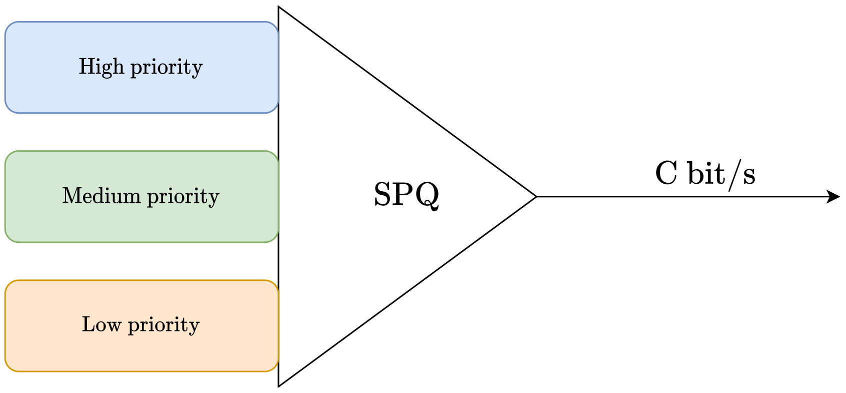


Figure 2.1: A SP scheduler with three input queues

In this chapter we consider a non-preemptive system. This means that a packet that is in transmission can not be interrupted to transmit another packet. This has implications for the strict priority scheduler: if the channel is busy, and a high priority packet arrives, it has to wait until the channel is free before it can start transmitting the high priority packet. This delay between the arrival of a packet in an empty queue and the same packet being served, is called the *vacation*.

We now describe in more detail the scheduler and refer to the corresponding sections in the text that detail them. In this chapter, the first and second queues to the strict priority scheduler are reserved for the high priority flows. These flows are assumed to be CBR traffic, i.e. comprising an aggregate of flows that periodically send packets of fixed size. The CBR traffic is typically used for applications that are critical for security or require live updates, and comprise regular probes that are forwarded. This class of traffic is treated in Chapter 3. The next priority is used for VoIP traffic, and is described in Chapter 4. We finally also briefly look at video traffic in Chapter 5, where we mainly focus on the fragmentation of large datagrams.

We look at the distributions for the following four metrics:

- **delay** the time difference between entering a node and having its last bit transmitted;
- **IPDV** the IPDV is the difference in delay of two successive packets;
- **queue** the number of packets present in the system
- **busy period** the consecutive time a queue is being served by the scheduler

2.3 Structure and contributions of this chapter

This chapter is structured as follows. We first analyze the behavior of the different priorities of traffic for a single node, in Chapter 3 for an aggregate of CBR sources, in Chapter 4 for an aggregate of VoIP sources and in Chapter 5 for a single video source emitting large datagrams that require fragmenting. Then, we revisit these results in Section 6.1, but applied to a series of nodes and in light of additional cross-traffic in each node. In Chapter 6 we evaluate our analysis using a simulation of an IP network for various scenarios, looking at the E2E and the IPDV. We close this chapter in Chapter 7 with conclusions.

The contributions of this chapter are:

- we characterize the queue size, delay and busy period of an aggregate of CBR sources using closed-form expressions. Each aggregate can comprise a different period and packet length. An optional vacation can be taken into account;
- we characterize the single node delay for multiple priorities of the strict priority scheduler;
- we take fragmentation into account when calculating delays;
- we take both high priority and low priority through-traffic packets into account when calculating the E2E delay and IPDV.

Chapter 3

High priority CBR traffic

3.1 Introduction

In this section we discuss some important distributions of the metrics relating the high priority traffic flows of the system described above.

The aforementioned high priority smart grid applications can oftentimes be modeled as CBR traffic, i.e. periodically sending constant sized packets. For example synchrophasors measure currents and voltages at a very high frequency [12, 95], and can be represented by a CBR traffic model [152]. It is of utmost importance that these packets arrive quickly at their destination such that catastrophic events can be avoided.

A single CBR source is often notated in Kendall's notation as $D/D/1$. The first D denotes constant time between arrivals in the queue, the second D indicates a constant service time, and the 1 indicates there is one channel serving the queue. In this work we employ aggregates of CBR sources, notated as $N_j * D_j/D_j/1$, meaning we have several classes of CBR sources, each class having identical parameters.

We introduce now the symbols that will be used throughout the text. The CBR aggregate we consider consists of m different classes, each class of which can be identified by $A_j = (N_j, L_j, D_j), j \in [1, m]$. We define $\mathbf{N} = [N_1, \ldots, N_m]$, $\mathbf{L} = [L_1, \ldots, L_m]$ and $\mathbf{D} = [D_1, \ldots, D_m]$. Here, N_j is the number of sources that comprises aggregate j, L_j is the length of the frame and D_j is the period with which a frame is transmitted. For convenience, the aggregates are sorted by L in ascending order, i.e. such that $L_0 = \min(\mathbf{L})$. Each source in an aggregate starts transmitting uniformly and independently distributed in the continuous interval $[0, D_j[$. Once a source has started, all future timestamps are fixed. We assume here without loss of generality that the channel has a capacity of 1 packet/time unit. A time unit is defined here as $\frac{\min(\mathbf{L})}{R}$, where R is the rate of the outgoing channel.

3.1.1 Preemption

The networks we consider in this chapter are non-preemptive. This means that once the first bit of frame is put on a wire, the whole frame must be sent before another frame can be transmitted. More urgent frames have to wait in the high priority queue until the current frame has been transmitted completely.

Due to the presence of low priority traffic and the non-preemptive character of the scheduler, a high priority packet might encounter a delay, called the vacation, as it has to wait for the transmission of the low priority packet to finish. Thus, from the point of view of the high priority traffic the scheduler is idling. This vacation must be taken into account when calculating the high priority metrics. We assume that the vacation, denoted by v, is not too large: $M + v < \min(\mathbf{D})$, where $M = \mathbf{N}\mathbf{L}^T$. This restriction avoids complications, by avoiding that a packet will encounter its successor in the queue.

In practice this is not an unreasonable assumption as high priority traffic in general has a relatively small load. Having a large high priority traffic load will result in large delays, which is exactly what we are trying to avoid by using a strict priority scheduler. Ethernet frames in general have an upper bound of 1 500 byte, but even jumbo frames, going up to 9 000 byte, usually don't pose that much of a problem over 1 Gbps links. For example, assume a vacation of 9 000 byte and 100 CBR flows each transmitting a 414 byte high priority packet every 20 ms over 1 Gbps link. This is equivalent to a CBR aggregate $A = (100, 1, \frac{1 \text{ Gbps}}{414 \text{ byte}/fr} \cdot 0.05 \text{ ms}) \approx (100, 1, 15096)$ and $v = \frac{9\,000 \text{ byte}}{414 \text{ byte}} \approx 22$. It is clear that even in this case $100 \cdot 1 + 22 < 15096$.

There is an amendment [85] to the 802.1Q VLAN standard that defines a new class of service for time-critical frames. However, as it is a recent standard, not all network devices support this service class, and thus do not support preemption.

The schedulability of a static priority scheduler for period sources has also been discussed in literature. Notably, in [118] the authors introduce the rate monotonic priority assignment for a static, preemptive priority scheduler. They show that if the priority is determined by the period D_j, with a smaller period having a higher priority, that all packets can be scheduled if it holds that

$$U = \sum_i \frac{L_i}{D_i} \leq m \cdot (2^m - 1). \tag{3.1}$$

When m goes to infinity, then U approaches $\ln 2 \approx 0.693$.

When non-preemption is taken into account, as is the case in this chapter, then the authors of [145] show that rate monotonic priority assignment is optimal for a static, non-preemptive priority scheduler when each packet's relative deadline is equal to its period.

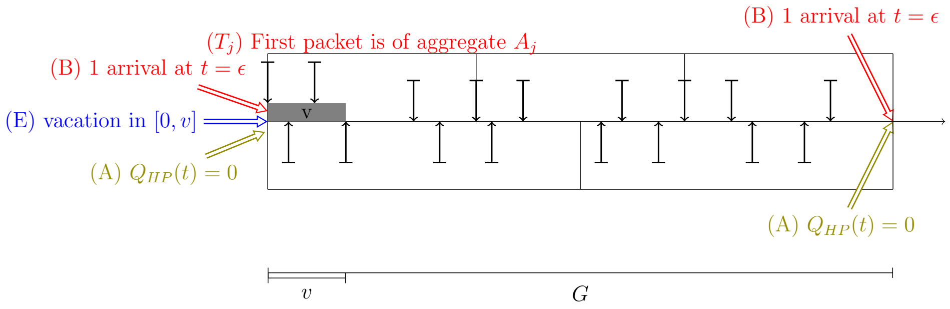

(B) 1 arrival at $t = \epsilon$
(T_j) First packet is of aggregate A_j
(B) 1 arrival at $t = \epsilon$
(E) vacation in $[0, v]$
v
(A) $Q_{HP}(t) = 0$
(A) $Q_{HP}(t) = 0$
v
G

Figure 3.1: Events characterizing two CBR aggregates, and with T_1

3.1.2 Events in the life of a CBR aggregate

In Figure 3.1 we show a typical scenario for a CBR aggregate with a vacation v. There are $m = 2$ classes, with $A_1 = (3, D_1, L_1)$ and $A_2 = (4, D_2, L_2)$. The whole figure covers 1 full period of size G. In the figure are some events indicated that are of importance. They will be referenced throughout the text, hence we list here the important ones.

(A) is the event that the HP queue is empty a time $t = 0$.

(B) indicates that there is an arrival from any of the HP sources at time ϵ.

(E) is the start of a vacation of size v. This can be either a LP frame that starts at time $t = 0$, or a frame that is in transmission when (B) happens.

(T_j) finally is the event that the frame from (B) is from aggregate class A_j.

Working with different CBR aggregates allows us to model several high level priorities. For example imagine a scenario with three different priority levels, each of which comprises different CBR aggregates, each of which serviced by a strict priority scheduler. Then the metrics calculated in the following sections, namely queue size, delay and busy period, allow us to model each level accurately.

One last point to note is that these metrics relate to the aggregate over many different sample paths. We can not say anything meaningful about a single flow over a single run. For example in a one-class aggregate each individual flow will, in the absence of vacation, always encounter the same delay due to synchronization.

3.2 Simulation setup

The simulation setup is shown in Figure 3.2. There are two queues that go into a strict priority scheduler that is served by a channel of capacity 1 packet per slot.

The slot size is the time required to send one packet with size L_1. One queue is for HP traffic, while the other one is for LP traffic. The latter queue transmits a LP packet right before the busy period of the HP traffic starts. We ran 100 000 simulations, in which each CBR flow had a different offset.

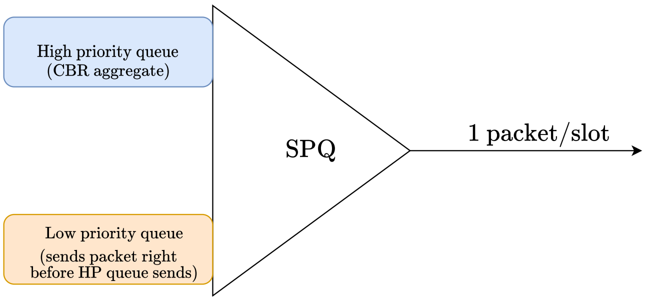

High priority queue (CBR aggregate)
SPQ
1 packet/slot
Low priority queue (sends packet right before HP queue sends)

Figure 3.2: Scheduler setup for the simulations of this section

In the plots below, the different curves indicate the maximum length of the vacation, caused by the LP packet. The length of the LP packet is chosen uniformly from $\mathbb{N} \cap \{0 \ldots v\}$. The dashed curves come from the analytical methods we provide, while the full curves are the results from the simulations.

3.3 Queue distribution

In this subsection we discuss the distribution of the queue size of the HP traffic. The queue process $Q(t)$ is the number of HP packets in the system at a time t. This includes the packets waiting in the queue itself, but also the packet that is in transmission. We use $Q_{N,D}$ (when $m = 1$, i.e. only one CBR aggregate) and $Q_{\mathbf{N},\mathbf{D},\mathbf{L}}$ (when $m > 1$) to indicate the queue process resulting from the corresponding aggregates.

3.3.1 $m = 1$

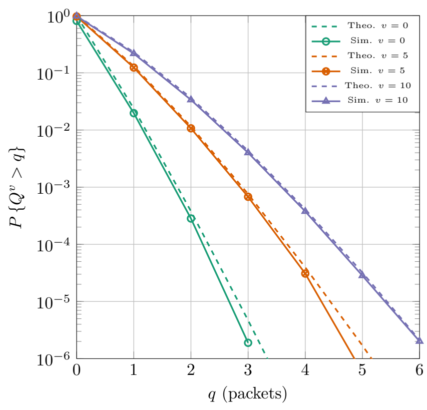
For the queue size distributions of a one-class aggregate of identical CBR sources a closed-form expression exists [84, 154]. As there is only one class, we set $N = N_1$, $D = D_1$ and $L = 1$. The distribution of the queue size is given by

$$P\{Q_{N,D} > q\} = \sum_{s=1}^{N-q} \left(1 - \frac{N-(q+s)}{D-s}\right) \cdot \mathrm{Bin}(N, q+s, \frac{s}{D}), \tag{3.2}$$

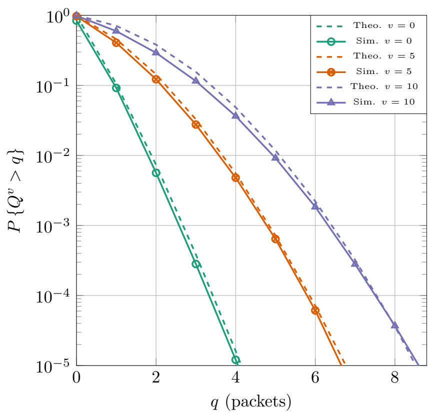
where $\mathrm{Bin}(N,k,p) = \binom{N}{k} (p)^k (1-p)^{N-k}$. This however does not take the vacation into account. We can approximate the queue size distribution with vacation as follows. Assume events (A), (B) and (E) as defined above in Section 3.1.2, and also introduce (J_l) the event that there are l arrivals during the interval $[0, v]$. Then we can condition (3.2) on (J_l):

$$\begin{aligned} P\left\{Q^v_{N,D} > q\right\} &\leq P\left\{Q^v_{N,D} > q|ABE\right\} \\ &= \sum_{l=0}^{N-1} P\left\{J_l|ABE\right\} P\left\{Q^v_{N,D} > q|ABEJ_l\right\} \\ &= \sum_{l=0}^{N-1} \text{Bin}(N-1, l, \frac{v}{D}) \cdot P\left\{Q_{N-l,D} > q - l\right\} \end{aligned} \tag{3.3}$$

On the first line, the right-hand side provides an upper bound on the left-hand side as we assume in the right-hand side that a packet arrives right after the vacation of size v has started (conditions A, B and E). The left-hand side has vacations of size v, but the arrival of a packet can occur at any time during that vacation. During the vacation the queue can only grow or remain the same, hence the earlier a packet arrives during a vacation, the larger the probability of a bigger queue.



(a) $\mathbf{A} = [(70, 1500, 1)]$



(b) $\mathbf{A} = [(15, 75, 1)]$

Figure 3.3: CBR queue distribution for two different CBR aggregates and various $v \in \{0, 5, 10\}$

Figure 3.3 shows the results for two different CBR aggregates. The x-axis shows the queue size, while the y-axis shows the distribution $P\{Q^v_{N,D} > q\}$ on a log-scale. The dashed curves come from (3.3), while the full curves are the results from 100 000 simulations. The queue size is sampled regularly, and it is this queue size distribution that is shown in the plots.

We can observe in Figure 3.3 that the queue size for $v = 0$ (i.e. the original formula from [154]) is, as expected, very close to the simulation. For vacation lengths $v = 5$ and $v = 10$, calculated according to (3.3), the results are also very close to the simulation results.

3.3.2 $m > 1$

It is possible that a single priority serves different types of HP traffic, each type being of a different CBR class. Therefore, we look here at the case when there are multiple CBR traffic classes.

In contrast to $N * D/D/1$ queue only an upper bound is provided in [154] for the $N_j * D_j/D/1$ queue. In [135] an exact distribution is given for the $N_j * D/D_j/1$ queue. We extend here the work of [154] to support different periods and a vacation period, at the cost of looser bounds. We first derive results for the upper bound to $P\{Q_{\mathbf{N},\mathbf{D},\mathbf{L}} > q\}$ for the $N_j/D_j/D_j/1$ queue, and then incorporate the vacation, using the same approach as in Section 3.3.1.

First we define

$$\alpha_{s,j} = \frac{s}{D_j} - \left\lfloor \frac{s}{D_j} \right\rfloor . \tag{3.4}$$

This $\alpha_{s,j}$ is the probability that one source from aggregate j has one extra arrival, on top of the guaranteed $\left\lfloor \frac{s}{D_j} \right\rfloor$ arrivals, in an interval of length s. Let $q_s(l)$ be the probability of a workload l being generated over an interval s if a flow can send at most 1 frame, and $M = \mathbf{N}\mathbf{L}^T$ the maximal possible consecutive work. Then $q_s(l)$ can be written as

$$q_s(l) = \begin{cases} \displaystyle\sum_{\{\mathbf{n}:\mathbf{n}\mathbf{L}^T=l\}} \left(\prod_{j=1}^{m} \mathrm{Bin}(N_j, n_j, \alpha_{s,j}) \left(1 - \frac{L_j \cdot (N_j - n_j)}{D_i \cdot (1-\alpha_{s,j})}\right)\right), & 0 \le l \le M \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

Now we can express the distribution of the excess workload at the end of an interval s in terms of $q_s(l)$:

$$p_s(q) = \begin{cases} q_s(q + s - \mathrm{LB}), & \mathrm{LB} \le q + s \le \mathrm{UB} \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

Here, $\mathrm{LB} = \sum_j N_j L_j \left\lfloor \frac{s}{D_j} \right\rfloor$ and $\mathrm{UB} = \sum_j N_j L_j \left(\left\lfloor \frac{s}{D_j} \right\rfloor + 1 \right)$. We finally can now define the upper bound on the queue distribution (without vacation) as

$$P\{Q_{\mathbf{N},\mathbf{D},\mathbf{L}} > q\} \le \sum_{s=1}^{\infty} p_s(q). \tag{3.7}$$

To also include the vacation into the queue size distribution, we have

$$\begin{aligned}P\left\{Q^v_{\mathbf{N},\mathbf{D},\mathbf{L}} > q\right\} &\leq P\left\{Q^v_{\mathbf{N},\mathbf{D},\mathbf{L}} > q|ABE\right\} \\ &= \sum_{j=1}^{m} P\left\{T_j\right\} P\left\{Q^v_{\mathbf{N},\mathbf{D},\mathbf{L}} > q|ABET_j\right\} \\ &= \sum_{j=1}^{m} \frac{N_j}{\sum N} P\left\{Q^v_{\mathbf{N},\mathbf{D},\mathbf{L}} > q|ABET_j\right\} \qquad (3.8)\end{aligned}$$

$P\{Q^v_{\mathbf{N},\mathbf{D},\mathbf{L}} > q|ABET_j\}$ is the distribution of the queue size, given that we have a vacation of duration v which starts when the first packet of class j arrives in an empty HP queue. From the point of view of the HP frame, any arrivals during the vacation period $[0, v]$ encounter a server on vacation. Therefore, we must differentiate between packets arriving in the vacation and the servicing periods. If we indicate the work arriving in the vacation period with l, then we can write $P\{Q^v_{\mathbf{N},\mathbf{D},\mathbf{L}} > q|ABET_j\}$ in function of the regular queue size distribution as

$$\begin{aligned}&P\left\{Q^v_{\mathbf{N},\mathbf{D},\mathbf{L}} > q|ABET_j\right\} \\ &= \sum_{l} P\left\{J_l|ABET_j\right\} P\left\{Q^v_{\mathbf{N},\mathbf{D},\mathbf{L}} > q|ABET_jJ_l\right\} \\ &= \sum_{l=0}^{M-L_j} \left[\sum_{\{\mathbf{k}:\mathbf{k}\mathbf{L}^T=l\}} \prod_{i=1}^{m} \mathrm{Bin}(N_i - \delta_{j-i}, k_i, \frac{v}{D_i}) P\left\{Q_{\mathbf{N}-\mathbf{k},\mathbf{D},\mathbf{L}} > q - l\right\}\right] \qquad (3.9)\end{aligned}$$

where J_l is the event of having a workload l arriving in the vacation period, and δ_x is the Kronecker delta which is 1 if $x = 0$, and 0 otherwise. Note that this reduces to (3.3) if we have only one class.

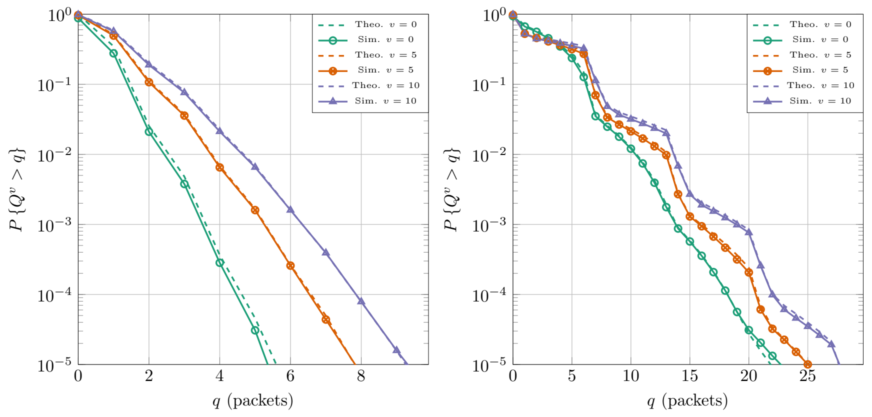
In Figure 3.4 we show the upper bound on the queue size distribution with and without vacation for $N_j/D_j/D_j/1$. We can observe that we can still accurately match the queue size distribution given by (3.3) with the simulation results.

3.4 Delay distribution

In this section we give an upper bound on $P\{D^v_{\mathbf{N}-\mathbf{k},\mathbf{D},\mathbf{L}} > d\}$, the delay distribution of an aggregate of high priority CBR flows, given that we start with a vacation v. We consider here the delay as the time between arrival in the system and the packet being completely put on the wire.

We first look at the delay distribution without vacation. The delay for a packet of size s that enters a queue of size q will have a delay of $q + s$. Thus, the delay distribution can be obtained from the queue size distribution:

$$P\{D > d\} = P\{Q > d - s\}. \qquad (3.10)$$



(a) $\mathbf{A} = [(15, 500, 1), (10, 400, 2)]$ (b) $\mathbf{A} = [(4, 150, 1), (10, 800, 7)]$

Figure 3.4: CBR queue distribution multiple aggregates and for various $v \in \{0, 5, 10\}$

However, packets can have different lengths, and as we depend on the packet length of the packet under consideration, we have to condition on the packet length. We introduce therefore a new event (F_j), the event that a packet is of class j.

The delay distribution without vacation can then be written as

$$\begin{aligned} P\{D_{\mathbf{N},\mathbf{D},\mathbf{L}} > d\} &= \sum_{j=1}^{m} P\{F_j\} P\{D_{\mathbf{N},\mathbf{D},\mathbf{L}} > d | F_j\} \\ &= \sum_{j=1}^{m} \frac{N_j/D_j}{\sum_{i=1}^{m} N_i/D_i} P\{Q_{\mathbf{N},\mathbf{D},\mathbf{L}} > d - L_j\}, \end{aligned} \tag{3.11}$$

where $P\{Q_{\mathbf{N},\mathbf{D},\mathbf{L}} > d - L_j\}$ can be obtained using (3.7).

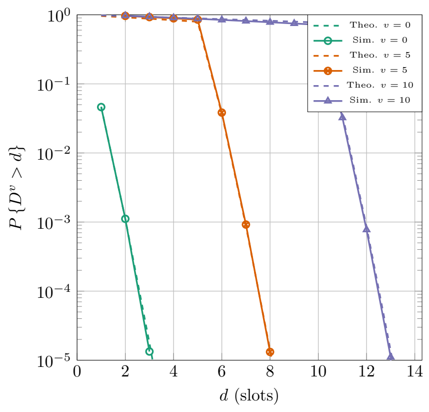
Now we obtain the delay distribution with vacation. If a busy period starts with a vacation v, then all the packets in the queue will encounter a delay of at most v, in addition to the queuing delay, and we get for a fixed v

$$P\left\{D^v_{\mathbf{N},\mathbf{D},\mathbf{L}} > d\right\} \leq P\{D_{\mathbf{N},\mathbf{D},\mathbf{L}} > d - v\}. \tag{3.12}$$

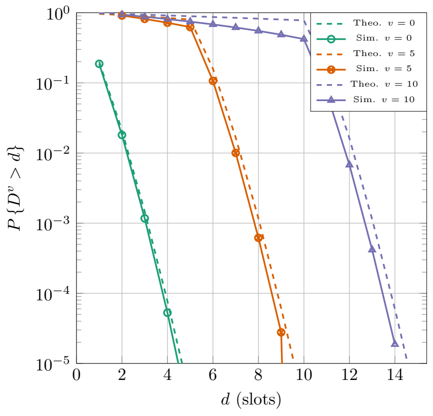
If the vacation is a distribution, then the delay distribution with vacation can also be written as the convolution of the delay distribution and the vacation:

$$P\{D_{\mathbf{N},\mathbf{D},\mathbf{L}} = d - V\} = D_{\mathbf{N},\mathbf{D},\mathbf{L}} \circledast V. \tag{3.13}$$

In Figures 3.5 and 3.6 we can see the delay distribution compared to simulations of the same scenarios used in the queue section. We can observe that the predicted delays closely follow the simulation results.

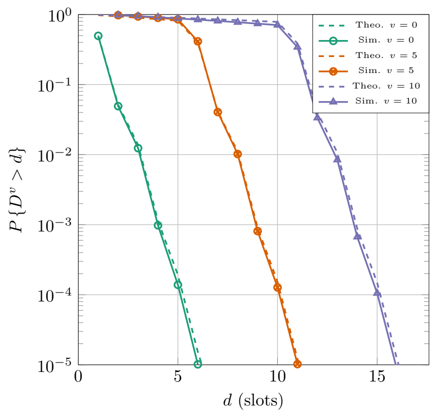


(a) $\mathbf{A} = [(70, 1500, 1)]$

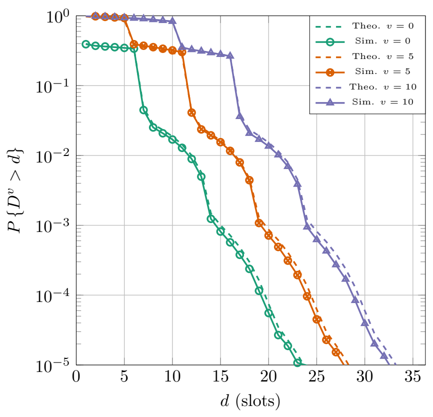


(b) $\mathbf{A} = [(15, 75, 1)]$

Figure 3.5: CBR delay distribution for various $v \in \{0, 5, 10\}$



(a) $\mathbf{A} = [(15, 500, 1), (10, 400, 2)]$



(b) $\mathbf{A} = [(4, 150, 1), (10, 800, 7)]$

Figure 3.6: CBR delay distribution for multiple aggregates and various $v \in \{0, 5, 10\}$

3.5 IPDV distribution

The packet delay variation (PDV) is defined in [58] as $d_i - d_{ref}$, i.e. the difference between a reference E2E delay d_{ref} and the E2E delay of the i-th packet d_i over the same path. The IPDV [57] is the PDV delay between successive packets, i.e. using d_{i-1} as the reference E2E delay. It is often also referred to as *jitter*, however, the term "jitter" is used in various disciplines with slightly differing meaning, hence we will use IPDV. The IPDV is an important metric for applications that depend on a steady and timely delivery of packets, such as VoIP, multimedia applications etc. The effects of IPDV can be mitigated by a suitably sized buffer at the receiver.

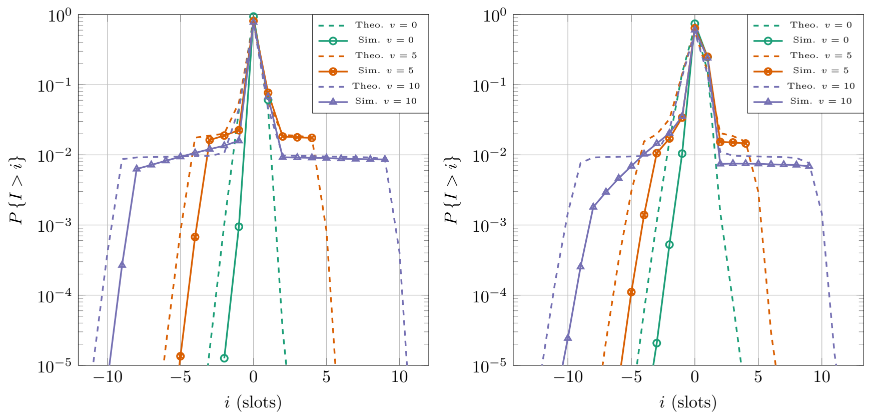
The IPDV distribution of the aggregate can be expressed as

$$P\left\{I^v(k) = i\right\} = \sum_{d>0} P\left\{D^v(k) = d - i \wedge D^v(k-1) = d\right\}. \tag{3.14}$$

Assuming stationarity and independence between successive arrivals, we can write this as

$$\begin{aligned} P\left\{I^v(k) = i\right\} &= \sum_{d>0} P\left\{D^v(k) = d - i\right\} \cdot P\left\{D^v(k-1) = d\right\} \\ &= \sum_{d>0} P\left\{D^v(k) = d - i\right\} \cdot P\left\{D^v(k) = d\right\}. \end{aligned} \tag{3.15}$$

Hence, we can readily use the delay distribution obtained in Section 3.4 to calculate the IPDV. Figures 3.7 and 3.8 show the IPDV pmf for the same one-node scenarios of the previous section. The predictions made using (3.15) follow the simulation's results, including the peaks. The different peaks occur when we have an aggregate comprising multiple lengths, and are the result of the interweaving of packets with different lengths (and hence different delays). For example, in Figure 3.8b this is the most visible: the peaks occur at $-6 = L_1 - L_2 = 1 - 7$, and $6 = L_2 - L_1 = 7 - 1$. In Figure 3.8a we also have the peaks, but they are adjacent to 0 (as we round the bin size of the pmf to 1 slot), at $-1 = L_1 - L_2 = 1 - 2$, and $1 = L_2 - L_1 = 2 - 1$.

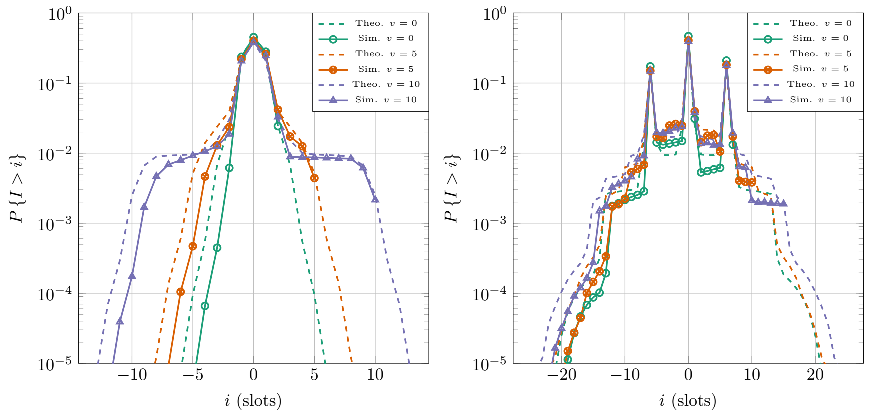


(a) $\mathbf{A} = [(70, 1500, 1)]$ (b) $\mathbf{A} = [(15, 75, 1)]$

Figure 3.7: CBR IPDV distribution for various $v \in \{0, 5, 10\}$

3.6 Busy period distribution

We now discuss the busy period of the CBR aggregate in the presence of a vacation. The busy period is the time during which a channel is busy. Knowing



(a) $\mathbf{A} = [(15, 500, 1), (10, 400, 2)]$ (b) $\mathbf{A} = [(4, 150, 1), (10, 800, 7)]$

Figure 3.8: CBR IPDV distribution multiple aggregates and for various $v \in \{0, 5, 10\}$

the busy period distribution can useful in a couple of settings. For example, to reduce complexity in algorithms, time scales up to the busy period are considered (e.g. admission control [150] or network calculus [22]). In our scenario, it allows giving an upper bound on the delay encountered by lower priority traffic, by incorporating the busy period distribution into the waiting time, as we will discuss in the next section.

The author of [189] analytically determines the probability of a busy period having a length z. However, their formula only works for the $N * D/D/1$ queue, and does not take the vacation into account. In this section we will derive a closed formula that supports vacation and a $N_j * D_j/D_j/1$ queue.

We first introduce some new notation. The least common multiple of all periods is denoted as $G = \text{lcm}(\mathbf{D})$. This means that the superposition of all the flows repeats every G slots. Furthermore, we define $E_j = G_j/D_j$, indicating how many packets aggregate j sends in G slots. Finally, the load in the interval $[x, y]$ is written as $\rho_{[x,y]}$.

To calculate the busy period distribution, we introduce two new events, (C) and (D). We show an updated diagram in Figure 3.9 and describe all events:

(A) the system is empty at $t = -\epsilon$

(B) there is an arrival from a source at $t = \epsilon$

(C) there are l_j arrivals (uniformly distributed) from aggregate A_j in the interval $[0, z + v]$, such that $\mathbf{l}^\top \mathbf{N} = z$

(D) the system is or becomes empty at $t = z + v$

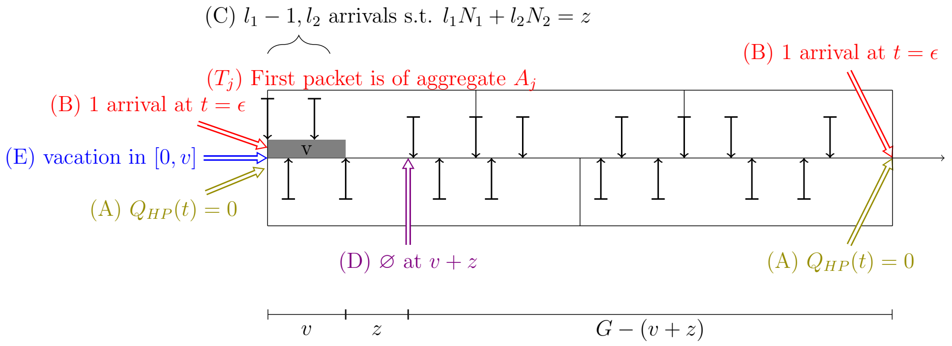

(C) $l_1 - 1, l_2$ arrivals s.t. $l_1N_1 + l_2N_2 = z$
(B) 1 arrival at $t = \epsilon$
(T_j) First packet is of aggregate A_j
(B) 1 arrival at $t = \epsilon$
(E) vacation in $[0, v]$
v
(A) $Q_{HP}(t) = 0$
(D) $\varnothing$ at $v + z$
(A) $Q_{HP}(t) = 0$
v
z
$G - (v + z)$

Figure 3.9: Events characterizing the busy period for two CBR aggregates, and with T_1

(E) a workload of v starts at $t = 0$

(T_j) the packet from event (B) is of aggregate A_j

The distribution of the busy period process for a CBR aggregate $Z^{CBR}(k)$ can then be written as

$$\begin{aligned}
P\left\{Z^{CBR} = z|\text{starting with vacation } v\right\} &= P\left\{CD|ABE\right\} \\
&= \sum_{i=1}^{m} P\left\{T_j\right\} P\left\{CD|ABET_j\right\} \\
&= \sum_{i=1}^{m} \frac{P\left\{T_j\right\}}{P\left\{A|BET_j\right\}} P\left\{C|BET_j\right\} P\left\{AD|BECT_j\right\}.
\end{aligned}$$

Now we discuss each of these probabilities.

3.6.0.0.1 $\mathbf{P\left\{T_j\right\}}$ The probability of our first packet being of aggregate A_j, considering the starting times of flows are uniformly distributed, can be expressed as

$$P\left\{T_j\right\} = \frac{N_j/D_j}{\sum_{i=1}^{m} N_i/D_i}.$$

3.6.0.0.2 $\mathbf{P\left\{A|BET_j\right\}}$ Next is the probability of the system being empty at $t = -\epsilon$, given a packet of aggregate A_j at $t = +\epsilon$ and the remainder of the LP packet is of length v at $t = 0$.

Denote $\hat{D} = \max\{\mathbf{D}\}$, and $\hat{E}$ the E corresponding to $\hat{D}$. As $\rho_{\left[G-\hat{D},G\right]} < 1$, our system must be empty somewhere in $\left[G - \hat{D}, G\right]$ and thus anything happening in

$\left[0, G - \hat{D}\right]$ will not contribute to event (A), reducing the probability to $P\left\{A|BT_j\right\}$.

In the interval $\left[G - \hat{D}, G\right]$, an aggregate A_j will occupy $\frac{N_j L_j E_j}{\hat{E}}$ slots. Given event (T_j) and $\rho_{\left[G-\hat{D},G\right]} < 1$, we are sure that the initial $\frac{L_j E_j}{\hat{E}}$ slots from event (T_j) will not contribute to the load towards the end of the slot.

The probability of a system being empty at a point s in an interval $[x, y]$ is equal to $1 - \rho_{[x,y]}$, as noted in [154, 189], hence the probability is

$$P\left\{A|BET_j\right\} = 1 - \frac{\sum_{i=1}^{m} N_i L_i E_i - L_j E_j}{G}.$$

3.6.0.0.3 $\mathbf{P\left\{C|BET_j\right\}}$ Third, the probability that there are l_j arrivals in $[0, v + z]$ such that the busy period equals z, given a vacation v and starting with a packet from A_j, can be calculated by obtaining the probability that the convolution of the aggregates' binomial distributions equals z:

$$P\left\{C|BET_j\right\} = P\left\{F_{BP}^{\circledast}(j) = z\right\}$$

where

$$\begin{aligned} F_{BP}^{\circledast}(j) &= \circledast_{i=1}^{m} F_{BP}(i,j) \\ F_{BP}(i,j) &= \begin{cases} x : (\delta_{i-j}..N_j){\cdot}L_j \\ y : \mathrm{Bin}(N_j\text{-}\delta_{i-j}, 0..N_j\text{-}\delta_{i-j}, \frac{v+z}{D_j}) \end{cases} \end{aligned}$$

As the period starts with a packet from aggregate A_j due to our event (T_j), we must compensate for having one active source less, which is taken into account by the Kronecker delta δ_{i-j}.

3.6.0.0.4 $\mathbf{P\left\{AD|BECT_j\right\}}$ The final probability that the system is empty at both time $t = v{+}z$ and $t = -\epsilon$, given a vacation of v at $t = 0$, l_j arrivals from aggregate A_j in $[0, v{+}z]$ and the first arrival at $t = \epsilon$ being of aggregate A_j can be split into two parts:

- $\mathbf{P\left\{A|BECT_j\right\}}$ the probability that the system is empty at time $t = G{-}\epsilon$, given an arrival of type T_j and a busy period of length z in the interval $[0, v{+}z]$ is $1{-}\rho_{[D_j-(v+z),D_j]}$. This load can be written as

$$\rho_{[0,v+z]} = \sum_{j=1}^{m} \frac{L_j(N_j - l_j)}{D_j - (z + v)},$$

for a specific configuration of arrivals. However, there are several configurations of arrivals which need to be taken into account, each with their own probability and hence weight. Say $\mathbf{H}$ is a matrix of m columns, H_r is the r-th configuration and $H_{r,c}$ is the number of arrivals in the $r-th$ configuration from aggregate A_c in the interval $[0, v+z]$, such that $\mathbf{H}^T\mathbf{L}$ is a vector with all values equal to z. An additional constraint on this matrix is that $H_{r,i} > 0$, i.e. for event (T_j) there must always be at least one packet. Then the probability for a certain configuration equals

$$p_{j,H_r} = \prod_{i=1}^{m} \text{Bin}(N_i - \delta_{j-i}, H_{r,i} - \delta_{j-i}, \frac{z+v}{D_i}). \tag{3.16}$$

Note that here also the first packet must be taken into account. This is expressed by the Kronecker delta. We can now write the probability as

$$P\{A|BECT_j\} = \sum_r \frac{p_{j,H_r}}{\sum_s p_{j,H_s}} \cdot \left(1 - \sum_{i=1}^{m} \frac{L_i(N_i - H_{r,i})}{D_i - (z+v)}\right). \tag{3.17}$$

In Appendix A we list pseudocode to calculate $\mathbf{H}$.

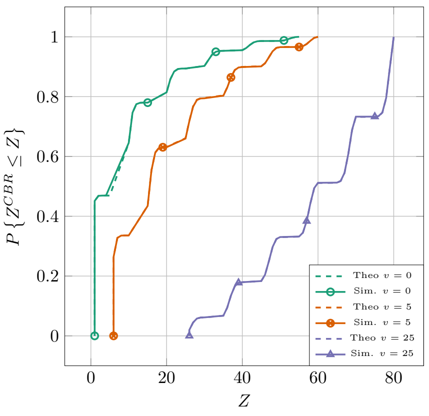
- $\mathbf{P\{D|BECT_j\}}$ the empty probability at time $t = z+v$ due to l_j arrivals in $[0, z+v]$ is
$$P\{D|BECT_j\} = 1 - \rho_{[0,v+z]} = 1 - \frac{z - L_j}{z+v}$$

Combining all these different probabilities, we get

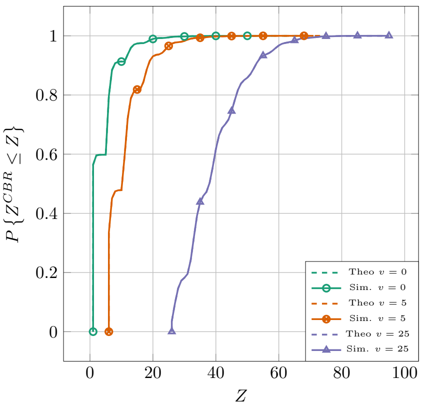
$$\begin{aligned} P\left\{Z^{CBR} = z|v\right\} = \sum_{j=1}^{m} \frac{N_j/D_j}{\sum_{i=1}^{m} N_i/D_i} \cdot \frac{1}{1 - \frac{\sum_{i=1}^{m} N_i L_i E_i - L_j E_j}{G}} \cdot \\ P\left\{F_{BP}^{\circledast}(j) = z\right\} \cdot \\ \left[\sum_r \frac{p_{j,H_r}}{\sum_s p_{j,H_s}} \cdot \left(1 - \sum_{i=1}^{m} \frac{L_i(N_i - H_{r,i})}{D_i - (z+v)}\right)\right] \cdot \\ \left(1 - \frac{z - L_j}{z+v}\right). \end{aligned} \tag{3.18}$$

In Figure 3.10 we show the cumulative distribution function (cdf) of the busy period distribution calculated as in (3.18) (the dashed line) and the busy period distribution obtained through running 10^6 simulations (the full line). Each curve represents a vacation period $v \in \{0, 5, 25\}$. It is clearly visible that changing the vacation length not only shifts, but also changes the distribution, as the load increases and the different flows will interact with each other with higher probability. The simulation results match the prediction very closely.

In Figure 3.11 we look briefly at the delay incurred due to HP traffic. This plot shows the worst-case delay for 2nd priority and low priority traffic as it is serviced in a strict priority scheduler. The highest priority HP0 is an aggregate



(a) $\mathbf{A} = [(5, 100, 1), (5, 100, 10)]$



(b) $\mathbf{A} = [(10, 150, 1), (10, 250, 6)]$

Figure 3.10: Busy period for two different scenarios with $v \in \{0, 5, 25\}$

$A_1 = (10, 80, 1)$, while the second priority HP1 is an aggregate $A_2 = (8, 120, 7)$. The lowest priority represents the vacation $v \in \{6, 12\}$, each being picked with equal probability. The plot shows the upper bound on the waiting time distribution for HP1 and LP, i.e. the time before a frame from that priority is served. We condition on worst-case behavior, e.g. for HP1 we first start with a LP packet followed by the busy period of HP0 traffic, only then serving HP1. This is calculated using

$$P\{D = d\} = \sum_{v+z=d} P\{V = v\} P\left\{Z^{CBR} = z|v\right\}. \tag{3.19}$$

The simulation curves in Figure 3.11 are the result of 10^6 simulations, and match the theoretical curves exactly. This approach can be applied to any stack of prioritized CBR traffic aggregates.

We note, though, that also here the restriction $v+M < \min(\mathbf{D})$ is important to ensure independence between busy period distributions. Assume for example that we have 2 CBR aggregates with $A_1 = (2, 1, 8)$ and $A_2 = (5, 3, 100)$, and that the first packet is from aggregate A_1. Then a busy period can be less than, or more than 8 slots. It can not be exactly 8 slots, as at $t = 8$ a source from A_1 will send again, and extend the busy period with at least 1 slot. The distribution $F_{BP}(i, 1)$ for $z < 8$ differs from $F_{BP}(i, 1)$ for $z > 8$, hence convolution can not be used.

3.7 Conclusion

In this section, we first gave an introduction the CBR traffic in Section 3.1, which also touched upon the subject of a non-preemptive channel, and how it impacts the strict priority scheduler. We first derived the queue size distribution,

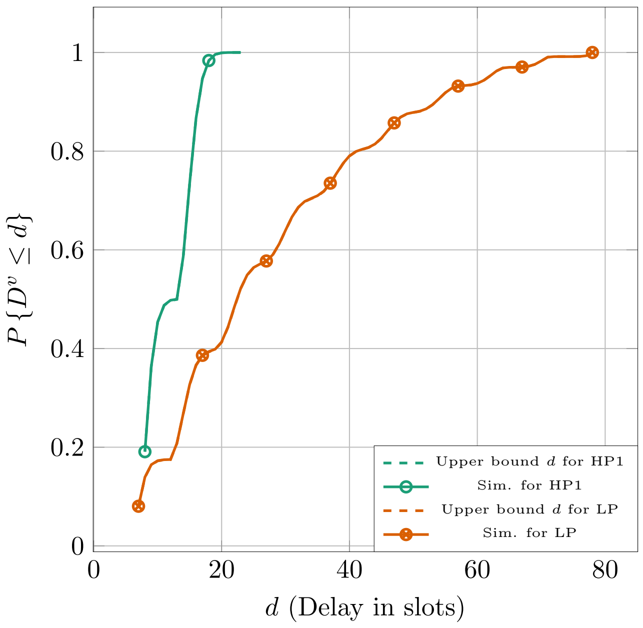


Figure 3.11: Waiting time distribution for HP1 and LP traffic. $A_1 = (10, 80, 1)$, $A_2 = (8, 120, 7)$, $v \in \{6, 12\}$

taking a vacation into account, in Section 3.3, for a CBR aggregate where all flows have similar characteristics, and then expanded this to calculate an upper bound on the queue size distribution (with vacation) when the CBR aggregate comprises flows with different periods and/or packet lengths. In Section 3.4 we then proceeded to calculate the delay distribution, given a vacation. In Section 3.5 we approximated the IPDV distribution using the delay distribution from the previous section. We then closed the section on the CBR traffic with the development of an algorithm to calculate the busy period of a CBR aggregate which starts with a vacation.

Through simulations, we have verified that the analytical results provide a good upper bound on the results from the simulations.

Chapter 4

Medium priority VoIP traffic

4.1 Introduction

In the previous section, we looked at the behavior of the highest priority traffic in a strict priority scheduler. In this section, we discuss the flows with lower priority than the CBR flows, which we call here the medium priority (MP) flows. For our purpose these flows comprise an aggregate of on-off flows.

We first give a small introduction to network calculus in Section 4.2, which we will use further in this section. Then we discuss the characteristics of the MP traffic in Section 4.3, and compare the bounds given by NC with the simulation results for the queue size (Section 4.6), delay (Section 4.7), IPDV (Section 4.8) and busy period (Section 4.9) distributions.

4.2 Network calculus primer

Deterministic network calculus [111] is a theoretical framework for analyzing worst-case performance in computer networks. Upper bounds on certain performance metrics, such as delay and queue size, can be obtained through the use of functions that indicate constraints on traffic flows and the service offered.

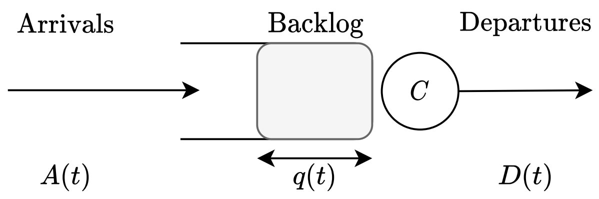


Figure 4.1: The queuing model

In Figure 4.1 we show the model that is used: bits arrive in a buffer, that is served by a work-conserving scheduler over a fixed capacity link with rate C bps. The cumulative amount of arrivals over the interval $[0, t]$ are denoted by the arrival curve (AC) $A(t)$, the number of bits in the queue at a time t is written as $q(t)$, while the cumulative number of bits that have left the queue is given by the departure curve (DC) $D(t)$. The functions $A(t)$ and $D(t)$ are non-decreasing, and are assumed to be 0 for $t \leq 0$.

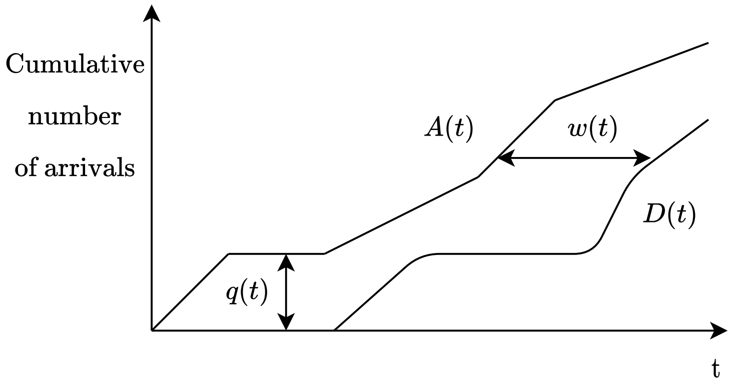


Figure 4.2: Arrival curve $A(t)$ and departure $D(t)$ curve for DNC

In Figure 4.2 we can see an example of $A(t)$ and $D(t)$. Using these two functions we can easily derive the backlog at a time t using $q(t) = A(t) - D(t)$, i.e. the vertical deviation, and the waiting time using $w(t) = \inf\{y > 0 | D(t+y) \geq A(t)\}$, i.e. the horizontal deviation between the arrival and departure curves.

However, $A(t)$ is not a useful traffic model. In general the function $A(t)$ of a traffic flow is not known and difficult to obtain. Additionally, it is a verbose description and an explicit function of time. A more suitable and compact representation is the traffic envelope (TE) function $E(\tau)$ [129]. Such a traffic envelope describes the deterministic upper bound of a flow's arrivals over any interval τ. More formally, $E(\tau)$ is a traffic envelope for an arrival curve $A(t)$ if

$$A(t+\tau) - A(t) \leq E(\tau), \forall \tau \geq 0, \forall t \in \mathbb{R}. \tag{4.1}$$

The server counterpart of the traffic envelope is the service curve (SC) $S(\tau)$, which provides a deterministic lower bound on the guaranteed service a server can offer in an interval τ. A server offers a service curve $S(\tau)$ if for all departures holds that

$$D(t+\tau) - D(t) \geq S(\tau), \forall \tau \geq 0, \forall t \in \mathbb{R}. \tag{4.2}$$

Thus, a server *might* provide more service, but in the worst case it will give a service of $S(\tau)$ in an interval τ.

We will now give a simple example of a traffic envelope and a service curve, and the bounds that can be derived from it. A descriptor that is usually reasonably easy to obtain and is fairly intuitive, is the linear bounded process. It has the form $f(x) = ax+b$ where constants a and b are, respectively, a rate in bits per second and a burstiness in bit, a measure for how much deviation from the rate a is allowed.

In Figure 4.3 we show such a function for the traffic envelope and service curve. The traffic envelope is given by $\rho_a\tau+b_a$, where ρ_a can be seen as the average arrival rate and b_a the burstiness, the maximal number of bits that can arrive at one time. Note that this function $E(\tau)$ is not unique. For example, both $E(\tau)$ and $E_2(\tau)$ could be valid traffic envelopes that bound the arrivals of the traffic flow. The traffic envelope $E_2(\tau)$ has a lower average arrival rate $\rho_{a,2}$, but a higher allowed burst size $b_{a,2}$.

The service curve is given by $S(\tau) = (\rho_s\tau-b_s)^+$, and is also called a rate-latency server. It offers a minimal service ρ_s with an upper bound on the latency, the latest time after which a server starts serving a flow at the service rate ρ_s. It is of course possible that the server already services a flow at an earlier time, possibly at a lower rate, however, in the worst case, it will take b_s/ρ_s seconds before it starts serving at the rate ρ_s.

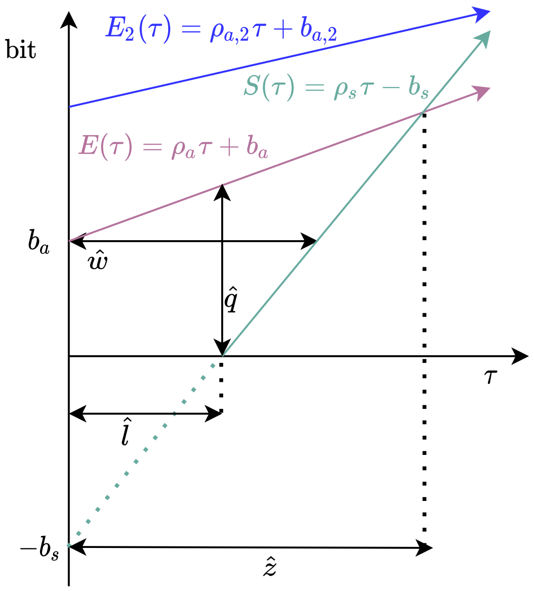

bit
$E_2(\tau) = \rho_{a,2}\tau + b_{a,2}$
$S(\tau) = \rho_s\tau - b_s$
$E(\tau) = \rho_a\tau + b_a$
b_a
$\hat{w}$
$\hat{q}$
τ
$\hat{l}$
$-b_s$
$\hat{z}$

Figure 4.3: Bounds for DNC

Using this description we are able to obtain some important metrics, which we can also visually inspect in Figure 4.3. The maximal vertical distance between $E(\tau)$ and $S(\tau)$, indicated by the vertical arrow labeled $\hat{q}$ in Figure 4.3, provides an upper bound on the queue size. The maximal horizontal distance between the arrival curve and the service curve, as indicated by the horizontal arrow, labeled $\hat{d}$, in the figure, can be interpreted as the upper bound on the waiting time. The upper bound on the busy period can be seen as the moment when the service curve becomes larger than the traffic envelope, i.e. the intersection of both curves. The intersection of the service curve and the x-axis, finally, is the upper bound on the server's latency.

Formally, the upper bound on the queue, the waiting time, busy period and latency are given respectively by

$$\hat{q} = b_a + \rho_a \frac{b_s}{\rho_s}, \tag{4.3}$$

$$\hat{d} = \frac{b_s + b_a}{\rho_s}, \tag{4.4}$$

$$\hat{z} = \frac{b_s + b_a}{\rho_s - \rho_a}, \text{ and} \tag{4.5}$$

$$\hat{l} = \frac{b_s}{\rho_s}. \tag{4.6}$$

As the bounds provided by DNC assume the worst-case behavior, they are often very pessimistic (but exact). The DNC can be extended to SNC, in which the traffic envelope and/or service curve are stochastically bounded, often resulting in drastically improved upper bounds, at the cost of exceeding these bounds with a certain probability.

In the SNC the deterministic arrival and services curves are replaced with a stochastic counterpart: the traffic envelope of (4.1) becomes the stochastic traffic envelope (STE) $\mathcal{G}$

$$P\left\{A(t) - A(s) > \mathcal{G}(t-s, \varepsilon^A)\right\} \leq \varepsilon^A, \forall s \leq t, \tag{4.7}$$

and the stochastic service curve (SSC) $\mathcal{S}$ given by

$$P\left\{S(t) - S(s) < \mathcal{S}(t-s, \varepsilon^S)\right\} \leq \varepsilon^S, \forall s \leq t, \tag{4.8}$$

where ε^A and ε^S are the violation probabilities for the respective curves.

In Figure 4.4 we show three different $\mathcal{G}$ functions, where ρ_a is kept constant, but for different burst sizes $b_{a,1} \leq b_{a,2} \leq b_{a,3}$ and different violation probabilities $\varepsilon_2 \geq \varepsilon_3$. If we set $\varepsilon^A = 0$, then we obtain the bounds from DNC. We can see in the figure that we now obtain stochastic bounds on the delay (the queue size also has its stochastic counterpart, but is left out, in order to not clutter the image), allowing us to construct a probability distribution of the metrics.

Thus, for a STE $\mathcal{G}(\tau, \varepsilon_a)$ and SC $S(\tau)$, we can write

$$P\left\{Q(t) > b_a + \rho_a \frac{b_s}{\rho_s}\right\} \leq \varepsilon_a, \tag{4.9}$$

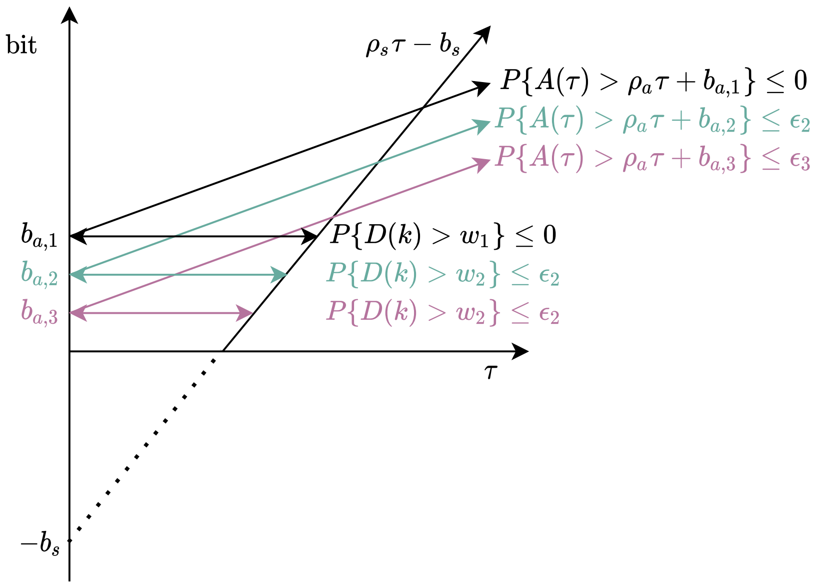

bit
$\rho_s\tau - b_s$
$P\{A(\tau) > \rho_a\tau + b_{a,1}\} \leq 0$
$P\{A(\tau) > \rho_a\tau + b_{a,2}\} \leq \epsilon_2$
$P\{A(\tau) > \rho_a\tau + b_{a,3}\} \leq \epsilon_3$
$b_{a,1}$
$b_{a,2}$
$b_{a,3}$
$P\{D(k) > w_1\} \leq 0$
$P\{D(k) > w_2\} \leq \epsilon_2$
$P\{D(k) > w_2\} \leq \epsilon_2$
τ
$-b_s$

Figure 4.4: Some bounds for SNC

$$P\left\{D(k) > \frac{b_s + b_a}{\rho_s}\right\} \leq \varepsilon_a, \text{ and} \tag{4.10}$$

$$P\left\{Z(k) > \frac{b_s + b_a}{\rho_s - \rho_a}\right\} \leq \varepsilon_a. \tag{4.11}$$

4.3 VoIP

We now describe the traffic and the model that we use for the MP traffic. An on-off source is a two-state Markovian chain, as depicted in Figure 4.5. Such a source generates in the ON state traffic at a fixed rate $\lambda = \frac{L_{VoIP}}{\gamma}$, with L_{VoIP} being the length of a packet, and γ the inter-arrival (IA) time between packets, while in the OFF state the source remains silent. Every time a packet is sent, the source will remain in the ON state with probability α, and switch with probability $1-\alpha$. In the OFF state, every γ seconds the state switches to the ON state with probability $1-\beta$.

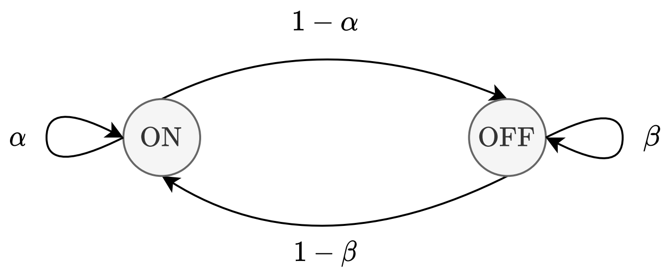


Figure 4.5: The state transitions for a two state on-off traffic model

Such an on-off model is often used to represent certain kinds of bursty traffic. A typical application is a single VoIP flow that uses silence detection [56, 15]. An aggregate of on-off sources can also be used to model a single video source [75, 171].

Here, we use it to model VoIP traffic with silence detection. Common values for the model are $\alpha = 0.0285$ and $\beta = 0.0154$, i.e. on average we stay $\frac{1}{\alpha} \approx 35$ slots in the ON state, and $\frac{1}{\beta} = 65$ slots in the OFF state. We notate the probability that a single VoIP source is in the ON state by $p_{ON} = \alpha^{-1}/(\alpha^{-1}+\beta^{-1}) = 0.35$. In the ON state, a VoIP packet is generated every $\gamma = 10$ ms, hence, a packet burst is on average 0.35s and is followed by 0.65s of silence. A VoIP packet's length has a payload length of around $L_{VoIP} = 38$ byte which results in a total packet length of 80 byte, due to Ethernet, IP and UDP headers. A single VoIP source thus generates traffic at a peak rate of $\lambda = 64$ kbps and on average 22.4 kbps.

To accommodate a single on-off source, it must be serviced at a rate of at least λ. Any less service rate, and packets will encounter a delay that is too large or get dropped when the source is too long in the ON state. Thus, we must provision for the worst case. When $M > 1$ and all the sources are in the ON state, then we should reserve $M \cdot \lambda$ to guarantee no loss and a low delay. But this wastes service because only a fraction of the time all sources are sending at the same time. Usually, only some sources will be in the ON state. If we accept a small probability of having "too many" active sources, we can reduce the required service rate significantly.

We can quantify how much we can reduce the service rate before the QoS deteriorates significantly. For this we need to construct a traffic model, a general description of the behavior. A model that is often used to describe traffic is the token bucket model. Such a token bucket logically consists of a bucket that holds a number of tokens $k \in [0, b]$ (for example a number of bits). The bucket is filled at a rate λ_{tb}, as long as the amount of tokens is less than b. Whenever a packet is checked for conformance, the TB is inspected to see if it contains sufficient tokens. If so, the packet length is removed from the tokens, and the packet is conformant and can pass. If there are no sufficient tokens, no tokens are removed, and the packet is marked non-conformant or dropped, in the case of traffic policing. This process is also shown in Figure 4.6.

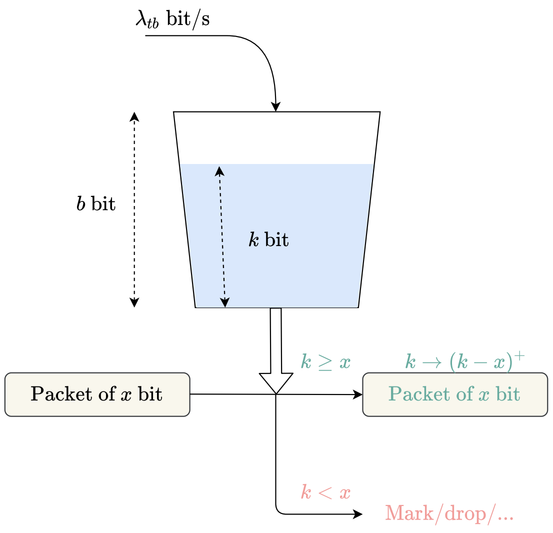


Figure 4.6: A depiction of the token bucket algorithm

This algorithm allows traffic to arrive at a long-term average rate of λ_{tb} with bursts up to b bits, which, conveniently, coincides with the linear bounded process described in the previous section. Token buckets occur in many areas of literature, such as regulating irregular traffic in packet switched networks [182, 175], and providing tighter bounds in NC models [111] and communication topologies with cyclic dependence [160].

We now proceed to define the token bucket parameters for an aggregate of on-off sources. Assuming the on-off model described above with parameters α and β, an algorithm to calculate the non-conforming probability ε_{nc} for an aggregate of M on-off sources is given in [71]. In [38] the authors approximate the on-off aggregate as a Poisson process. It becomes more accurate as more VoIP sources are aggregated. Here, we will combine both approaches to get a more accurate stochastic token bucket descriptor.

First, we discuss [71]. Let $M_t \in [0, M]$ and $k_t \in [0, b]$ indicate respectively the number of active sources and number of tokens in the TB at a time t, and let $\pi_i(x)$ be the steady state probability of having $i \in [0, M]$ active sources and having x or less than x tokens available. Then the non-conforming probability can be expressed as the probability that we have no tokens, while there are active sources:

$$\varepsilon_{nc} = P\{k_t = 0 | M_t \geq 1\} = \frac{\sum_{1=0}^{M} \pi_i(0)}{1 - P\{M_t = 0\}} \tag{4.12}$$

The paper shows the algorithm to calculate the steady state $\pi_i(0)$, where $\pi_i(0)$ depends on λ_{tb} and b (and the on-off model's parameters), the token bucket parameters, which is then used to give the function $f_{\varepsilon_{nc}}(\lambda_{tb}, b)$, which gives the non-conforming probability of the on-off aggregate to a token bucket with parameters λ_{tb} and b. The model is able to predict the non-conforming probability within a factor of 10 when b is large. However, for small b it is less accurate.

Therefore, we combine it with another model, that is more accurate for small burst sizes or when the TB rate λ_{tb} is large and the burst sizes are small. In [38] they discussed that an aggregate of a large number on-off sources can be modeled using a Poisson model. In [131] they approximated the leaky bucket for Poisson arrivals of fixed packet lengths, and concluded that it approaches the steady state queue of the M/D/1 system, given by (4.13)-(4.15)

$$\pi_0 = 1-\lambda \tag{4.13}$$

$$\pi_1 = (1-\lambda)(e^{\lambda}-1) \tag{4.14}$$

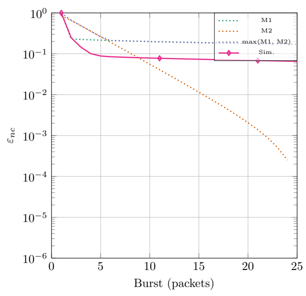
$$\pi_n = (1-\lambda)\left(e^{n\lambda}+\sum_{k=1}^{n-1} e^{k\lambda}(-1)^{n-k}\left[\frac{(k\lambda)^{n-k}}{(n-k)!}+\frac{(k\lambda)^{n-k-1}}{(n-k-1)!}\right]\right) \quad (n \geq 2). \tag{4.15}$$

Hence, we can also model the aggregate for the flows in this section also by Poisson traffic with fixed packet lengths. We set the Poisson parameter λ to $\frac{M}{\gamma}p_{ON}$.

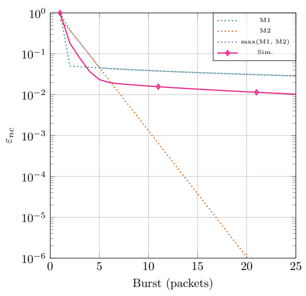
In Figure 4.7 we show the curves for both methods presented above, and combine the two to get better TB parameters. The plots show on the x-axis the burst size, and on the y-axis the non-conforming probability for 20 VoIP sources for a TB rate $\lambda_{tb} = M \cdot \lambda \cdot R$, where R differs for each plot. The full curve describes the TB non-conforming probability of the simulations, calculated using [177]. The dotted curves show the theoretical approximations. In the plot we refer to [71] as M1, and the M/D/1 approximation of [131] as M2. We also added a curve max(M1,M2) which takes the maximum violation probability of M1 and M2 for a certain burst size. We can observe that both M1 and M2 are accurate for some burst sizes. The larger R becomes, the better the Poisson approximation becomes, however the tail is better described by M1. This method has some inaccuracies due to the fact that the model itself assumes fluid arrivals, while the simulations use discrete packets. Additionally, the M1 method is not always computationally stable. For example, when using $R = 0.6$, the system can not be solved, and we can not obtain the violation probability.

In the plots we can also clearly see the trade-off between a small λ_{tb} and b. As λ_{tb} becomes smaller, the burstiness of the corresponding traffic model becomes much larger.

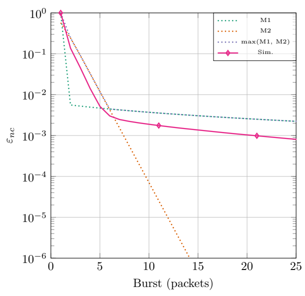
Due to the presence of high priority traffic in the strict priority scheduler a VoIP packet can only be serviced if the high priority queue is empty. Hence, the vacation of the VoIP traffic is the busy period of the high priority traffic, as calculated in Section 3.6. The delay distribution of the VoIP traffic can then be convoluted with the busy period to obtain the encountered delay.



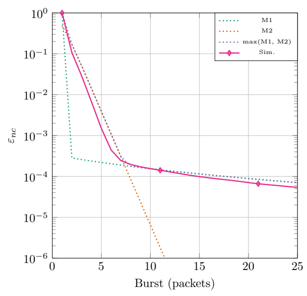
(a) $R = 0.41$



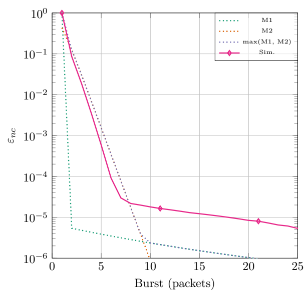
(b) $R = 0.51$



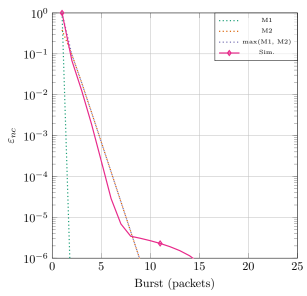
(c) $R = 0.61$



(d) $R = 0.71$



(e) $R = 0.81$



(f) $R = 0.91$

Figure 4.7: Matching the aggregate of $M = 20$ VoIP sources to a TB process for a rate $\lambda_{tb} = M \cdot \lambda \cdot R$

4.4 The server vacation

The upper bound on the server's vacation (or latency), with respect to the VoIP flows, is written using the busy period of the high priority CBR aggregates, as given by (3.18) on Page 38, and the length of the LP packets.

If we assume low priority packets have a fixed length of L slots, we can write

$$P\left\{V^{VoIP}(k) = v\right\} \leq P\left\{Z^{CBR}(k) = v - L | L\right\}. \tag{4.16}$$

This gives an upper bound on the vacation, as a VoIP packet will not always arrive at the same moment as the low priority packet starts its transmission. We can improve (4.16) by assuming that the VoIP packet will arrive somewhere in $]0, L]$, uniformly distributed, and write:

$$P\left\{V^{VoIP}(k) = v\right\} = \sum_{l \in \mathcal{U}_{[0,L]}} P\left\{Z^{CBR}(k) = v - l | l\right\} P\left\{l\right\}. \tag{4.17}$$

where $\mathcal{U}_{[a,b]}$ is the (discrete) uniform distribution over $[a, b]$.

In more realistic scenarios, however, it is likely that low priority traffic comprises packets of different lengths, given by the packet length distribution L^{LP}. In this

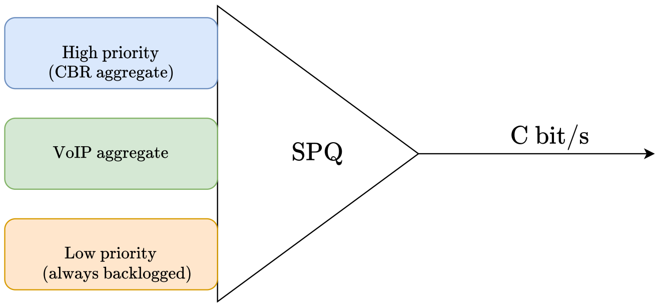


Figure 4.8: Scheduler setup for the simulations of this section

case, we replace $l \in \mathcal{U}_{[0,L]}$ in (4.17) with $l \in \mathcal{L}(L^{LP})$, where

$$\mathcal{L}(X) = \sum_{x \in X} P\{X = x\} \mathcal{U}_{[0,x]} \tag{4.18}$$

gives a new distribution that takes the lengths into account. Thus, we calculate the server vacation pmf as

$$P\left\{V^{VoIP}(k) = v\right\} = \sum_{l \in \mathcal{L}(L^{LP})} P\left\{Z^{CBR}(k) = v - l | l\right\} P\{l\}. \tag{4.19}$$

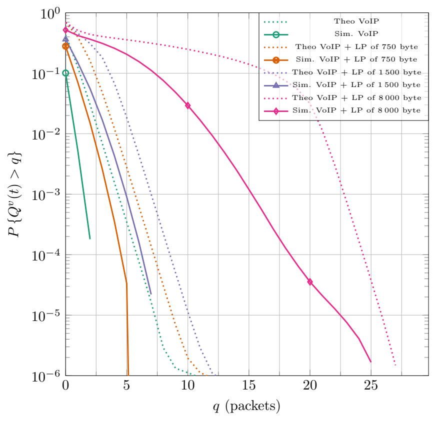
4.5 Simulation setup

The simulation setup for the VoIP simulations of this section is shown in Figure 4.8. There are three queues that go into a strict priority scheduler that is served by a channel of capacity R bps. One queue is for high priority traffic (in figure (a) of 4.9 and 4.10, however, there was no high priority traffic), one for VoIP and one for low priority traffic. The low priority source generates packets of fixed length at uniformly distributed intervals, such that the channel load is 50%. The length of the low priority packet can vary in different simulations: we simulated for sizes of 750 byte, 1 500 byte and jumbo frames of 8 000 byte.

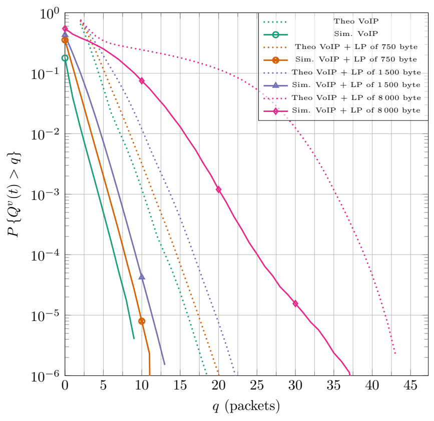
In the plots below, the different curves indicate the length of the low priority packet. On the x-axis we show the number of packets (for the queue) or the number of slots (other plots), where a slot is the time required to send a single VoIP packet. We ran simulations for four different scenarios, two for a channel with capacity $R = 5$ Mbps and $M = 20$, and two for $R = 100$ Mbps and $M = 40$, and with and without an aggregate of high priority CBR streams.

4.6 Queue size distribution

In Figure 4.9 we show the queue size distribution for an aggregate of 20 VoIP sources and a channel rate of 5 Mbps, while Figure 4.10 shows the queue size distribution for 40 VoIP sources and a channel rate of 100 Mbps. The (a) figures have no HP packets, while the (b) figures also have an aggregate of CBR sources, indicated in the caption of the figure. The curves are calculated using the queue size obtained in the network calculus primer.

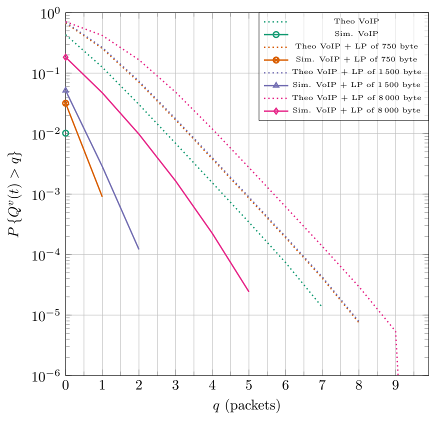


(a) No HP traffic

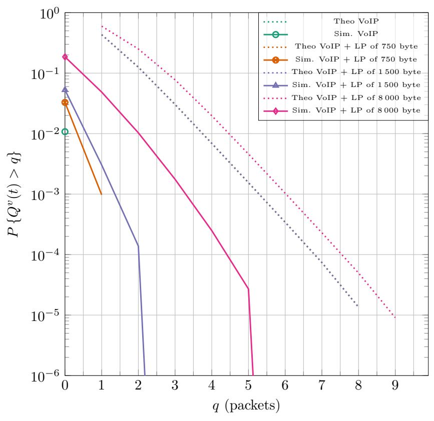


(b) HP: $\mathbf{A} = [(10, 68, 1), (10, 182, 1)]$

Figure 4.9: VoIP queue size distribution for $R = 5$ Mbps, $M = 20$



(a) No HP traffic



(b) HP: $\mathbf{A} = [(15, 1370, 1), (20, 3654, 1)]$

Figure 4.10: VoIP queue size distribution for $R = 100$ Mbps, $M = 40$

We can observe that the upper bound is not always close to the simulation results. This is due to the fact that the TB model can not exactly capture the arrival dynamics. For example, the black curve in Figure 4.11 is an example traffic envelope, while the purple curve is a possible token bucket descriptor for the black curve. We can see that the burst parameter b_a (i.e. the value of the curve at $\tau = 0$) must be large enough in order to be able to accommodate the long-term average arrival rate ρ_a.

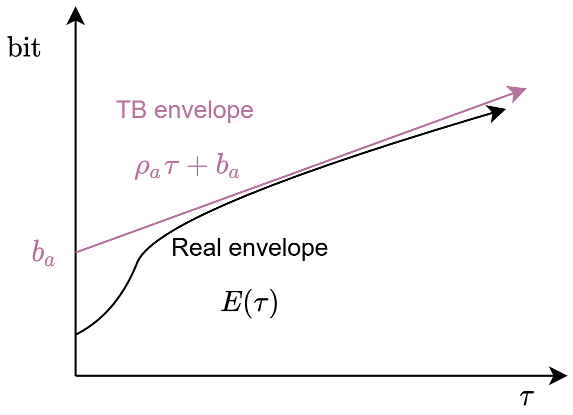
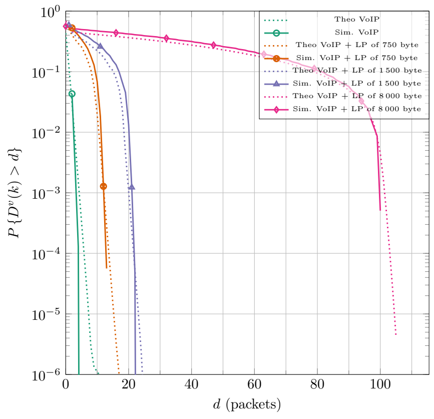


Figure 4.11: Traffic envelope vs. token bucket

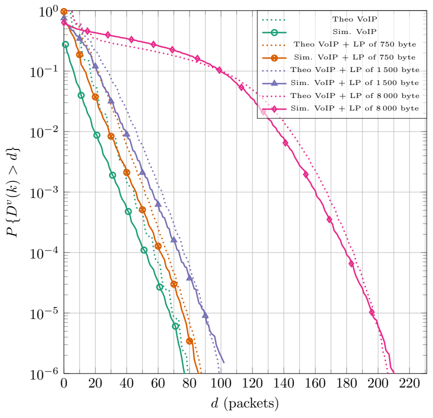
Within each plot, we can see that the relative error decreases with increasing vacation. In the network calculus primer, on Page 44, we defined the queue upper bound as $\hat{q} = b_a + \rho_a \frac{b_s}{\rho_s}$. As the vacation b_s increases, we can see that the emphasis becomes on the long term average arrival rate ρ_a, which provides a better approximation to the arrival envelope, resulting in a (relatively) more accurate bound.

4.7 Delay distribution

In Figure 4.12 we show the delay distribution for the same scenarios as for the queue size distribution. The curves are calculated using the delay bound obtained in the network calculus primer.

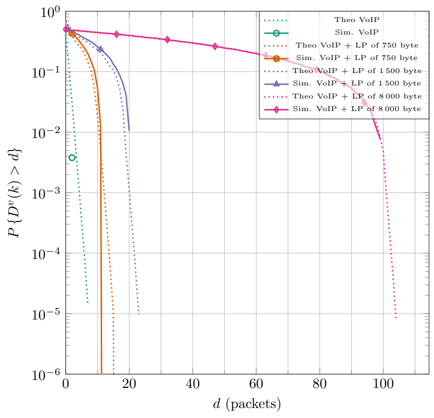


(a) No HP traffic

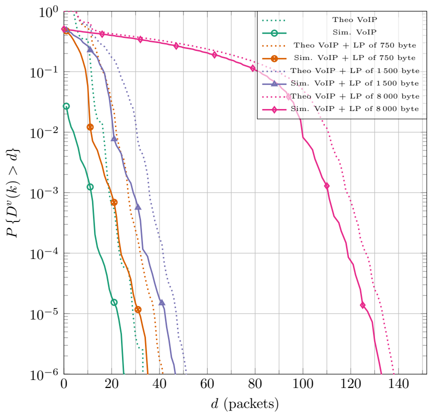


(b) HP: $\mathbf{A} = [(10, 68, 1), (10, 182, 1)]$

Figure 4.12: VoIP delay distribution for $R = 5$ Mbps, $M = 20$



(a) No HP traffic



(b) HP: $\mathbf{A} = [(15, 1370, 1), (20, 3654, 1)]$

Figure 4.13: VoIP delay distribution for $R = 100$ Mbps, $M = 40$

Here, we can observe that the upper bound is tighter than the bound for the queue size distribution. On Page 44, we also defined the delay upper bound as $\hat{d} = \frac{b_s}{\rho_s} + \frac{b_a}{\rho_s}$. Here we can see that for the delay, the dependency on the burst parameter b_a is much smaller than for the queue size, due to the factor $\frac{1}{\rho_s}$. As VoIP traffic is bursty, the initial burst b_a must then be large enough to accommodate for the worst case. Compared to the typical ρ_s, however, b_a is still small, and hence has little effect on the delay.

4.8 IPDV distribution

The IPDV distribution, shown in Figures 4.15 and 4.16 is in general quite accurate for all curves, except for the LP of 8 000 byte for $R = 5$ Mbps. At this rate, a single LP packet takes 0.0128s, a very large delay, and is more than the inter-arrival of the VoIP traffic. However, the bounds to the left and right seem to be quite representative of the simulation's IPDV.

A positive IPDV means that the delay of a packet was larger than previous packet's delay. In Figure 4.15a we see that the right bound for 8 000 byte sized LP packets caps off at 100 slots, the size of 1 LP packet, expressed in VoIP sized slots. This occurs when the LP starts transmission, and right afterwards a VoIP packet arrives, having to wait for the full LP packet to finish. Assuming that the previous delay was 0 slots, then the IPDV results in 100 slots.

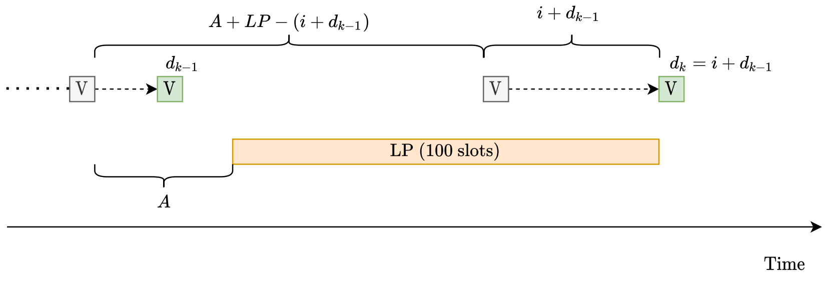
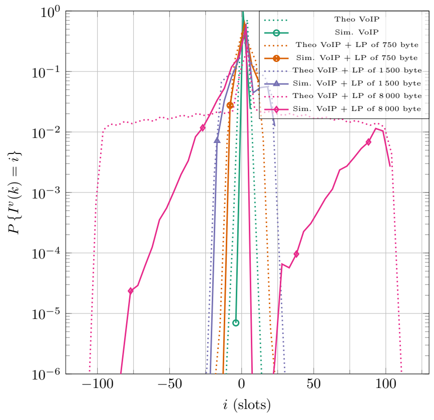


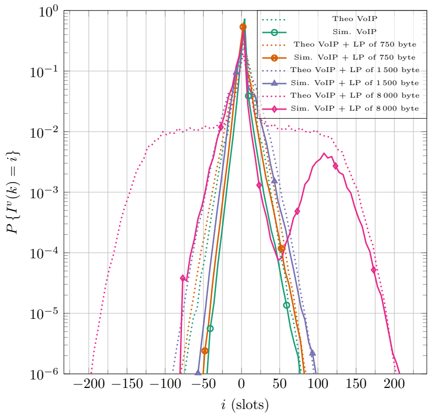
Figure 4.14: A positive IPDV

Another thing to notice is that the probability of an IPDV in the range $[1, 18[$ is very small (in the simulations it did not occur). In Figure 4.14 we show in a bit more detail how an IPDV of positive length i occurs. The rectangles with "V" inside are VoIP packets. The left green packet is the time at which the reference packet is sent, and has a delay d_{k-1}, while the right VoIP packet has a delay $d_k = i+d_{k-1}$. The interval between the arrival of the first VoIP packet and the start of the LP packet is denoted A. In the figure we can see that for a small IPDV i to occur, the interval over which no arrivals occur must be large. In the paper [37], which we referenced to in Section 4.3 to obtain the stochastic TB model, the transition matrix of the birth-and-death Markov Chain relating to the number of active VoIP sources was given. We are now only interested in the transition from 0 active sources to a positive number of active sources. This process is given by the exponential distribution, with parameter $M\beta$ (where M is the number of VoIP sources, and β the OFF state probability). The complementary cumulative distribution function (ccdf) is given by $\exp(-M\beta x)$. From this ccdf, we can see the probability decreases exponentially for longer intervals x. In our simulations intervals smaller than 18 slots (i.e. no arrivals from any VoIP source during more than 82 slots) were so unlikely that they did not occur.

When the rate R is large, in Figure 4.16, the amount of VoIP packets generated during the transmission of a LP packet is much smaller, and the IPDV behaves much more as expected.

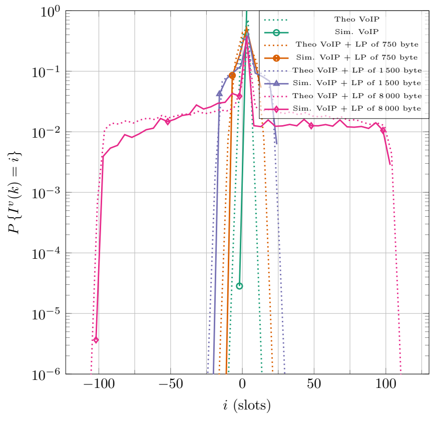


(a) No HP traffic

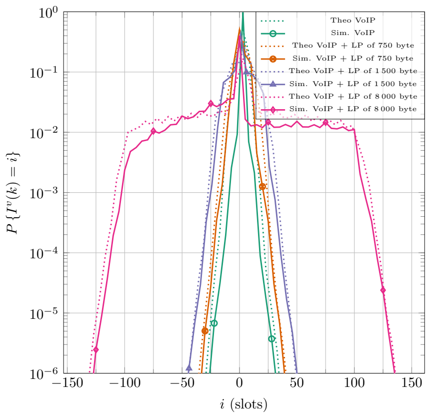


(b) HP: $\mathbf{A} = [(10, 68, 1), (10, 182, 1)]$

Figure 4.15: VoIP IPDV distribution for $R = 5$ Mbps, $M = 20$



(a) No HP traffic



(b) HP: $\mathbf{A} = [(15, 1370, 1), (20, 3654, 1)]$

Figure 4.16: VoIP IPDV distribution for $R = 100$ Mbps, $M = 40$

4.9 BP distribution

In this section we characterize the busy period distribution of the output of the VoIP traffic flows.

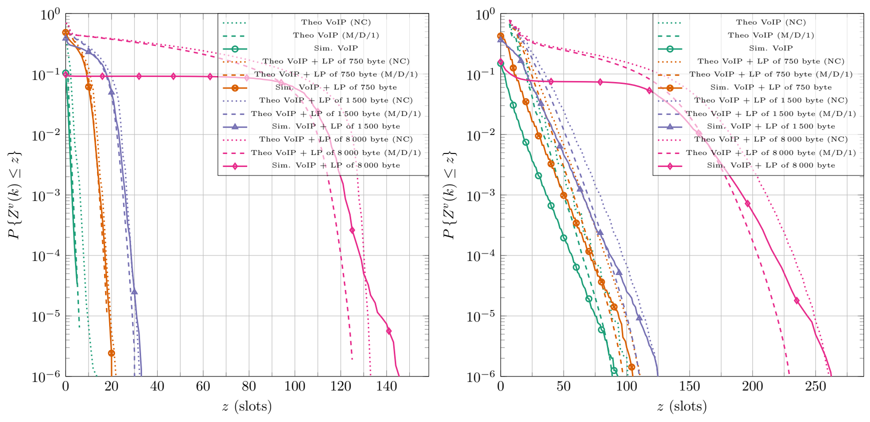
As we saw before, we can also model the aggregate of VoIP flows as an $M/D/1$

queue. This model has been around for a long time, hence a lot of literature is available that we can employ (see for example [78, 125, 126, 102, 124, 126, 24, 36, 89]).

The busy period distribution with vacation for an $M/D/1$ queue is described by the Borel-Tanner distribution [178]. Given two parameters, ρ and q, respectively the Poisson parameter and the initial queue occupancy, the busy period distribution is given by

$$P\left\{Z^{VoIP}(k)=z\right\}=\frac{q}{z}\frac{e^{-\rho z}(\rho z)^{z-q}}{(z-q)!}, z \geq q. \tag{4.20}$$

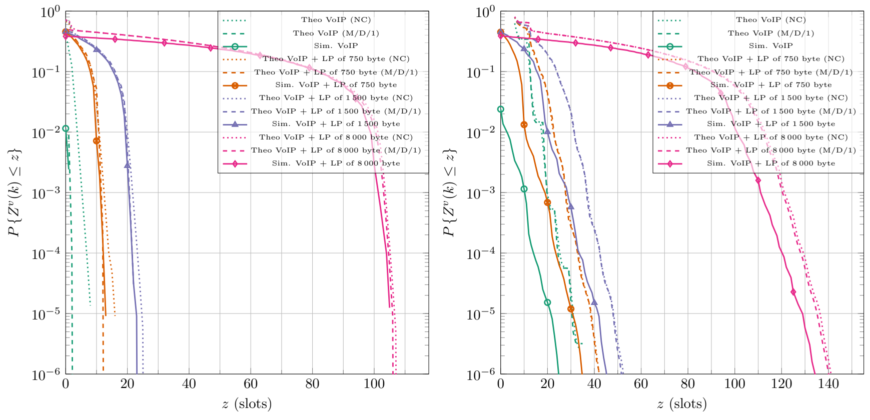
In Figure 4.17a we compare the busy period of the VoIP scheduler calculated using Equation (4.20) and with the SNC method. We can see there that both the $M/D/1$ and SNC models approximate the busy period distribution with vacation quite well, except for the very large low priority background packets and a low data rate. The $M/D/1$ model is often close to the simulation curve, but does not provide an upper bound. The SNC method, on the other hand, is often a little less close to the simulation curve, but provides an upper bound. For regular scenarios, both can thus be used to calculate the vacation period for lower priority traffic.



(a) No HP traffic

(b) HP: $\mathbf{A} = [(10, 68, 1), (10, 182, 1)]$

Figure 4.17: VoIP BP distribution for $R = 5$ Mbps, $M = 20$



(a) No HP traffic (b) HP: $\mathbf{A} = [(15, 1370, 1), (20, 3654, 1)]$

Figure 4.18: VoIP BP distribution for $R = 100$ Mbps, $M = 40$

4.10 Conclusion

In this section, we looked at an aggregate of VoIP traffic, coming in priority, after an aggregate of CBR sources. We first gave a brief introduction to network calculus in Section 4.2. Then we discussed the characteristics of a single VoIP flow in Section 4.3, and introduced a traffic model for an aggregate of VoIP flows. The model makes use of a token bucket, which gives a linear upper bound on the amount of arrivals over a certain interval. We also briefly compare the aggregate to a Poisson traffic model with fixed size packets. This model is convenient to use in the NC. In Section 4.4 we discussed the vacation that a VoIP packet can encounter. Then the subsequent sections discussed, as before, the queue size, delay, IPDV and busy period distributions.

Through simulations, we have found that the analytical results provide in general a good bound. For the queue size distribution, the bound is not so tight, due to the token bucket model that is used. The delay distribution is quite accurate, while the IPDV distribution does not follow the peaks for the extreme case of jumbo frame sized low priority packets. The domain of the IPDV distributions, however, match closely. The busy period, finally, can be approximated quite well, if the service rate is large enough or the size of the packets is not jumbo-sized.

Chapter 5

Low priority Video traffic

5.1 Introduction

In this section, we look at low priority traffic. The focus of this section is fragmentation. Hence, for this purpose, we use video traffic, which often comprises large frames, which are then fragmented into multiple packets once scheduled for transmission. We also use here the NC, discussed previously in Section 4.2. In Section 5.2 we discuss the vacation caused by the fragmentation, and then compare the results given by network calculus to the simulation results for the queue size (Section 5.4), delay (Section 5.5) and IPDV (Section 5.6) distributions.

For the NC we need a traffic arrival model. Here, to keep it simple, we extracted the traffic envelope for various violation probabilities from a trace file. Each traffic envelope was then converted to a linear bounded traffic envelope. Figure 5.1 shows these traffic envelopes.

5.2 The server vacation

In Figure 5.2 we see an example of fragmentation. The datagram is too large for the link, and hence must be fragmented into multiple packets, resulting in additional delays. The first extra delay is due to an additional header for each of the fragments. The data that is transmitted over the link is thus slightly more than the datagram's size. The second additional delay comes from the fact that each packet can encounter a busy channel. The datagram's first fragmented packet can encounter a vacation due to both high priority and low priority traffic. However, subsequent packets of the datagram can only have a vacation due to high priority traffic, as the video queue will be backlogged until the complete frame is sent.

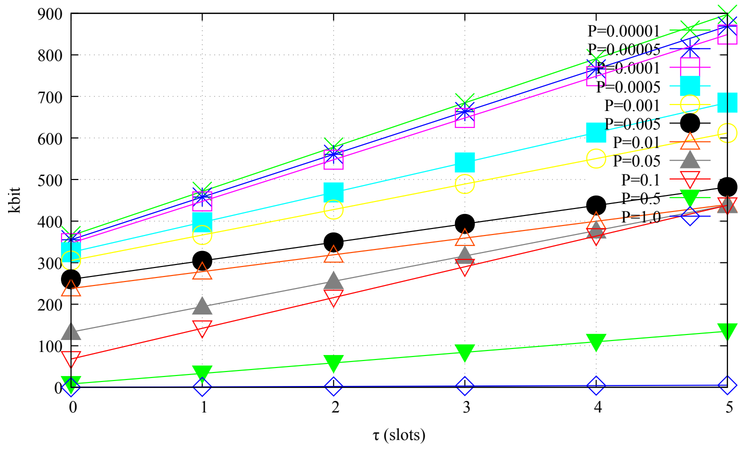


Figure 5.1: Traffic envelopes for the trace file for various violation probabilities

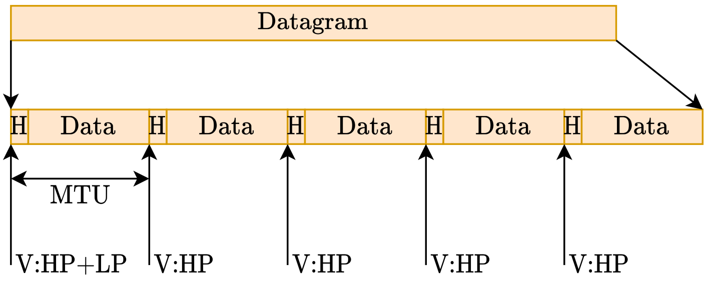


Figure 5.2: A datagram is fragmented into five packets, with a header and data

To calculate the vacation distribution a video frame encounters, we need to know the distribution of frame lengths. Video traffic in the networks considered often comprises CCTV traffic. As the imagery that is captured often has few changes from frame to frame, its rate is relatively constant [9], and the packet length distribution can be obtained in advance.

If we notate the frame length distribution by L and header size by H, then we can write the vacation as

$$V^{video} = \sum_i P\left\{L(k) \in [iU, (i+1)U[\right\} \cdot \left((V^{HP+LP}) \circledast (V^{HP^{\circledast i-1}}) \circledast \delta_{i \cdot H} \right)$$

where U is the MTU size, δ_x is the unit impulse (which shifts a distribution X by x when convoluted), and $X^{\circledast n}$ is the convolution power of X to the n-th. The busy period due to high priority and low priority traffic is given by V^{HP+LP}, while V^{HP} is the busy period due to high priority traffic only.

This approach does not take arrivals into account, and will lead to an upper bound as not every packet will encounter a vacation. Imagine, for example, that we have only five inter-site rapid response (ISRR) flows and one video flow serviced by a high data rate line. If a video frame is fragmented in 10 packets, then the ISRR busy period will be taken into account in each of those 10 packets, however, there are at most 5 ISRR packets that could lead to a vacation.

5.3 Simulation setup

The simulation setup for the video simulations of this section is shown in Figure 5.3. There are three queues that go into the SP scheduler, served by a channel of capacity $C = 10$ Mbps (scenarios 1 and 2) or $C = 100$ Mbps (scenarios 3 and 4). The first queue is for high priority traffic comprising an ISRR aggregate. The next queue is for the video flow, while the last queue is for low priority traffic. The low priority traffic has a load of about 50%, and sends packets of fixed size $U = 1\,500$ byte, and was only used in scenarios 2 and 4.

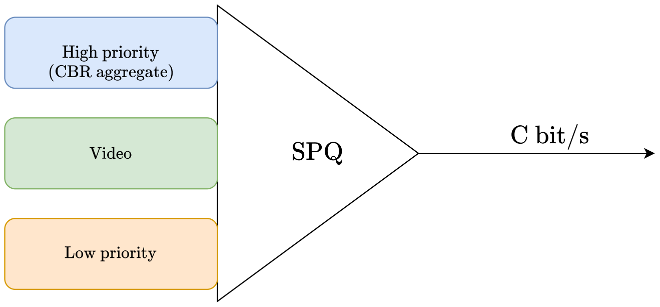
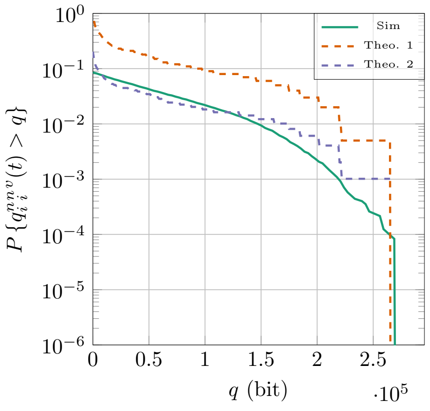


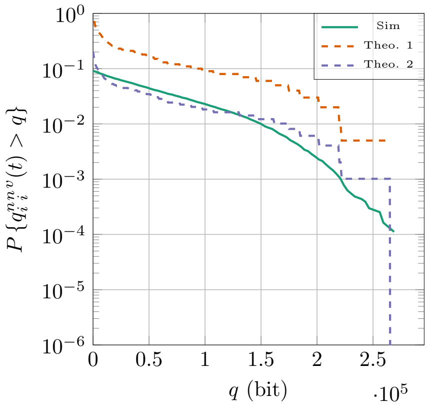
Figure 5.3: Scheduler setup for the simulations of this section

5.4 Queue size distribution

In the plots of this section the curves labeled "Theo. 1" come from the NC for the queue size distribution. The curves labeled "Theo. 2" are an optimization. The formula from SNC does not take the load into account, and $P\{Q^v(t) = 0\}$ is an overestimate. In the optimization, we fix $P\{Q^v(t) = 0\} = \rho_{HP} + \rho_{video}$, and normalize the distribution by scaling the other entries by $1 - \rho_{HP} + \rho_{video}$. For VoIP traffic in the previous section, this was not possible as the vacation due to LP traffic was relatively large. However, for video traffic, the vacation due to LP traffic is limited, compared to the video frame sizes.

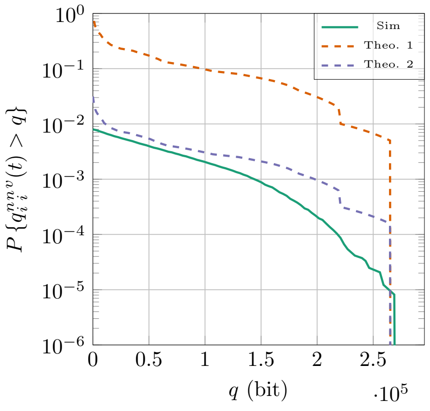


(a) Only ISRR and video

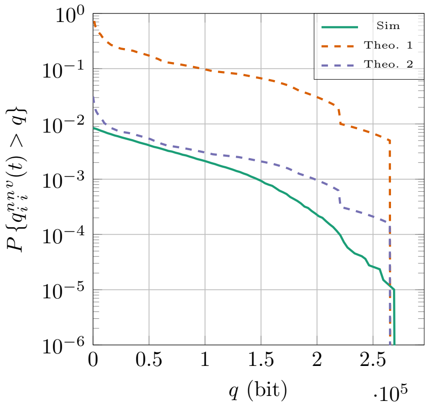


(b) ISRR, video and LP

Figure 5.4: Video queue size distribution for $R = 10$ Mbps, HP: $\mathbf{A} = [(10, 137, 1), (10, 265, 2)]$



(a) Only ISRR and video



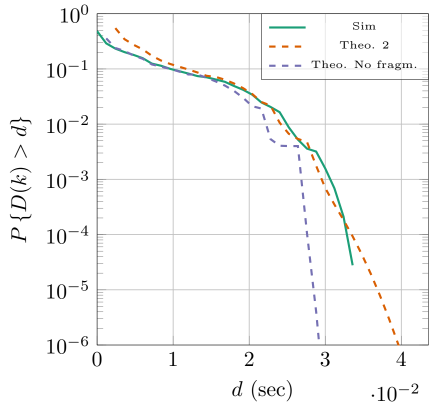
(b) ISRR, video and LP

Figure 5.5: Video queue size distribution for $R = 100$ Mbps, HP: $\mathbf{A} = [(20, 1370, 1), (20, 2653, 2)]$

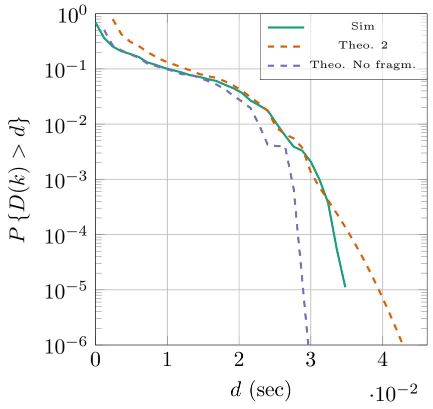
5.5 Delay distribution

In the plots of this section the curves labeled "Theo. 2" show the result where fragmentation is taken into account. The curves labeled "Theo. No fragm." are calculated by using the vacation $V = V^{HP+LP}$, i.e. ignoring fragmentation, and are provided for comparison. The delays reported for the video are the delays of the reassembled packets.

We can observe in the figures below that the fragmentation indeed is important to take into account to ensure that the tail of the curve is represented correctly. The impact of low priority traffic is negligible. Compared to the high priority traffic and the back-to-back packets of a frame, a single LP packet is relatively small, and hence will not increase the vacation too much.

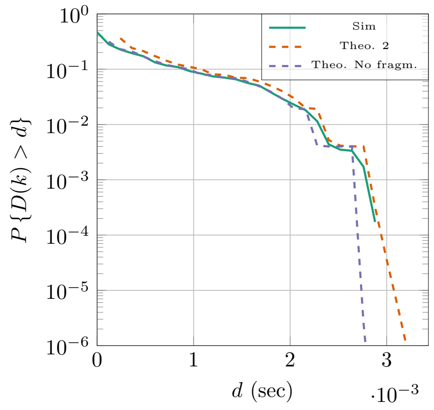


(a) Only ISRR and video

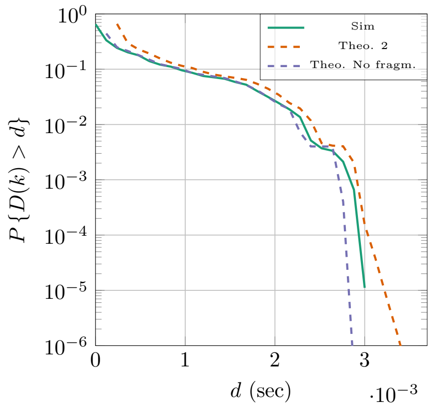


(b) ISRR, video and LP

Figure 5.6: Video delay distribution for $R = 10$ Mbps, HP: $\mathbf{A} = [(10, 137, 1), (10, 265, 2)]$



(a) Only ISRR and video



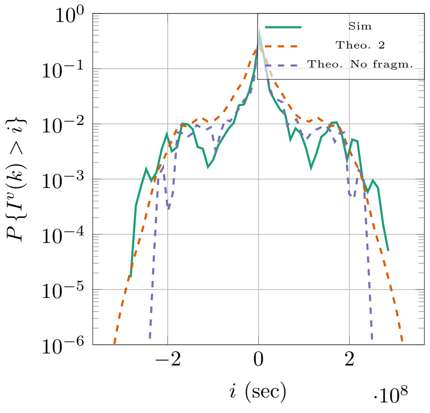
(b) ISRR, video and LP

Figure 5.7: Video delay distribution for $R = 100$ Mbps, HP: $\mathbf{A} = [(20, 1370, 1), (20, 2653, 2)]$

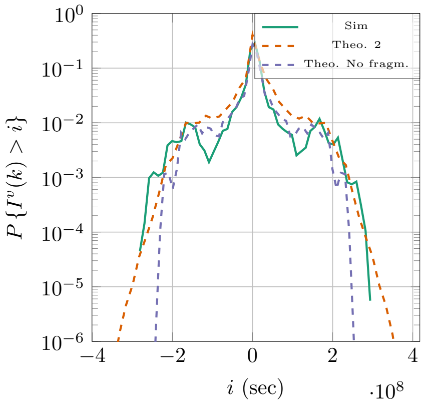
5.6 IPDV distribution

The naming of the curves is the same as for the delay distribution.

Also, here we can see that the distribution is followed closer for the "Theo. 2" curve than for the curve that does not take the fragmentation into account.

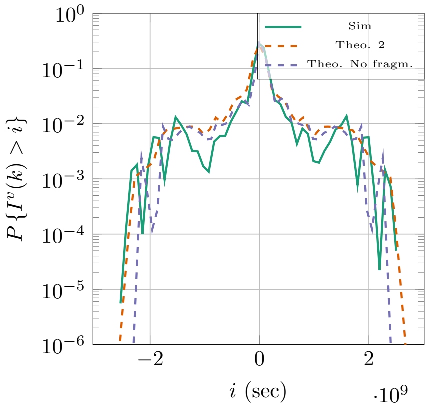


(a) Only ISRR and video

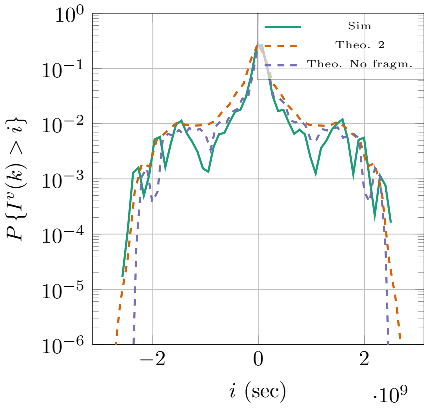


(b) ISRR, video and LP

Figure 5.8: Video IPDV distribution for $R = 10$ Mbps, HP: $\mathbf{A} = [(10, 137, 1), (10, 265, 2)]$



(a) Only ISRR and video



(b) ISRR, video and LP

Figure 5.9: Video IPDV distribution for $R = 100$ Mbps, HP: $\mathbf{A} = [(20, 1370, 1), (20, 2653, 2)]$

5.7 Conclusion

In this section we briefly discussed a single flow of video traffic. The main focus was on the impact of the fragmentation of large datagrams on the delay and IPDV, which usually occurs with traffic that sends large datagrams, such as video streams. In Section 5.2 we discussed the vacation that a video frame might encounter. Then the subsequent sections discussed the queue size, delay and IPDV distributions. For a single node, we are able to provide a nice bound on the delay and IPDV, given that we can obtain the distribution of video frame lengths, necessary to calculate how much vacation each packet of a frame will encounter.

Chapter 6
Evaluation

In this section, we perform simulations in a simulated IP network over a number of hops and with more realistic background traffic. First, in Section 6.1 we look at how multiple hops impact the E2E delay, and look at two cases. In the first case, where the load of the through-traffic is relatively low, compared to the cross-traffic and the through-traffic comprises relatively small packets, we can sum the delays in the individual nodes. The second case is more general, leading to a more complex formulation. In Section 6.2 we discuss the setup of the simulations. In Section 6.3 we give the practical details of all the traffic types we consider in the simulations. Section 6.4 gives a quick overview of a typical plot. Then, from Section 6.5 to Section 6.13 we describe and give the results for different scenarios. We then close this section in Section 6.14.

6.1 Multi-hop

6.1.1 Introduction

In this section, we discuss the delay behavior when packets cross multiple links, as is to be expected in a typical network. This section only discusses the model, while in the next section we will then apply the models to a simulation of subsequent nodes in an Ethernet network.

Figure 6.1 shows the sequence of nodes we consider. The traffic we are interested in enters the network in Node 1 (marked with "new traffic") and travels to Node $N+1$, over N links, with link $n \in [1, N]$ having a capacity R_n. We call this traffic, going from Node 1 to Node $N+1$, also the through-traffic. At the same time, there is also cross-traffic (indicated by the rectangles labeled "CT n"). The cross-traffic from node CT n enters in Node n and has the destination of Node $n+1$, and is discarded there. This introduces in each node a possible additional delay to our packets.

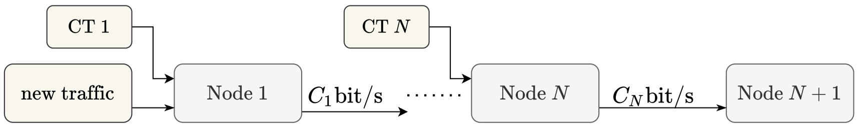


Figure 6.1: A simple network

We look at the delays for two different cases: one where there is only CBR and VoIP traffic going from the source to the destination (but with any type of cross-traffic in any of the nodes), and one where we additionally also have large low priority through-traffic packets, originating from video and data sources. These large through-traffic packets can introduce a large, fixed delay to HP traffic when traversing the links.

6.1.2 Without large through-traffic packets

In this simple case, a mix of CBR and VoIP flows go through a series of nodes, from node 1 to node $N+1$. We assume the load and packet sizes of these sources are small. As the aggregates move through the nodes with background traffic, they will lose their characteristics. Assuming that cross-traffic has a higher load than the through-traffic only travels over one hop (and thus in each hop there will be new traffic), we can ignore the impact of the through-traffic as it will be likely that a through-traffic packet will end up behind a cross-traffic packet, and we can use the single node delay. Hence, we can approximately model the E2E delay distribution D_{E2E} as the sum of the distributions encountered in each node, or more formally:

$$D^v_{E2E} = \mathop{\circledast}_{i=1}^{N} D^v_i,$$

where D^v_i is the delay encountered in node i, if it were to start in that node.

6.1.3 With large background through-traffic packets

6.1.3.1 Introduction

When we have background through-traffic, usually comprising large packets, we can encounter a situation as shown in Figure 6.2. In this figure, we model time on the horizontal axis, and on the vertical axis we show the packets present in a node. Each node is separated by a dashed line. A gray packet indicates that the packet has arrived but has to wait in the queue. It has an arrow to the moment it is sent. A colored packet indicates its transmission.

In Node i we can observe that a large low priority packet, called LP1 and colored orange, is being sent, right before the blue high priority packet arrives. As there

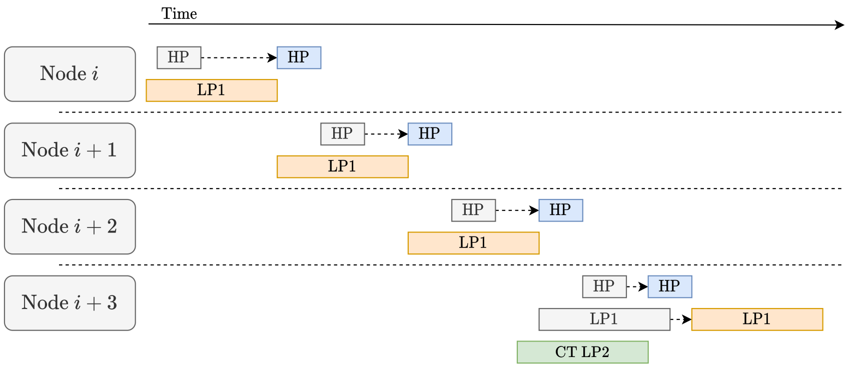


Figure 6.2: The HP packet is stuck behind LP1, until Node 3

is no preemption, this high priority packet has to wait until LP1 has finished transmission. When the high priority packet arrives in Node $i+1$, it again has to wait for LP1 to finish its transmission, this time HP has to wait a fixed time of the length of $LP1-1$ (assuming 1 HP packet takes 1 slot to send). This continues, until either the packets reach their destination, or, as shown in Node $i+3$, a cross-traffic packet, called CT LP2 here, is in transmission as LP1 arrives. If this delays the transmission of the LP1 packet more than one slot, then the high priority packet can jump in front of LP1. In subsequent nodes, the high priority packet can again get stuck behind large through-traffic packet, repeating the above process. Similarly, a low priority through-traffic packet can also encounter a similar enduring delay due to the busy period of the high priority traffic.

We can see that the delay of the packet depends on what happened in the previous node. There is thus no independence, and we can not use a convolution to obtain the E2E delay, like in Section 6.1.2 where the absence of large packets limits the extra delay drastically.

6.1.3.2 Notation

We first define the symbols used. We denote the MTU as U. We assume the MTU is the same for all links. Define L as the distribution of packet lengths, expressed in slots. This might be further specialized using a superscript, e.g. $L_i^{LP,TT}$ indicates the distribution of packet lengths for (the aggregate of) low priority through-traffic traffic, in node i. The notion of slots, low priority and high priority of course depends on the traffic class under consideration. Furthermore, the load of the i-th channel is indicated by ρ_i. Also here the load might be further specialized through a superscript. E.g. $\rho_i^{HP,TT}$ is the load of the through-traffic of a priority equal to or higher than the traffic class in consideration, while $\rho^{LP,CT}$ is the load of the cross-traffic of strictly lower priority. We make again use of the distribution

$$\mathcal{L}(X) = \sum_{x\in X} P\{X = x\}\mathcal{U}_{[0,x]},$$

which was also defined previously in (4.18), on Page 50. It is used to e.g. calculate the vacation for a packet due to low priority traffic:

$$V^{LP} = \mathcal{L}(L^{LP}).$$

The vacation due to high priority traffic, denoted by V^{HP}, is taken into account by using the busy period of the high priority traffic. The symbol $V^{v,HP}$ is the vacation due to high priority traffic, including the vacation due to low priority traffic.

Finally, let V^{HP+LP} denote the vacation if a packet Pk arrives during a packet (and it didn't encounter a through-traffic packet in the previous node). This can be written as the weighted sum of the busy period distributions of the high priority and low priority traffic.

$$\begin{aligned} P\left\{V^{HP+LP}(k) = x\right\} = P\left\{V^{v,HP}(k) = x\right\}\cdot\frac{\rho^{HP}(1+\rho^{LP})}{\rho}+ \\ P\left\{V^{LP}(k) = x\right\}\cdot\frac{\rho^{LP}\cdot((1-\rho)+\rho^{LP})}{\rho}. \end{aligned}$$

These weights are explained as follows. The probability of a packet arriving during a high priority busy period can be written as $\rho^{HP}(1+\rho^{LP})$: it is the probability of arriving during either a high priority packet, or during a low priority packet which is immediately followed by a high priority packet. Similarly, the probability of a packet arriving during a low priority packet that does not start a high priority busy period is given by $\rho^{LP}\cdot((1-\rho)+\rho^{LP})$. Both factors are divided by ρ as we condition on the fact that there is a packet in transmission when Pk arrives.

6.1.3.3 Probabilities and vacations

In this section, we derive the vacation. If we observe a single packet Pk in a node, it can encounter following important scenarios. The packet Pk can arrive in node i

- $\mathbf{0_i}$ in an idle channel, and can be transmitted immediately;
- $\mathbf{A_i}$ during the transmission of a cross-traffic packet;
- $\mathbf{B_i}$ during a LP through-traffic packet, where the LP through-traffic packet is the same as in node $i-1$;
- $\mathbf{C_i}$ during a LP through-traffic packet, where the LP through-traffic packet is not the same as in node $i-1$;

D_i during a HP busy period, where the HP busy period is the same as in node $i-1$;

E_i during a HP busy period, where the HP busy period is not the same as in node $i-1$.

As mentioned before, there is a dependence on the previous node (present in scenarios B-E), and thus we must consider two nodes at once, i.e. for a packet Pk in node i, we look at all possible permutations that can occur over the tuple (node $i-1$, node i).

Table 6.1: Permutations of all scenarios in a single node

Nodes	P	V	Comments
{00,A0}	$(1-\rho)$	0	
{0A,AA}	ρ^{CT}	V^{HP+LP}	
{0B,AB}	0	-	No through-traffic (TT) packet in previous node
{0C,AC}	$\rho^{LP,TT}$	$\mathcal{L}(L^{LP,TT})$	
{0D,AD}	0	-	No TT packet in previous node
{0E,AE}	$\rho^{HP,TT}$	V^{HP}	
{B0,C0}	0	-	A TT packet can not disappear
{BA,CA}	ρ^{CT}	V^{HP+LP}	
{BB,CB}	$(1-\rho)$	$L^{LP,TT} \circledast \delta_{-1}$	Negative values should be removed from V
{BC,CC}	$\rho^{LP,TT}$	$\mathcal{L}(L^{LP,TT})$	
{BD,CD}	0	-	A TT LP packet can not overtake a TT HP packet
{BE,CE}	$\rho^{HP,TT}$	V^{HP}	
{D0,E0}	0	-	A TT HP can not disappear
{DA,EA}	ρ^{CT}	V^{HP}	
{DB,EB}	0	-	No HP TT BP in previous node
{DC,EC}	$\rho^{LP,TT}$	V^{HP}	
{DD,ED}	$(1-\rho)$	$\mathcal{L}(V^{HP}) \circledast \delta_{-1}$	Negative values should be removed from V
{DE,EE}	$\rho^{HP,TT}$	V^{HP}	

In Table 6.1 we list all these permutations. The first column of the table lists the scenarios for the row. E.g. a scenario “AC” means that in the node $i-1$ the packet Pk arrived during the transmission of a cross-traffic packet (A), and in the current node Pk arrives during a low priority through-traffic packet (C). The second column lists the probability of this scenario occurring. The sum of the probabilities of each block of 6 rows must sum to 1. The third column of the table gives (an upper bound on) the distribution of the vacation the packet Pk will encounter. A vacation 0 indicates no vacation, while the vacation will be “-” if the corresponding probability is 0.

We can now generate for N links all possible permutations. For each permutation j, we then can calculate a list ps_j of N probabilities, and a list vs_j of N vacation distributions. The probability that permutation j is taken is then simply

$$p_j = \Pi_{p \in ps_j}(p),$$

with associated vacation distribution

$$v_j = \underset{v \in vs_j}{\circledast} v.$$

The vacation is then the weighted sum of the distributions:

$$V = \sum_j p_j v_j.$$

In Appendix B this algorithm is implemented in pseudocode.

In Table 6.1 we can also see that if the cross-traffic load is high, we are more likely to always encounter scenario A. In this case, indeed, a packet is less likely to be stuck behind a low priority through-traffic packet, and we can reduce it to the simpler case from Section 6.1.2.

If the load $\rho^{LP,TT}$ is low, then scenarios B and C are unlikely to occur. Nonetheless, if the load ρ is also low, then the impact of scenario BB or CB (where packet Pk remains behind the same low priority through-traffic packet) is relatively high. The only solution to reduce the impact is to make the maximal length of $L^{LP,TT}$ small, such that $L^{LP,TT} \circledast \delta_{-1}$, and the additional delay, becomes smaller.

6.2 Setup

To assess the results from previous sections, we have run multiple simulations. The simulation is written in C++ using the OMNeT++ simulator [4] in combination with the INET libraries [2]. Custom elements were written to generate the traffic and provide additional metrics, such as the busy period and the sampled queue size distribution. An IP network is simulated over which UDP packets are sent. The simulations were run for a total of about 4 hour, by repeating a 5-minute simulation 50 times.

We use the same network here as shown in Figure 6.1. In Figure 6.3 we show the elements in a single node. A node k has 3 inputs. The first input is the aggregate of all traffic that comes from the previous node. The traffic from this input is checked for its destination: if it matches the current node, it is discarded and the metrics are stored, if it is the traffic we are interested in. After the destination check, it is fed into the classifier, which looks at the DSCP marker of the packet, and sends it to the appropriate queue.

The second input is for new traffic that we are interested in. It is only used in the first node. The third input is for cross-traffic. This type of traffic is not of any interest to us, but is introduced to ensure that there will be additional queuing in every node. Traffic from the second and third inputs are entering the network, and thus must be marked with the appropriate DSCP marker. This marking happens in our implementation based on the destination port, as each traffic class has a fixed port range. Traffic that comes from the previous node is already marked, and hence can go directly to the classifier.

The classifier redirects the packets to the correct queue. All queues are directed into a strict priority scheduler, and then served by a channel to the next node.

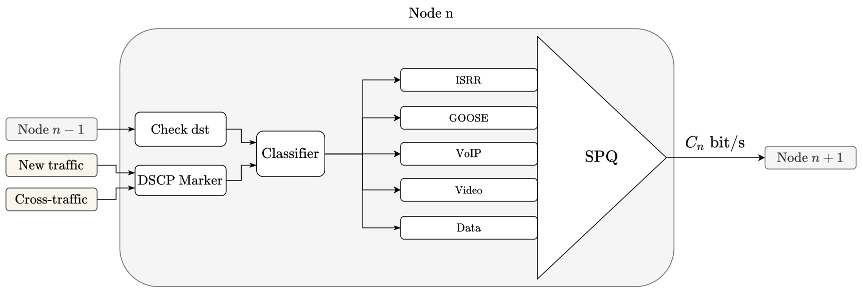


Figure 6.3: A single node

6.3 Traffic

In this section we briefly discuss the traffic used in the simulations.

6.3.1 ISRR

The ISRR traffic is a CBR source that produces every 50 ms a packet of 414 byte, resulting in a data rate of approximately 66 kbps. An aggregate of ISRR sources repeats itself after some time. As we are not interested in a single configuration, and to avoid the overhead of starting a new simulation for every different configuration, every 10 packets we introduce a waiting time that is chosen uniformly from the interval $[0 \text{ ms}, 150 \text{ ms}]$.

6.3.2 GOOSE

The generic object oriented substation events (GOOSE) traffic is a CBR source that produces every 200 ms a packet of 900 byte, resulting in a data rate of 36 kbps per source. As with the ISRR, also here we introduce a random waiting time every 10 packets.

6.3.3 VoIP

The VoIP sources used in these simulations are the same as described in Chapter 4. A single VoIP source generates in the ON state fixed packets of 80 byte (including all headers) every 10 ms, and thus has a peak rate of $\lambda = 64$ kbps, but generates on average 22.4 kbps.

6.3.4 Video

This data source is typically produced by something like a CCTV, however, here, we reproduce a part from Starwars video, at a rate of 33 fps. The video has an average arrival rate of 754 kbps and a peak arrival rate of 10 942 kbps. A video frame can exceed the MTU, in which case it will be fragmented. The delays reported for the video are the delay of the video frames (i.e. reassembled packets).

6.3.5 Data

This data source represents random background traffic. Each packet has a length that is uniformly distributed in the range $[100 \text{ byte}, U]$, where U is the MTU. For scenarios 1 to 5 (i.e. Section 6.5 to Section 6.9), the inter-arrival time is chosen uniformly from the range $[10 \text{ ms}, 50 \text{ ms}]$ for $U = 1\,500$ byte, and from the range $[53 \text{ ms}, 266 \text{ ms}]$ for $U = 8\,000$ byte. This results in an average data rate of approximately 400 kbps.

For the scenarios 6 to 9 (i.e. Section 6.10 to Section 6.13), which feature an increased load, the inter-arrival time is chosen uniformly from the range $[1 \text{ ms}, 10 \text{ ms}]$ for $U = 1\,500$ byte, and from the range $[5.3 \text{ ms}, 53 \text{ ms}]$ for $U = 8\,000$ byte. This results in an average data rate of approximately 2.1 Mbps.

6.4 Plot layout

A typical plot for this section is shown in Figure 6.4.

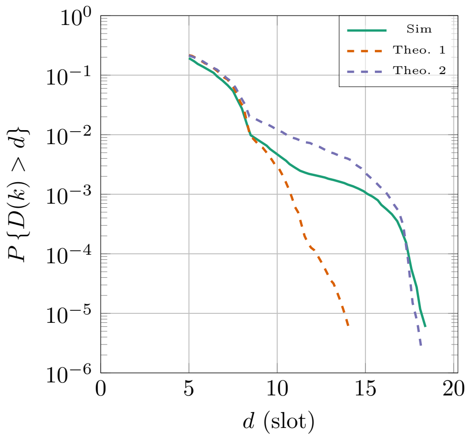


Figure 6.4: A sample plot of the E2E distribution

The E2E delay, queue and busy period plots are ccdf plots, while the E2E IPDV plot is a pmf. Hence, the y-axis represents respectively $P\{X > x\}$ or $P\{X = x\}$. The y-axis follows a logarithmic scale. The value on the x-axis is shown in units of packets for that traffic class (except for video, which uses SI units). I.e. for the E2E delay and IPDV and busy period this is the time required to transmit one packet of that type over a single link, and for the queue size distribution this is the number of packets in the queue. The solid green curve is the result from the simulation, while the dashed curves come from the theoretical results, described in the previous sections. The curves labeled "Theo. 1" are the result of the simple convolution where no background through-traffic is assumed (see Section 6.1.2), while the curves labeled "Theo. 2" are calculated using the approach of Section 6.1.3.

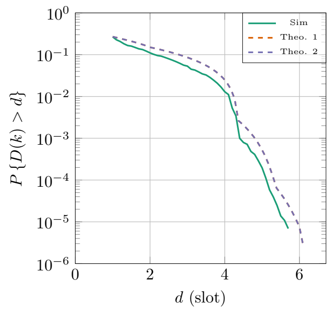
6.5 Scenario 1

The first scenario is quite simple, allowing for comparison with the results from the previous sections in a more realistic setting. There are two nodes, with 1 link of 10 Mbps, over which we run traffic. Table 6.2 lists the parameters. The n-th number after the slash indicates the amount of cross-traffic sources in the n-th node. In this case, there is only one node, hence we assume no cross-traffic.

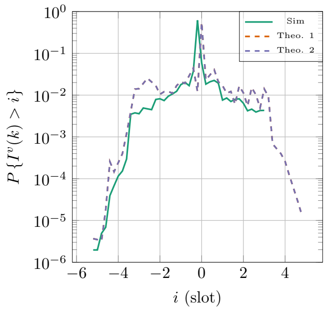
Table 6.2: Simulation parameters for scenario 1

Parameter	Value	Parameter	Value
N	1	ISRR	10 / 0 0 0 0 0
$\mathbf{R}$	$[0.01] \cdot$ Gbps	GOOSE	10 / 0 0 0 0 0
MTU	1 500 byte	VoIP	20 / 0 0 0 0 0
$\boldsymbol{\rho}$	[0.28]	Video	1 / 0 0 0 0 0
		Data	1 / 0 0 0 0 0

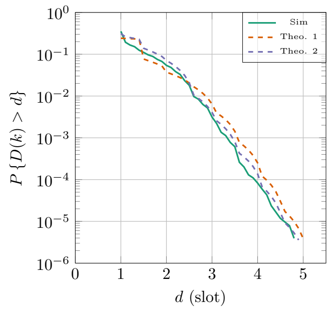
We can see that the delay and IPDV are quite well bounded by our algorithms. The curves labeled "Theo. 2" are slightly more accurate, providing a better upper bound. This is due to the fact that the combinations of an arrival during HP, LP or LP+HP are explicitly taken into account.



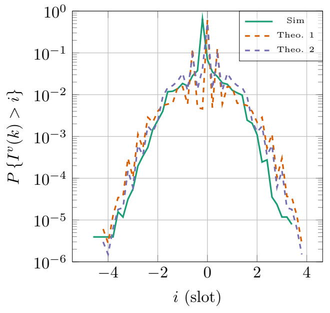
(a) ISRR E2E



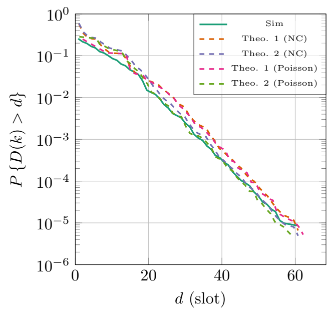
(b) ISRR E2E IPDV



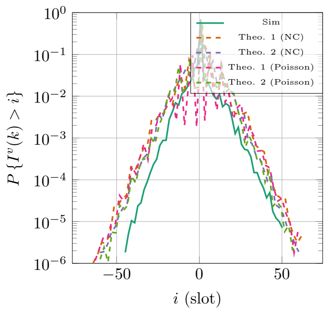
(c) GOOSE E2E



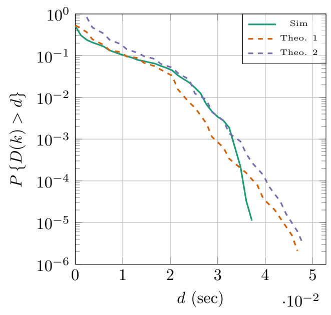
(d) GOOSE E2E IPDV



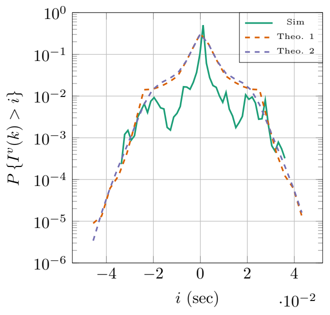
(e) VoIP E2E



(f) VoIP E2E IPDV



(g) Video E2E



(h) Video E2E IPDV

Figure 6.5: E2E results for scenario 1

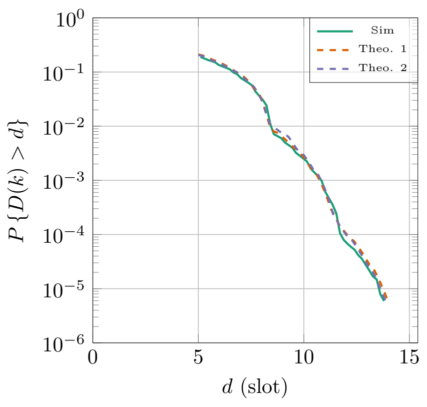
6.6 Scenario 2

In this scenario, there are 5 links of 1 Gbps. On each link there is cross-traffic of all traffic types. There is no video or data through-traffic. Table 6.3 lists the parameters.

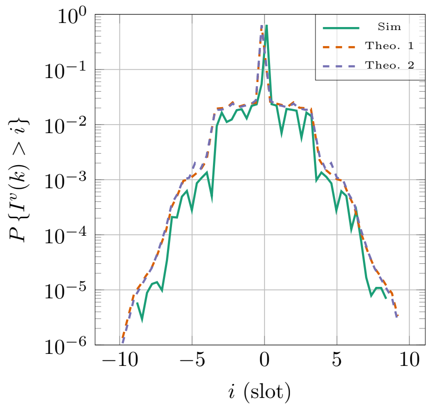
Table 6.3: Simulation parameters for scenario 2

Parameter	Value	Parameter	Value
N	5	ISRR	10 / 5 10 15 20 25
$\mathbf{R}$	[1,1,1,1,1] $\cdot$ Gbps	GOOSE	15 / 25 20 15 10 5
MTU	1 500 byte	VoIP	50 / 40 30 20 30 40
$\boldsymbol{\rho}$	[0.03, 0.04, 0.03, 0.04, 0.06]	Video	0 / 20 30 20 40 50
		Data	0 / 40 20 30 20 40

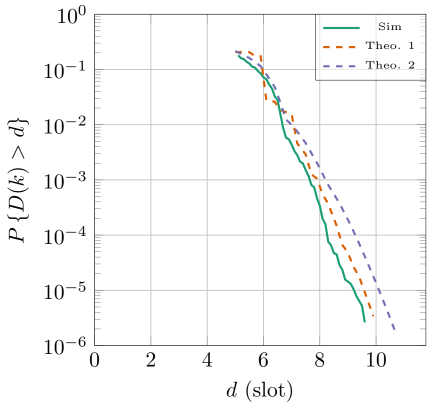
With no large background TT packets, the "Theo. 1" and "Theo. 2" curves are performing quite similar. The IPDV curves for the "Theo. 1" curves have a bit more spikes, due to the sudden jumps in the delays. They occasionally also fall below the simulation curve. The curves calculated using Poisson slightly lower, most of the time, but it is not sure if this is an upper bound.



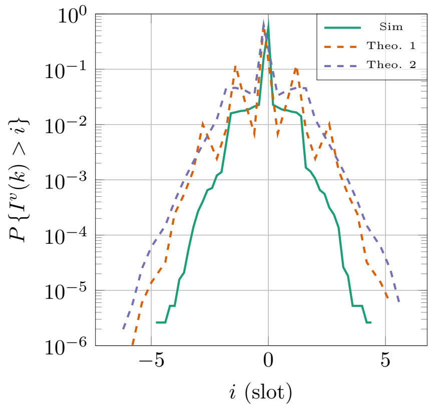
(a) ISRR E2E



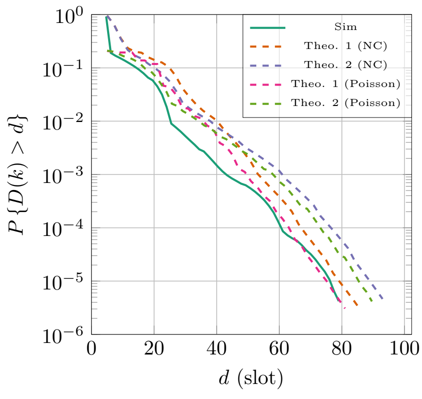
(b) ISRR E2E IPDV



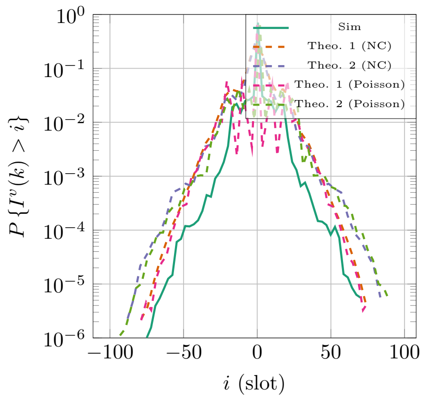
(c) GOOSE E2E



(d) GOOSE E2E IPDV



(e) VoIP E2E



(f) VoIP E2E IPDV

Figure 6.6: E2E results for scenario 2

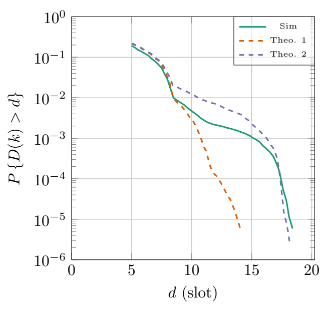
6.7 Scenario 3

This scenario is the same as scenario 2 (Section 6.6), except now there is video and data through-traffic. Thus, in this case the delay calculated as in Section 6.1.3 should provide a better bound. Table 6.4 lists the parameters.

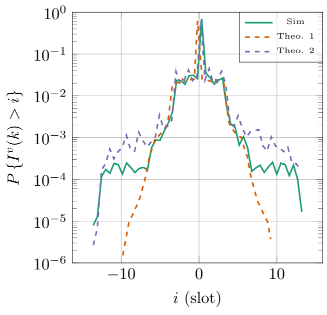
Table 6.4: Simulation parameters for scenario 3

Parameter	Value	Parameter	Value
N	5	ISRR	10 / 5 10 15 20 25
$\mathbf{R}$	[1,1,1,1,1] $\cdot$ Gbps	GOOSE	15 / 25 20 15 10 5
MTU	1 500 byte	VoIP	50 / 40 30 20 30 40
$\boldsymbol{\rho}$	[0.04, 0.04, 0.03, 0.05, 0.06]	Video	1 / 20 30 20 40 50
		Data	2 / 40 20 30 20 40

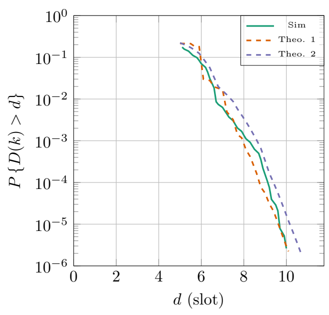
In these scenarios, we can indeed clearly see the need to take the TT packets into account. The large low priority video and data packets significantly increase the delays for ISRR and VoIP. The "Theo. 2" curves follow the shape of the delays, but we overestimate the probability a bit This also clearly impacts the IPDV, where "Theo. 2" provides the better bound.



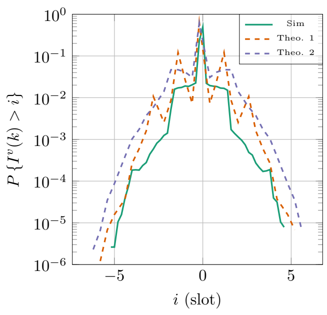
(a) ISRR E2E



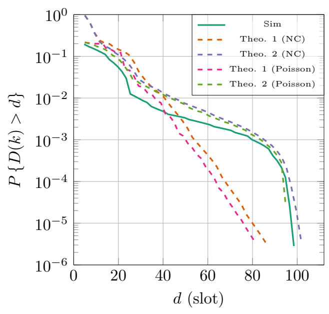
(b) ISRR E2E IPDV



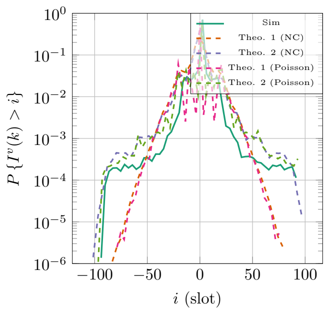
(c) GOOSE E2E



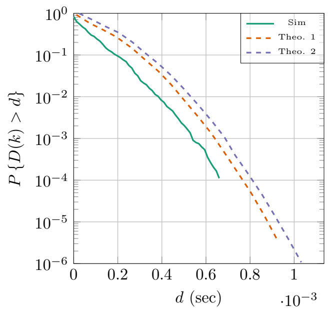
(d) GOOSE E2E IPDV



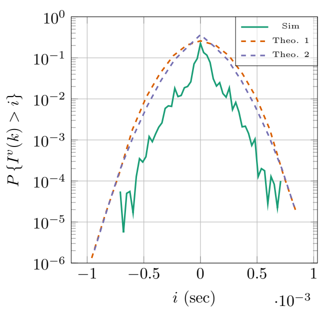
(e) VoIP E2E



(f) VoIP E2E IPDV



(g) Video E2E



(h) Video E2E IPDV

Figure 6.7: E2E results for scenario 3

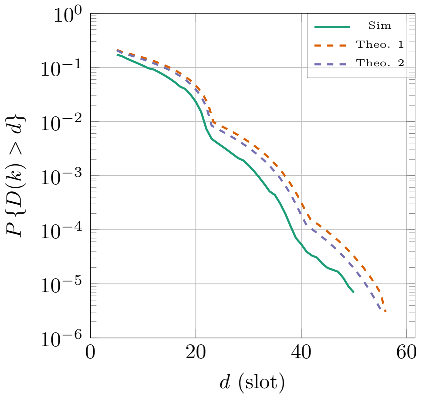
6.8 Scenario 4

In this scenario, we use the same parameters as in scenario 2 (Section 6.6), however, now the MTU is set to 8 000 byte. Thus, the video and data cross-traffic can now put packets of up to 8 000 byte on the line. Table 6.5 lists the parameters.

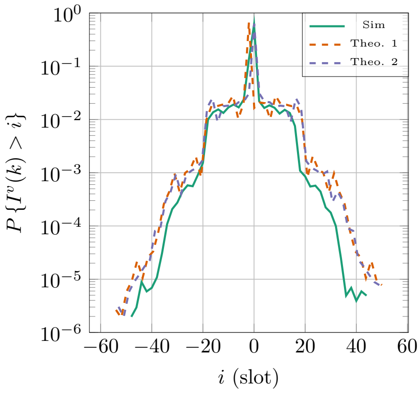
Table 6.5: Simulation parameters for scenario 4

Parameter	Value	Parameter	Value
N	5	ISRR	10 / 5 10 15 20 25
$\mathbf{R}$	$[1,1,1,1,1] \cdot$ Gbps	GOOSE	15 / 25 20 15 10 5
MTU	8 000 byte	VoIP	50 / 40 30 20 30 40
$\boldsymbol{\rho}$	[0.03, 0.04, 0.03, 0.04, 0.06]	Video	0 / 20 30 20 40 50
		Data	0 / 40 20 30 20 40

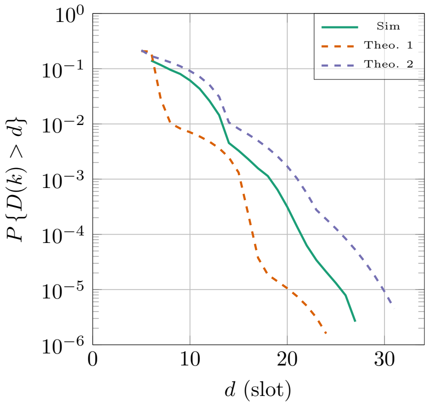
For jumbo frames, the bounds are not as close as for a regular MTU size. The "Theo. 1" method accurately represents the ISRR traffic class, however, the lower priorities such as GOOSE and VoIP are not accurate. This is due to the fact that the vacation for "Theo. 1" only comprises the HP traffic, with optional LP, while a packet can also arrive during the arrival of a LP packet, not followed by a high priority packet, which also increases a packet’s delay.



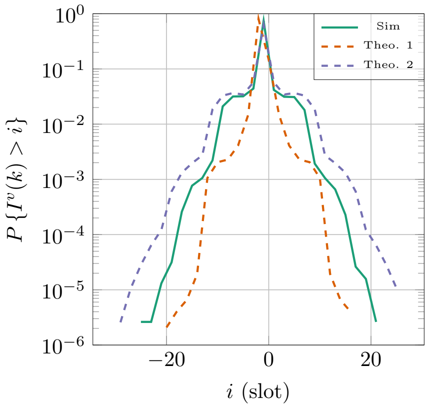
(a) ISRR E2E



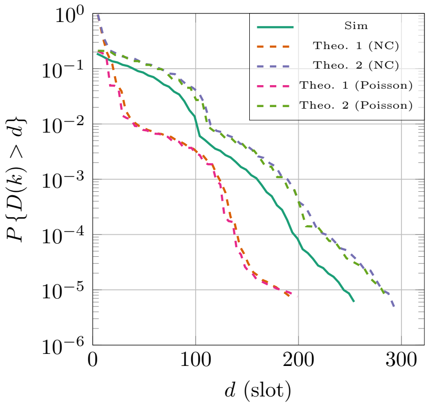
(b) ISRR E2E IPDV



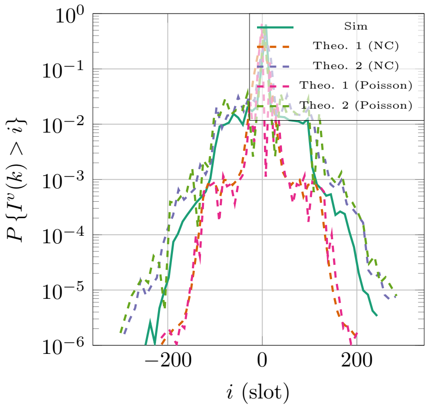
(c) GOOSE E2E



(d) GOOSE E2E IPDV



(e) VoIP E2E



(f) VoIP E2E IPDV

Figure 6.8: E2E results for scenario 4

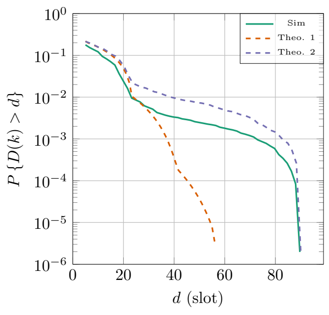
6.9 Scenario 5

In this scenario, we use the same parameters as in scenario 3 (Section 6.7), but, like in the previous scenario, we increase the MTU to 8 000 byte. Now also the through-traffic video and data traffic have much larger packets. Table 6.6 lists the parameters.

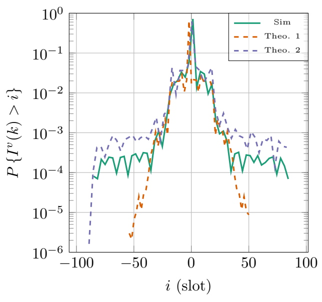
Table 6.6: Simulation parameters for scenario 5

Parameter	Value	Parameter	Value
N	5	ISRR	10 / 5 10 15 20 25
R	[1,1,1,1,1] · Gbps	GOOSE	15 / 25 20 15 10 5
MTU	8 000 byte	VoIP	50 / 40 30 20 30 40
$\boldsymbol{\rho}$	[0.04, 0.04, 0.03, 0.05, 0.06]	Video	1 / 20 30 20 40 50
		Data	2 / 40 20 30 20 40

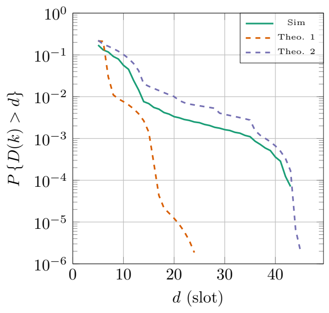
For jumbo frames, in combination with through-traffic video and data traffic, the difference between the simple method becomes very clear for ISRR, GOOSE and VoIP traffic aggregates. The "Theo. 1" method for the video traffic is much closer to the delay distribution of the simulation, as the length of the LP data packets is much smaller, compared to the video frame sizes.



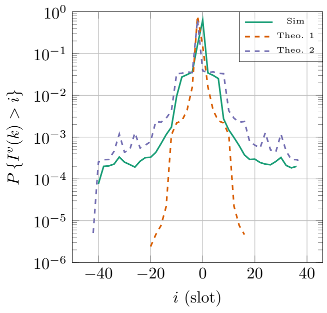
(a) ISRR E2E



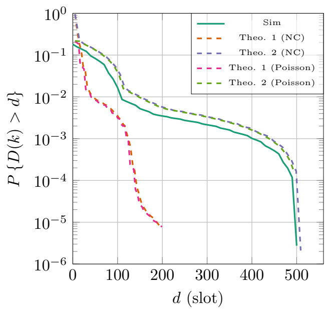
(b) ISRR E2E IPDV



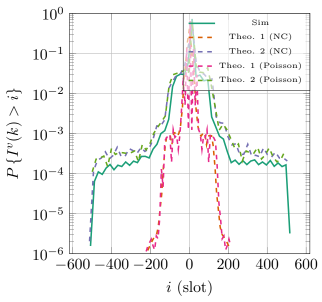
(c) GOOSE E2E



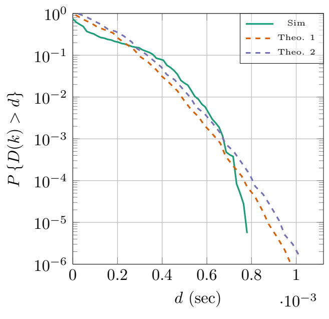
(d) GOOSE E2E IPDV



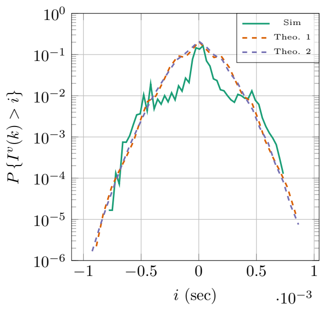
(e) VoIP E2E



(f) VoIP E2E IPDV



(g) Video E2E



(h) Video E2E IPDV

Figure 6.9: E2E results for scenario 5

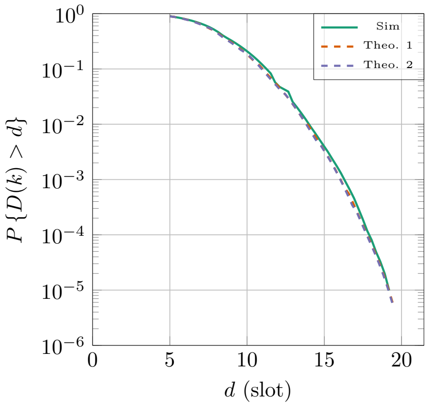
6.10 Scenario 6

This scenario is similar to scenario 2 (Section 6.6), but now we have increased the amount of data flows, increasing the load significantly. Table 6.7 lists the parameters.

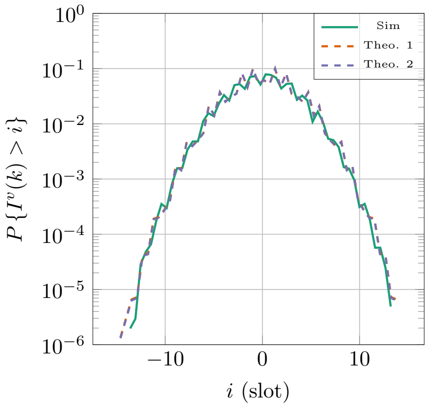
Table 6.7: Simulation parameters for scenario 6

Parameter	Value	Parameter	Value
N	5	ISRR	10 / 5 10 15 20 25
$\mathbf{R}$	[1,1,1,1,1] $\cdot$ Gbps	GOOSE	15 / 25 20 15 10 5
MTU	1 500 byte	VoIP	50 / 40 30 20 30 40
$\boldsymbol{\rho}$	[0.45, 0.24, 0.34, 0.25, 0.47]	Video	0 / 20 30 20 40 50
		Data	0 / 200 100 150 100 200

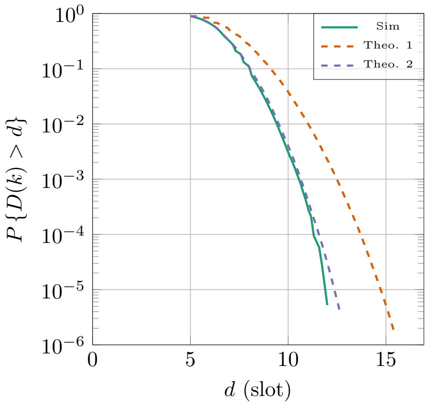
The high load increases the predicted delays for the simple method "Theo. 1", while the distribution for the "Theo. 2" method closely follows the distributions for all traffic classes. As the high load is due to the cross-traffic (CT) traffic, this reduces the impact of the TT traffic on the other priorities.



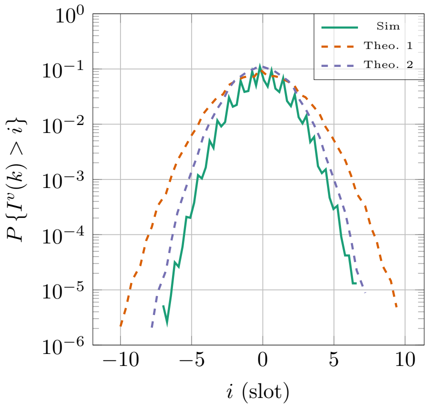
(a) ISRR E2E



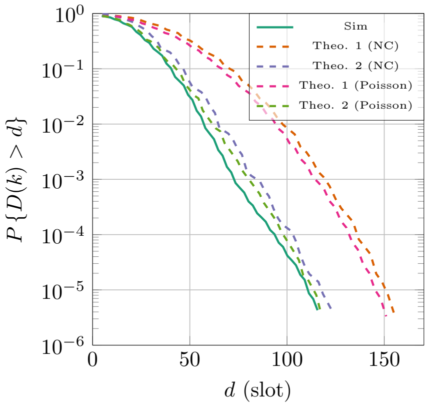
(b) ISRR E2E IPDV



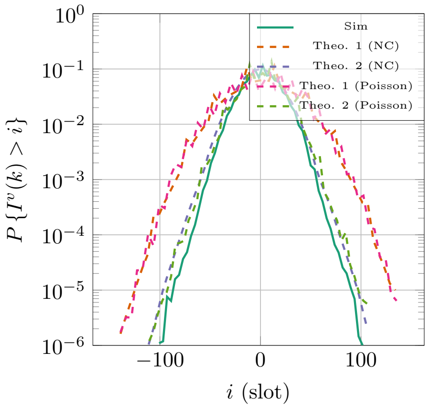
(c) GOOSE E2E



(d) GOOSE E2E IPDV



(e) VoIP E2E



(f) VoIP E2E IPDV

Figure 6.10: E2E results for scenario 6

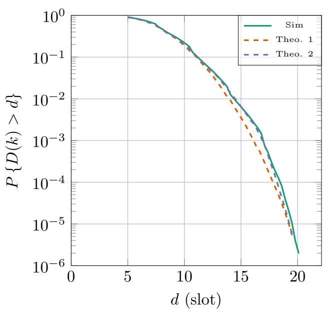
6.11 Scenario 7

This scenario is the same as scenario 6 (Section 6.11), except now there is background video and data through-traffic. Table 6.8 lists the parameters.

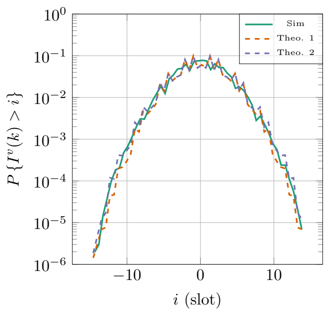
Table 6.8: Simulation parameters for scenario 7

Parameter	Value	Parameter	Value
N	5	ISRR	10 / 5 10 15 20 25
$\mathbf{R}$	$[1,1,1,1,1] \cdot$ Gbps	GOOSE	15 / 25 20 15 10 5
MTU	1 500 byte	VoIP	50 / 40 30 20 30 40
$\boldsymbol{\rho}$	[0.45, 0.25, 0.34, 0.26, 0.48]	Video	1 / 20 30 20 40 50
		Data	2 / 200 100 150 100 200

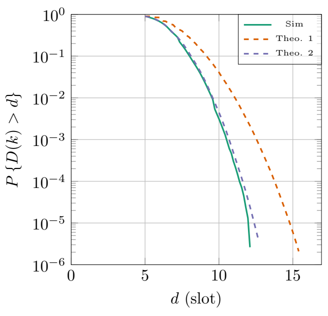
Adding TT video and data traffic does not impact the distributions much. This is because, as mentioned in the previous scenario, the high load of the CT causes a larger independence over the nodes. The IPDV distributions also still concur with the simulation results.



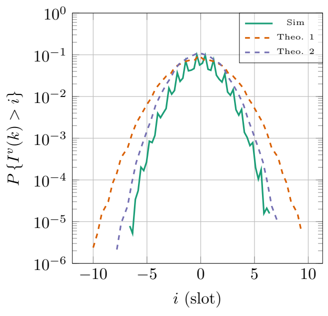
(a) ISRR E2E



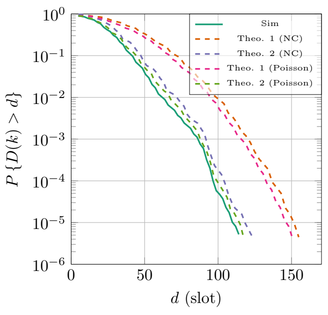
(b) ISRR E2E IPDV



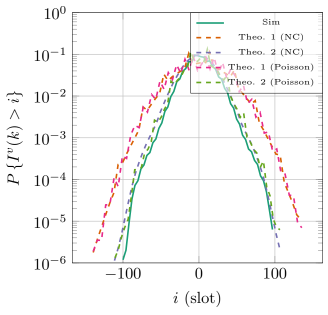
(c) GOOSE E2E



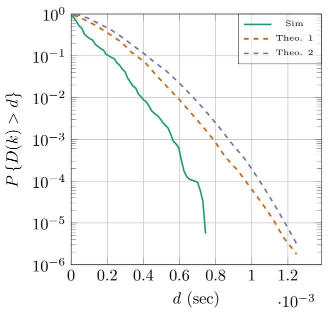
(d) GOOSE E2E IPDV



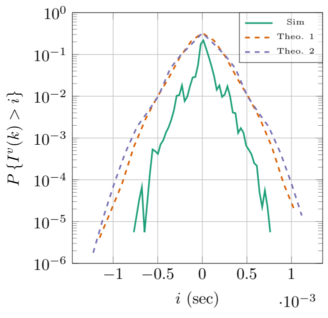
(e) VoIP E2E



(f) VoIP E2E IPDV



(g) Video E2E



(h) Video E2E IPDV

Figure 6.11: E2E results for scenario 7

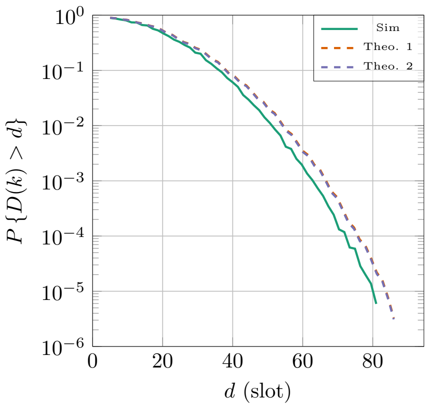
6.12 Scenario 8

In this scenario, we use the same parameters as in scenario 6 (Section 6.10), however, now the MTU is set to 8 000 byte. Thus, the Data traffic can now put packets of up to 8 000 byte on the line. Table 6.9 lists the parameters.

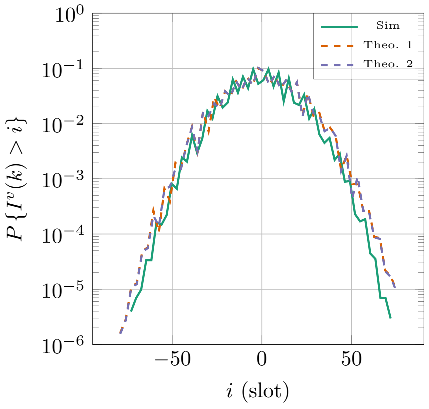
Table 6.9: Simulation parameters for scenario 8

Parameter	Value	Parameter	Value
N	5	ISRR	10 / 5 10 15 20 25
$\mathbf{R}$	[1,1,1,1,1] $\cdot$ Gbps	GOOSE	15 / 25 20 15 10 5
MTU	8 000 byte	VoIP	50 / 40 30 20 30 40
$\boldsymbol{\rho}$	[0.45, 0.24, 0.34, 0.25, 0.48]	Video	0 / 20 30 20 40 50
		Data	0 / 200 100 150 100 200

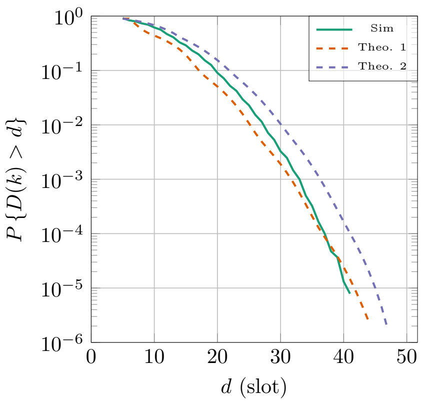
The "Theo. 2" distributions for this scenario provide a nice upper bound. Whereas in the previous scenario (Section 6.11) the predicted delays for the "Theo. 1" method was much larger, we have here that for the GOOSE and VoIP traffic it fails to provide an upper bound.



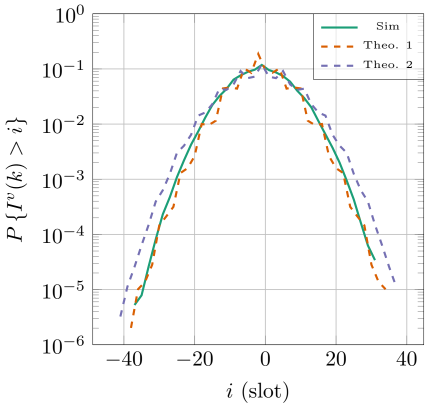
(a) ISRR E2E



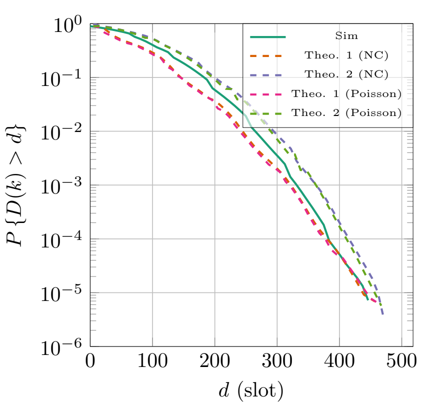
(b) ISRR E2E IPDV



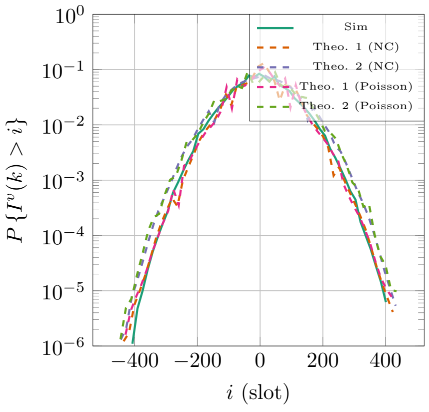
(c) GOOSE E2E



(d) GOOSE E2E IPDV



(e) VoIP E2E



(f) VoIP E2E IPDV

Figure 6.12: E2E results for scenario 8

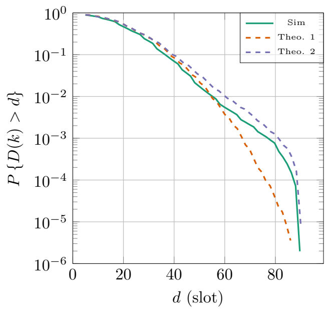
6.13 Scenario 9

In this scenario, finally, we use the same parameters as in scenario 8 (Section 6.12), but we also send data and video through-traffic, which can have a size of up to 8 000 byte. Table 6.10 lists the parameters.

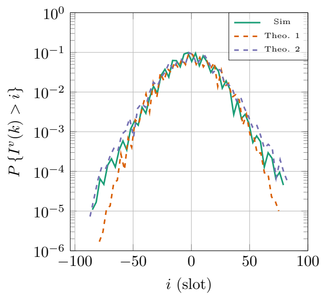
Table 6.10: Simulation parameters for scenario 9

Parameter	Value	Parameter	Value
N	5	ISRR	10 / 5 10 15 20 25
$\mathbf{R}$	$[1,1,1,1,1] \cdot$ Gbps	GOOSE	15 / 25 20 15 10 5
MTU	8 000 byte	VoIP	50 / 40 30 20 30 40
$\boldsymbol{\rho}$	[0.46, 0.25, 0.35, 0.26, 0.49]	Video	1 / 20 30 20 40 50
		Data	2 / 200 100 150 100 200

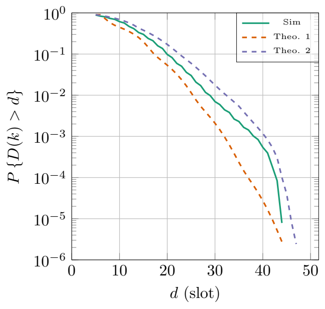
The "Theo. 2" provides an accurate bound, following the simulation curves closely, for both E2E delay and E2E IPDV. The "Theo. 1" method is now neither able to give good bounds on the curves. The large MTU and dependence of the TT traffic cause inaccuracies.



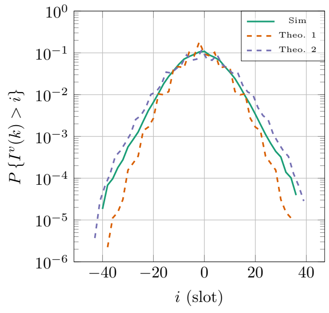
(a) ISRR E2E



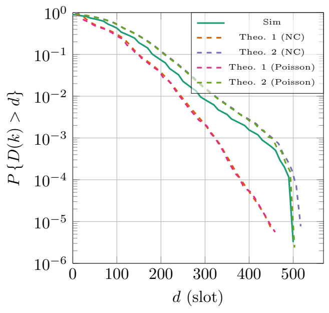
(b) ISRR E2E IPDV



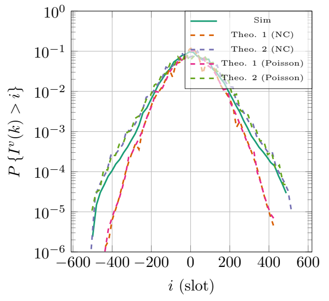
(c) GOOSE E2E



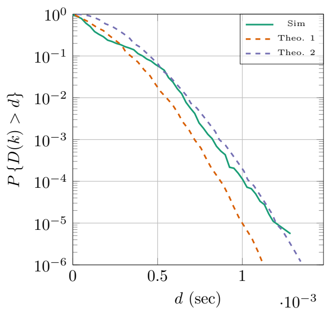
(d) GOOSE E2E IPDV



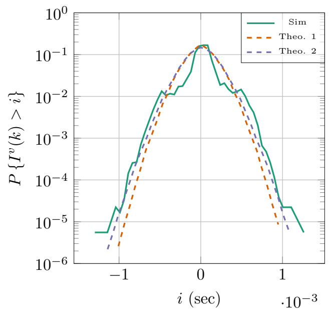
(e) VoIP E2E



(f) VoIP E2E IPDV



(g) Video E2E



(h) Video E2E IPDV

Figure 6.13: E2E results for scenario 9

6.14 Conclusion

In this section, we brought all the previous sections together, by combining all the different traffic types together in various scenarios. We first started discussion of how to incorporate the E2E delay, by looking at the delay encountered in each hop. For this we needed to take the through-traffic traffic into account, especially if the packets are large. We then discussed the simulation setup, followed by a revision of the traffic used in the simulation. We then proceeded to list plots the E2E delay and IPDV of various scenarios, which show both the simulation results and the predictions made by the analytical algorithms from previous sections. The simulations feature a mix of all the traffic, in all the nodes We found that the simple method from Section 6.1.2 is able to bound in some scenarios, whereas the more complex method from Section 6.1.3 that takes the TT into account, is able to bound the distribution in many more situations, when compared to the simulations we have run. For VoIP traffic we looked at a network calculus approach and one where we modeled the aggregate as an $M/D/1$ queue. Both approaches resulted in very similar behavior, with the Poisson one giving slightly tighter results, and thus both methods could be used to get a delay bound.

Chapter 7 Conclusion

In this chapter we analyzed the E2E delay and IPDV performance of different priorities for particular traffic in strict priority scheduler for a DiffServ network. The highest priorities of traffic were composed of aggregates of CBR traffic. The next priority comprised an aggregate of VoIP traffic, while for the lower priority we considered video. The lowest priority was not analyzed, but served as filler traffic.

For the analysis we made heavy use of the busy period of an aggregate of traffic. Paramount for this was the algorithm we developed to calculate this busy period for any aggregate of CBR flows. We then applied this principle to the other priorities, giving the delay, IPDV, queue size and busy period distributions for a single node.

We then extended these results to a sequence of nodes, where in each node additional cross-traffic was added. To calculate the E2E delay and IPDV, we tried two approaches. The simple approach assumed no video nor data through-traffic (but allows for video and/or data cross-traffic). In this case, we just applied the convolution of the queuing delay and vacation encountered in each node. The more complex approach could also deal with video and/or data through-traffic, and takes events in the previous node into account.

To evaluate our algorithms, we performed simulations for a 5 link network for different scenarios (with or without video/data through-traffic, a large MTU, a large system load). The analytical E2E delay and IPDV provided in general a nice bound to the simulation results, for all the different priorities.

Part II

Cross-layer resource allocation

Chapter 8
Introduction

In the late seventies and early eighties a surge of telecommunication occurred, as computers became valuable tools to increase corporate efficacy, and the advantage of having instantaneous access to data became apparent. This lead to a multitude of networks that were developed and sold by manufacturers as complete and integrated solutions. This provided the customers with network technologies such as Token Ring, ATM and Ethernet. All these technologies were developed independently of each other, with limited regard for cooperation between different network technologies. As time passed, demand increased for interoperability between all these different systems in order to reuse the available infrastructure and reduce costs. Due to the lack of standardization, connecting networks often proved difficult and sometimes even outright impossible.

The ISO recognized this problem early on, and researched existing architectures, such as IBM's System Network Architecture (SNA) and ARPANET of the American Department of Defense (DoD), to solve this inter-connectivity problem. The ISO proposed the OSI model, that consisted of a layered abstraction of communication. In this model, each of the seven layers represent a different class of problems that is solved.

Each layer builds on the layer beneath it, through interfaces, while at the same time providing an interface that implements the service to the layer directly above it. Each layer in this model communicates only with its peer on the same layer, and has only knowledge about the workings of its own layer. In Figure 8.1 we can see the different layers of this model. We will briefly discuss the first four lower layers.

At the lowest level, we can find the physical layer. It determines the transformation from bits to a signal and vice versa, such that the receiver can reconstruct the original signal. Common physical layer technologies are Ethernet, IEEE 802.11, Bluetooth, and more related to this chapter, DSL.

The second layer, the data link layer, deals with node-to-node communication on a network segment. It consists of two sublayers: the logical link control (LLC) and media access control (MAC). The LLC is responsible for frame

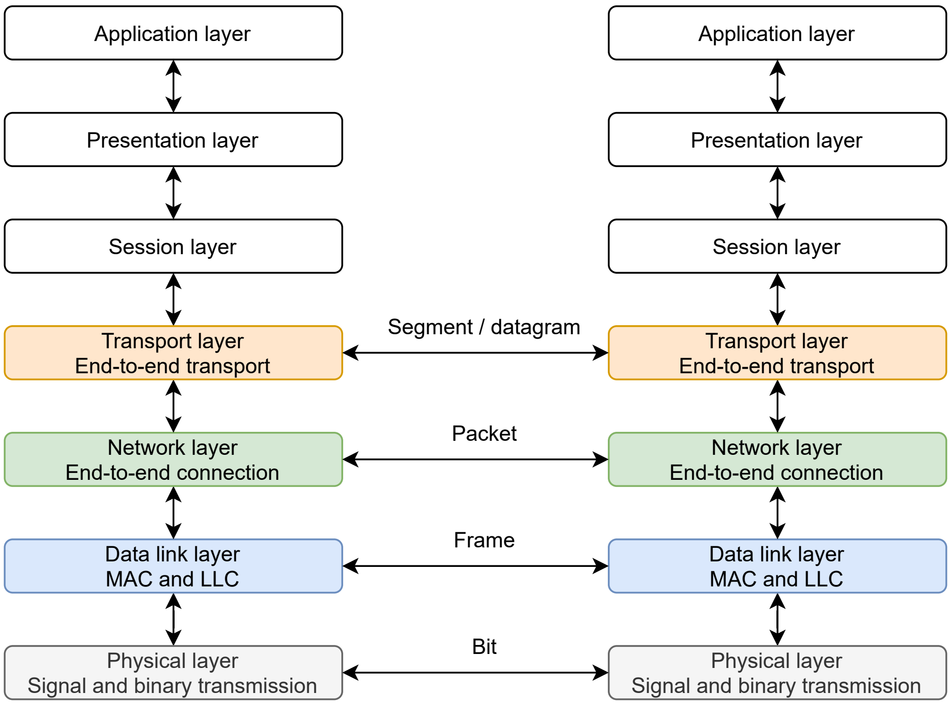


Figure 8.1: The seven layers of the OSI model

synchronization, error detection and handling network layer protocols. The MAC sublayer determines the device that is allowed to access the shared medium, but can also provide an addressing function (e.g. the MAC address). Common data layer protocol are (also) Ethernet and the point-to-point protocol (PPP).

The third layer, called the network layer, provides functionality for communication of packets between two nodes that can be separated by multiple devices. As such, it provides a mechanism to address individual nodes in a network, but also determines how to route packets from the source to the destination through intermediate nodes. The most common network layer protocol is internet protocol version 4 (IPv4). It is in use since the early 80s. However, not too long after its introduction, it was clear that IP addresses of 32 bit, resulting in an address space of about 4.3 billion IP addresses, would have to be used conservatively, lest there would be no free IP addresses. To deal with this problem of address exhaustion, the IETF finished in 1998 the development of its successor IPv6 which, among other things, introduced addresses of 128 bits. In 2017 IPv6 got ratified as an Internet Standard. The adoption for IPv6 is, however, going very slow (Réseaux IP Européens Network Coordination Centre (RIPE NCC), which coordinates allocation of IP addresses for Europe and the Middle-East, announced on 25th of November 2019 that it had run out of IPv4 addresses), and IPv4 and IPv6 are expected to be used alongside each other for the foreseeable future.

The fourth layer, the transport layer, builds upon the network layer to provide transport of data between two nodes. The difference with the third layer is that this layer provides a stream of data, rather than a stream of packets and that it is an E2E protocol. The two most used protocols are TCP and UDP, where the former provides a connection-oriented, reliable, ordered and error-free delivery of a stream, whereas the latter provides a much simpler connectionless stream of data.

In the OSI model, each layer is a black box with a standardized interface, guaranteeing high modularity leading to a simpler design of network protocols, as reuse of functionality is encouraged through these layers.

This abstraction, however, comes at an opportunity cost. To keep the interfaces as general as possible, only a minimal amount of information can be shared through the interface, discarding information that can possibly be used to increase performance. Cross-layering algorithms undo some of this information loss by carefully selecting useful information from different layers, i.e. they encourage transfer of information between layers that are not adjacent to each other, in order to increase performance of a layer.

For example, in recent DSL technologies, the data rates at which users can transmit are selected in advance from a fixed set of possible data rates that are possible on the physical layer. The data rate is chosen in function of the long-term usage. To ensure that all users get sufficient service, the configuration must provision for the usage for the duration of the connection. This implies that the DSL configuration must be dimensioned close to the peak rate to ensure the quality of experience (QoE) is optimal. This peak rate, however, might only occur a very limited amount of time, but consumes, nonetheless, service rate as if the user had been transmitting at peak rate. Thus, at short timescales, we usually require considerably less service to satisfy the QoE. Therefore, the physical layer can benefit from the information contained within the upper layers to steer selecting the configuration towards a more optimal service rate, leading to more efficient usage and better service for all users.

Another example where cross-layering is beneficial are wireless systems. When transmitting a signal over the shared medium, signals might not always arrive correctly at the receiver side, due to small-scale effects, such as multi-path fading, and large-scale effects such as path loss and shadowing. Therefore, a sender might try to estimate the current channel conditions, and based on that information decide whether to use more power and transmit a packet, or hold of transmission until the conditions are more favorable. This reduces power consumption, increases the system throughput and leads to a more efficient use of the shared medium, as different devices often encounter very different channel characteristics.

A final example, among the many, where cross-layering is useful is satellite communications. As bandwidth is expensive, it is in the interest of the user to use as much as possible of the service rate that is offered. Like in the DSL example, the traffic intensity changes over time, hence there is a need to dynamically reserve bandwidth to reduce the costs. This reservation, however, takes a lot of time due to the large distance the signal has to travel, requiring the need for a predictive and adaptive approach, that takes the system and arrivals into account and communicate those requirements to the physical layer.

However, cross-layering is also not without drawbacks. First, cross-layering introduces complexity and can slow down innovation. For example, if a protocol is developed or updated, the layers it interacts with might need additional updates to ensure proper functioning.

Second, with the number of possible interactions between the different layers increasing, grows also the number of possibilities in which the layers interact in unintended ways. This might lead to performance problems, which is why we introduced cross-layering in the first place! An example of such an unintended consequence can be found in [97]. There, a rate-adaptive MAC protocol is combined with minimum-hop routing. Minimum-hop routing will try to route using the lowest number of hops, by routing to the hops the furthest away. A device can order devices by distance by using the data rate as a surrogate: for a regular MAC protocol, we can assume that a low data rate implies a low received signal strength, hinting at a large distance. Combining this with a rate-adaptive MAC protocol which sets the transmission data rate in function of the channel quality, we can see that the data rate is not a good measure anymore for distance, resulting in a worse performance.

A large body of literature exists on cross-layer optimization, see e.g. [143, 117, 45, 127, 200, 68, 100] and references therein. Typically, the cross-layer problem is formulated as an optimization problem of the form

$$\text{maximize } \sum_n U_n(\mathbf{x_n}).$$

In here, $U_n(\cdot)$ is the utility of a device receiving a quantity x_n (e.g. a data rate), where the system itself is subject to constraints, such as power usage, routing restrictions, rate restrictions, stability etc. The function $U_n(\cdot)$ can be seen as the interface between the different layers.

The first cross-layer scheduler was the max-weight (MW) scheduler, defined in the seminal work [180], and is defined by $U_n(R) = q_n[t]\cdot R$, where $q_n[t]$ is the queue size of user n at time t, and R is the rate. Conceptually, the scheduler looks at all tuples of rates that are possible in the system, and then selects the tuple of rates for which the sum of the queue-rate product over the users is the highest. For example, in a two user system with a rate region (i.e. all possible rate tuples) $\{(3,0),(2,2),(0,3)\}$ and queue sizes $\mathbf{q}[t] = [1,2]$ the MW scheduler will select the rate tuple $(2,2)$, because $2\cdot1+2\cdot2 > 3\cdot1+0\cdot2$ and $2\cdot1+2\cdot2 > 0\cdot1+3\cdot2$. As the queue sizes are more or less equal, the scheduler tries to maximize the system data rate. However, for queue sizes $[1,5]$ the selected rate will be $(0,3)$ because $0\cdot1+3\cdot5$ trumps all other possibilities. In this case, the largest queue will receive the most data rate. Usually much more efficient algorithms are available, depending on the shape of the collection of rate tuples and technology, to determine the solution to the maximization problem. Other schedulers have a different utility function, and focus on other properties and metrics.

In this chapter we develop algorithms for cross-layer optimization for the three different contexts of the examples given above. In Chapter 9 we develop a resource allocation algorithm that combines the physical and data link layer to improve the performance of DSL networks. In Chapter 10 we slightly modify the algorithm developed in Chapter 9 and apply it to a LTE and 5G context. In Chapter 11 we develop two new algorithms for use in a satellite context where many users' flows are aggregated to reduce costs and make more efficient use of the bandwidth. The available service rate for this link is updated periodically according to the requirements. However, due to the large latency these service rates updates are applied with a delay, and hence a predictive approach is required to achieve good performance. Finally, in Chapter 12 an algorithm is developed to provide lower and upper bound data rate guarantees to cross-layer schedulers from Chapters 9 and 10.

Chapter 9

Cross-layer optimization in DSL networks

9.1 Introduction

9.1.1 DSL

In the 1990s, cable television and satellite industries had been offering data rates of 10 Mbps and more. The telephone industry, however, was stuck at offering 56 kbps (or 144 kbps for Integrated Services Digital Network (ISDN)). As the triple play combination of internet access, television en voice communication became more important, the telephone companies realized they should offer a more competitive product. As they already had the infrastructure going into each house, i.e. twisted-copper pair cables, they started to develop new technologies over the next decades that leveraged the available copper network, to increase the data rates and provide a better service. Over the course of decades, new generations of broadband emerged.

The second generation broadband introduced asymmetric digital subscriber line (ADSL) for the consumer market, which offered a larger downstream (up to 24 Mbit) than upstream rate (at most 1 Mbit), hence the *asymmetric* in the name. New equipment was needed at both sides of the network, the copper lines, however, remained untouched. Unlike in the first generation, data signals employed the unused frequencies of the spectrum above the voice signal, resulting in a higher bandwidth and also allowed for simultaneous voice calls and browsing the internet. However, the high frequencies attenuated strongly in the twisted-pair cable, reducing the achievable data rate over longer distances, giving users further away from the central office (CO) lower rates. For example, in Figure 9.1 we plot in blue the curve for ADSL2+ of the maximum achievable data rate in function of the distance. For small distances a user might attain 24 Mbps, however as the distance becomes larger it decreases quickly. Also plotted are curves for third generation broadband technologies.

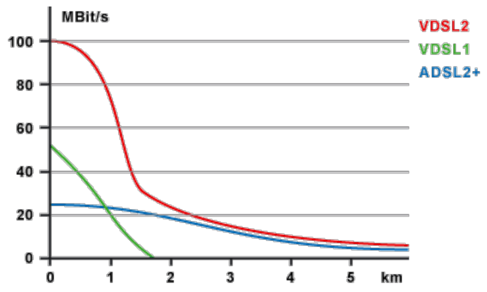


Figure 9.1: Rate versus distance

The third generation of broadband was able to increase the data rate by several factors by bringing the loop closer to the house. Ideally the local loop is less than 1 km. Now crosstalk, interference between the copper wires of different users, became the major contributor to performance degradation. The effects of crosstalk were minimized using vectoring, a sort of noise-cancelling, but then for electromagnetic signals.

The 4GBB is a hybrid fiber-DSL solution that brings the local loop even closer to the end user, up to 20 m. Wireline techniques, such as multiple input/multiple output (MIMO) signaling, discrete multitone (DMT) modulation and dynamic spectrum management (DSM) are used to optimally exploit the frequency dimensions. MIMO signaling exploits the fact that users are often provided two copper pairs, instead of one. DMT modulation is a form of orthogonal frequency-division multiplexing (OFDM) where transmission on each subcarrier is adapted depending on the channel conditions. Finally, DSM tries to solve the crosstalk problem, by allocating parts of the spectrum to users that have low interference.

Currently, fifth generation broadband access (5GBB), also referred to as XG.Fast, is part of active research. It will bridge the last gap between 4GBB and fiber to the home (FTTH). It improves on 4GBB, which has been standardized in G.fast [88, 87], by bringing the fiber closer to the end user and expanding the used spectrum. In this scenario, this fiber to the frontage (FTTF) reduces the copper wire length to less than 70 m, enabling data rates of up to 10 Gbps over the twisted-copper pairs.

In Table 9.1 we show some technologies, the International Telecommunication Union (ITU) recommendation, when it was ratified and the speed capabilities. Figure 9.2 shows the historical (up to 2009) and expected trend of the number of new installations and upgrades through time, for each DSL generation.

Table 9.1: DSL technologies

Family	ITU	Name	Ratified	Maximum rate
ADSL	G.992.1	G.dmt	1999	7 Mbps / 800 kbps
ADSL2	G.992.3	G.dmt.bis	2002	8 Mbps / 1 Mbps
ADSL2plus	G.992.5	ADSL2plus	2003	24 Mbps / 1 Mbps
ADSL2-RE	G.992.3	Reach Extended	2003	8 Mbps / 1 Mbps
SHDSL	G.991.2	G.SHDSL	2003	5.6 Mbps
VDSL	G.993.1	Very-high-data-rate DSL	2004	55 Mbps / 15 Mbps
VDSL2	G.993.2	Very-high-data-rate DSL 2	2005	100 Mbps
G.fast	G.9701	G.fast	2014	1 Gbps
XG.fast	-	XG.Fast	-	10 Gbps

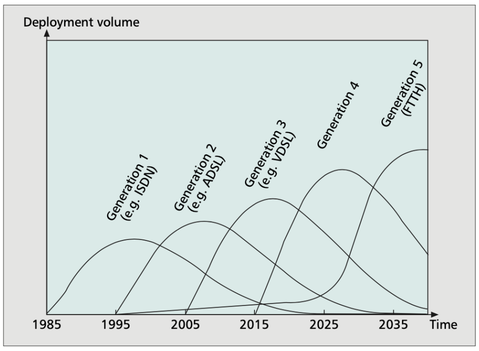


Figure 9.2: Trend of deployment volumes of DSL generations

Source: [140]

In Figure 9.3 we can see a general network architecture of DSL. At the left, we have the customer premises where the service is consumed. The distribution point unit (DPU) contains the Digital subscriber Line Access Multiplexer (DSLAM), which aggregates the different users, and the optical network terminal (ONT) which connects the DPU to the CO. The main CO typically aggregates passive infrastructure of multiple COs. The core CO connects the aggregation network to other networks, such as the Internet.

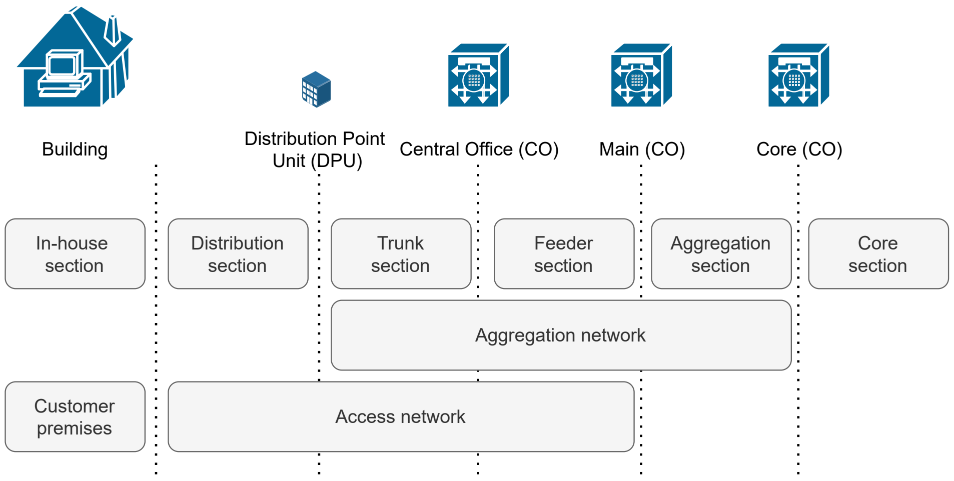


Figure 9.3: DSL network architecture

9.1.2 Cross-layer optimization

The work in this section applies to the connection between the customer premises and the DPU.

Maintaining a low delay in a communication network is critical to a wide variety of applications such as video conferencing, VoIP, gaming, and live-streaming. If many delay violations occur, QoE suffers considerably for these applications, and the allocated resources are wasted. The QoE is usually expressed through 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. As mentioned in the DSL introduction, multiple twisted pair lines connect the 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 9.4 shows a setup in which the signal from users 2 to N leaks into user 1's signal.

This crosstalk gives rise to a convex rate region, the set of all rate vectors that can be provided by the physical layer. Figure 9.5 shows an example rate region $\mathcal{R}$ for two users. Due to crosstalk there is no allocation that maximizes the service rate for both users at the same time: increasing the service rate ρ of one user invariably reduces the service rate of the other user.

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

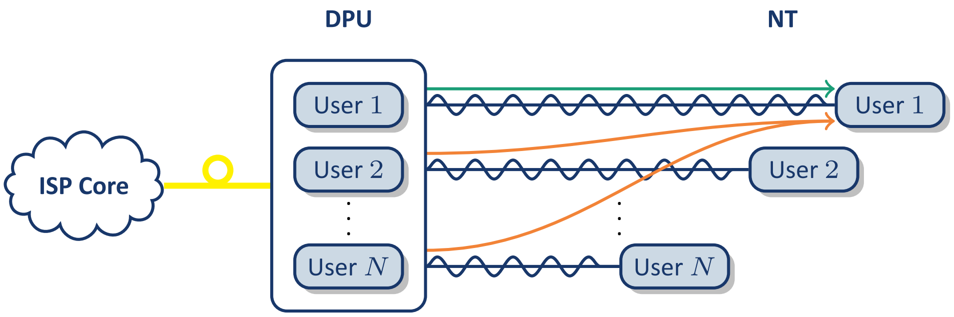


Figure 9.4: Example of crosstalk in a DSL system

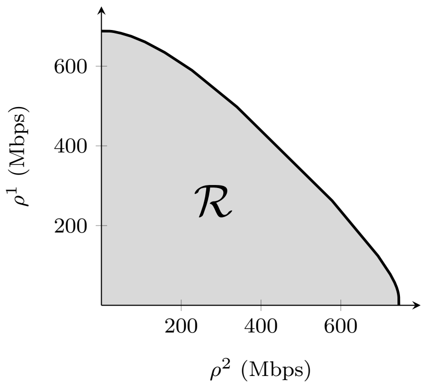


Figure 9.5: A rate region for a two user system

layer, meanwhile, there are multiple Pareto-optimal data rate points from which to choose (see for example Figure 9.5). As is dictated by the 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, especially since worst-case behavior must be assumed, even though these peak arrivals might occur only in e.g. 1% of the time. 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.

Traditionally, DSL techniques configure the physical layer statically using profiles which are not always the best choice for the current usage. Breaking here through the strict layers of the OSI model can prove very beneficial. In the example given above, the upper layers might instruct the physical layer to share service based on the expected arrivals in the next second. The physical layer then decides on how to allocate resources, based on the upper layer's preference information, the number of predicted arrivals, in this case.

Note that we here thus break through the abstraction of the physical layer, which is just a service that transmit bits, and transform it into a service that transmits bits, *depending* on information passed on from the upper layers, in this case for example.

Dynamically adjusting the service rates offered to users can resolve these problems. 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:

$$\mathbf{R}^* = \arg\max_{\mathbf{R}\in\mathcal{R}} \sum_n u^n(R^n) \tag{9.1}$$

where $\mathbf{R} = [R^1, \ldots, R^n]^T$. 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 [198] or power usage [197], or joint optimizing of several metrics [195]. These schedulers assume that the solution to the (9.1) 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.

Therefore, we develop a scheduler targeted towards DSL G.fast 4GBB and 5GBB communication networks that aids in assigning resources. The minimal delay violation 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, and offers significant improvements with respect to a static allocation.

The remainder of this section is structured as follows. We discuss related work in Section 9.2. In Section 9.3 we describe the system model. In Section 9.4 we present a formal description of the MDV scheduler, and an analysis of the scheduler together with a discussion of the stability. In Section 9.5 we briefly discuss the physical layer. In Section 9.6 we evaluate the MDV scheduler and compare the performance with other cross-layer schedulers using simulations.

9.2 Related work

Many of the schedulers listed here are used in wireless networks, but due to the general nature of the NUM problem (9.1) 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, (9.1) is often simplified to

$$\mathbf{R}^*[t+1] = \arg\max_{\mathbf{R}\in\mathcal{R}} \sum_n \omega^n[t] R^n \tag{9.2}$$

where t is the time slot and ω^n is called the weight of user n. The seminal work of the MW scheduler [180] 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 [16] has $\omega^n[t] = \Gamma^n[t]$, where $\Gamma^n[t]$ is called the 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 [170], $\omega^n[t] = \frac{|u'(\overline{w}^n)|}{\overline{\lambda}^n}$, 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 [28] scheduler differentiates between real-time and best-effort applications. The real-time application weights take 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 [157] 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 [96] a joint power allocation and transmit scheduling method is introduced for 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 [164] 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 EDF and modified largest weighted delay first (M-LWDF). Some approaches incorporate a neural network (NN). For example, in [161] 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 [20] another allocation algorithm is presented, but the tuning parameters require a priori knowledge of the applications, such as the required throughput. In [197] 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 [198] 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 [195] 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 [130] and Opt.-Fair [61] implement a mechanism similar to SP in every time slot in order to achieve a low 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 [110] different scheduling strategies are discussed together with a survey of the schedulers. A taxonomy of cross-layer schedulers can be found in [137].

Some of the schedulers listed above are also used in the simulations in Section 9.6 and are listed in Table 9.4 on Page 131 together with the expression used by each scheduler to calculate the user weights ω in the weighted sum rate maximization problem (9.1).

9.3 System model

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

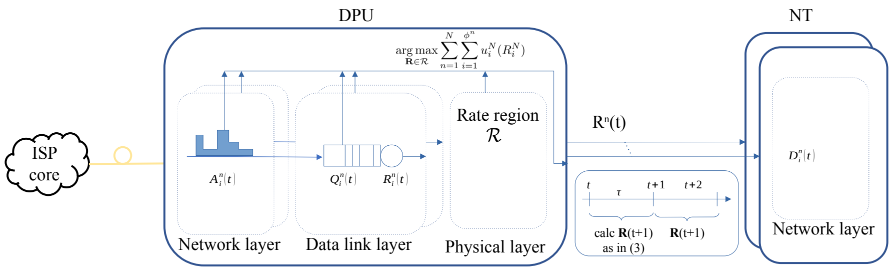


Figure 9.6: The system model

In Figure 9.6 we show an overview of the system model. The ISP core network is connected to the DPU through an optical fiber cable. The DPU is connected to N network terminations (NTs). In the DPU the different layers contribute here to solving the NUM problem. The physical and data link layers at the NT are omitted.

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

$$\mathcal{C} = \operatorname{conv}\left(\bigcup_{\mathbf{r}\in\mathcal{R}} \left((\{\mathbf{r}\} - \mathbb{R}_+^N) \cap \mathbb{R}_+^N\right)\right), \tag{9.3}$$

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 (9.1) 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 $\mathbf{R}[t+1]$, 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 $P\{D[t] > \hat{T}_i^n\} \le \varepsilon_i^n$, where $D[t]$ 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 $P\{D[t] > \hat{T}_i^n\} > \varepsilon_i^n$ 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 $A_{i,p}^n(s,t)$ and $E_{i,p}^n(s,t)$. The short-term PLR at time t is notated as $p[t] = \frac{d_{i,p}^n(t-10,t)}{A_{i,p}^n(t-10,t)}$, where $d_{i,p}^n(s,t)$ is the number of dropped packets in the interval $[s,t]$. We track this only over the last $10s$, which corresponded to the average scene length in our video traces.

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 $\mathscr{S}_i^n[t]$*. At the start of slot

*We will usually omit $\mathscr{S}_i^n[t]$

t the cross-layer scheduler selects the rate assignment $\mathbf{R}[t+1] \in \mathcal{R}$ that maximizes the system's performance, i.e.:

$$\mathbf{R}[t+1] = \arg\max_{\mathbf{R}\in\mathcal{R}} \sum_{n=1}^{N} \sum_{i=1}^{\phi^n} u_i^n(R_i^n, \mathscr{S}_i^n[t]). \tag{9.4}$$

A large family of scheduling algorithms is linear in $\mathbf{R}$ (e.g. [73, 132, 93, 170, 17]), i.e.

$$u(R, \mathscr{S}[t]) = R \cdot \omega(\mathscr{S}[t]). \tag{9.5}$$

For the MDV scheduler we present here, we have

$$u(R, \mathscr{S}[t]) = -\frac{\omega(\mathscr{S}[t])}{R+\zeta} \tag{9.6}$$

where ζ is a small constant to avoid division by 0. We refer to (9.5) and (9.6) as MW-style and MD-style schedulers respectively. The reason for using an MD-style scheduler is explained in the introduction of the next section.

9.4 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 that 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 $\mathbf{q}[t] = [0, 10]$, and at $t+0.01$ packets arrive for user 1, then if the weight depends on the queue (e.g. $u_n(\rho) = q_n\rho_n$, or [198, 138]) or HOL (e.g. [199, 202]), then user 1 might get assigned a zero service rate for the coming 50 ms slot (and thus have a delay larger than 50 ms), while user 2 receives all data rate.

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 (9.5) 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 and the HOL, 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 (9.6), which is more suitable for fair sharing [133].

$$\omega_i^n(\mathscr{S}_i^n[t]) = \underbrace{\tilde{\lambda}_i^n[t{+}1]}_{\text{(a)}} \cdot \underbrace{c_i^n}_{\text{(c)}} \left(\underbrace{\frac{q_i^n[t]}{(R_i^n[t]{+}\zeta)\cdot \hat{T}_i^n} + \ln_2\left(1{+}\frac{p_i^n[t]}{\varepsilon_i^n}\right) + \frac{\Gamma_i^n[t]}{\hat{T}_i^n}}_{(b)} \right) \tag{9.7}$$

The weight for the MDV scheduler is shown in (9.7), 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 9.4.1 we will first discuss the components that comprise the scheduler. Then in Section 9.4.3 we highlight a difference with MW-style schedulers. In Section 9.4.4 we discuss the stability of the MDV scheduler. In Section 9.4.6 we add some important notes on the discretization of the rate region.

9.4.1 The components

In this subsection we will have a detailed look at the components that make up the MDV scheduler, as described in (9.7).

9.4.1.1 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 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 slot $t{+}1$.[†] One approach to obtaining $\tilde{A}_i^n[t{+}1]$ is to use for each flow a suitable traffic model, describing its typical behavior (e.g. [37, 190, 80]). However, such an approach requires a model with all its correct parameters to be available for all the flows present in the network. Instead, we use an exponentially moving average (EMA) with weight 0.2, i.e. $\tilde{A}_i^n[t{+}1] = 0.8\cdot\tilde{A}_i^n[t]{+}0.2\cdot A_i^n(t{-}1)$. One could also use more

[†]For stability reasons (see Section 9.4.4), we assume $\tilde{A}_i^n[t] > 0$.

sophisticated prediction mechanisms, like the normalized least mean square (NLMS) predictor [76]. The added complexity, however, does not result in a significant improvement, so to keep complexity low, we use the EMA.

9.4.1.2 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

$$b_i^n = \underbrace{\frac{q_i^n[t]}{(R_i^n[t]+\zeta)\cdot\hat{T}_i^n}}_{\text{(b1)}} + \underbrace{\ln_2\left(1+\frac{p_i^n[t]}{\varepsilon_i^n}\right)}_{\text{(b2)}} + \underbrace{\frac{\Gamma_i^n[t]}{\hat{T}_i^n}}_{\text{(b3)}} \tag{9.8}$$

It expresses the closeness of a flow to violating its QoS delay requirement $P\{D[t] < \hat{T}_i^n\} \leq \varepsilon_i^n$, which is then used as the argument of c_i^n in (9.7). 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 avoid or reduce the delay violations.

We estimate the b_i^n metric based on the current queue size (b1), the past delay violations (b2) and the HOL (b3), which approximate the predicted delay of the most recently arrived packet, the delay percentile and delay of the current packet respectively. We now look at (b1), (b2) and (b3) in more detail.

(b1): The factor $\frac{q_i^n[t]}{R_i^n[t]+\zeta}$ in (9.8) 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 [139] 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. To track $p[t] = \frac{d_{i,p}^n(t-10,t)}{A_{i,p}^n(t-10,t)}$, we store the arrivals and drops in a circular buffer, each entry covering one second. It is also possible, and less complex, to just track the number of drops and arrivals over the lifetime of a flow. However, it might be beneficial for the performance to forget about the drops that have occurred long ago.

(b3): The term (b3) represents an estimate of the proximity of the HOL to its delay deadline. As the packets for a flow arrive in a first in first out (FIFO) queue, the HOL will be the oldest packet in the queue — if its delay is less than $\hat{T}_i^n$, then so are the other packets in the queue.

9.4.1.3 Function (c)

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

- $c_{stream}(x) = \beta(x, 0.5, 1.0, 0.2, 10)$
- $c_{BE}(x) = \beta(x, 0.7, 1.0, 0.7, 10)$

with

$$\beta(x,\mu,\gamma,\sigma,\rho) = \begin{cases} \beta_2(x,\mu,\gamma,\sigma) & \text{if } x \le 1 \\ \beta_2(1,\mu,\gamma,\sigma)+\log(\frac{x-1+1/\rho}{1/\rho}) & \text{if } x > 1 \end{cases} \tag{9.9}$$

where β_2 is the sigmoid function

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

The functions c_{stream} and c_{BE} behave like a regular sigmoid when $b_i^n \le 1$. When $b_i^n > 1$, however, c_{stream} and c_{BE} will switch to logarithmic mode, providing fairness among flows of the same class. The initial slope and value are 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 9.7, 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.

9.4.2 Saturating the channel

Some applications might try to send as much data as possible, by keeping the queue always backlogged. We call them here the SAT applications, as they try to saturate the channel. We can not use (9.7) as the queue is not a reliable metric in this case. We assume that these packets have no QoS requirements.

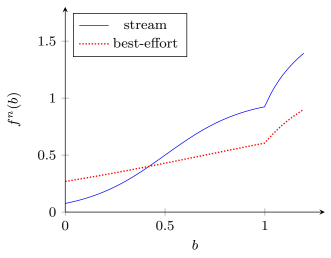


Figure 9.7: $c_{stream}(b)$ and $c_{BE}(b)$

A possible weight for these applications is

$$\omega_i^n(\mathscr{S}_i^n[t]) = \overline{\lambda}_{\text{QoS}} \cdot \Omega \cdot \frac{\check{R}^{\text{SAT}}[t]}{\overline{R}_i^n[t]}, \tag{9.10}$$

where $\overline{\lambda}_{\text{QoS}}$ is the average arrival rate of all QoS flows, $\Omega = 0.2$ a constant, $\overline{R}_i^n$ is the EMA of the service rate and $\check{R}^{\text{SAT}[t]} = \arg\min_{j \in \text{SAT}}\{\overline{R}_j[t]\}$ is the minimum service rate of all SAT flows. The factor $1/\overline{R}_i^n[t]$ is often used in schedulers for best-effort (BE) traffic, and provides fairness among the SAT applications. This value is scaled by $\overline{\lambda_{\text{QoS}}} \cdot \Omega \cdot \check{R}^{\text{SAT}}$ such that it has a weight that is in the same order of magnitude as the QoS flow's weights. The factor $\Omega \cdot \frac{\check{R}^{\text{SAT}}}{\overline{R}_i^n[t]}$ results in a value $]0, \Omega]$ and can be seen as the $f_c(\cdot)$ counterpart of the real time flows in (9.7). By increasing Ω, we can trade PLR performance of QoS flows for an increase in average throughput of the SAT traffic. The value $\Omega = 0.2$ was chosen using simulations, and had the least impact on the delay performance of the non-saturating flows while also maximizing the system throughput.

9.4.3 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. [108, 10, 40]), 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. The work presented in [188] discusses the possibility of having each flow having different properties in a DSL setting.

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 $\mathbf{R}$ 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 9.8 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, i.e. any point on the simplex is Pareto optimal.

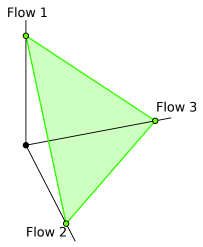


Figure 9.8: Intra-user rate region for a user with three flows

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 $\mathbf{r}^n = [r_1^n, \ldots, r_{\phi^n}^n]$ then the NUM problem can be written as

$$\begin{aligned} \underset{[r_1^n,\ldots,r_{\phi^n}^n]^T \succcurlyeq \mathbf{0}}{\arg\max} & \sum_{i=1}^{\phi^n} \omega_i^n r_i^n, \\ \text{subject to} & \sum_{i=1}^{\phi^n} r_i^n \leq R^n. \end{aligned} \tag{9.11}$$

Here, (9.11) 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:

$$\begin{aligned} \underset{[r_1^n,\ldots,r_{\phi^n}^n]^T \succcurlyeq \mathbf{0}}{\arg\max} & \sum_{i=1}^{\phi^n} -\frac{\omega_i^n}{r_i^n}, \\ \text{subject to} & \sum_{i=1}^{\phi^n} r_i^n \leq R^n. \end{aligned} \tag{9.12}$$

For this problem the closed form solution is

$$\mathbf{r^n} = \left(\sum_{i=1}^{\phi^n} \sqrt{\omega_i^n} \right)^{-1} \cdot \begin{bmatrix} \sqrt{\omega_1^n} \\ \vdots \\ \sqrt{\omega_{\phi^n}^n} \end{bmatrix} \cdot R^n. \tag{9.13}$$

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 to take advantage of the different flow properties.

9.4.4 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 λ the expected lengths of all queues in the system remain bounded. The stability region is the set of all arrival rate vectors λ for which a 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 [180] are proven to be throughput optimal in some scenarios [158].

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 [32, 133]. 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

$$\mathbf{R}^* = \underset{\mathbf{R}\in\mathcal{R}^\alpha}{\arg\max} \sum_i \left(A_i Q_i[t] + B_i\right)^{\beta-1} \frac{\left(R_i+\zeta\right)^{1-\alpha}}{1-\alpha} \tag{9.14}$$

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 [133]. 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 (9.14), $\mathcal{R}^\alpha \subseteq \mathcal{R}$ 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 [33] 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 9.9 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 9.9a 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 [119]. In Section 9.4.6 we have some remarks about this sampling process.

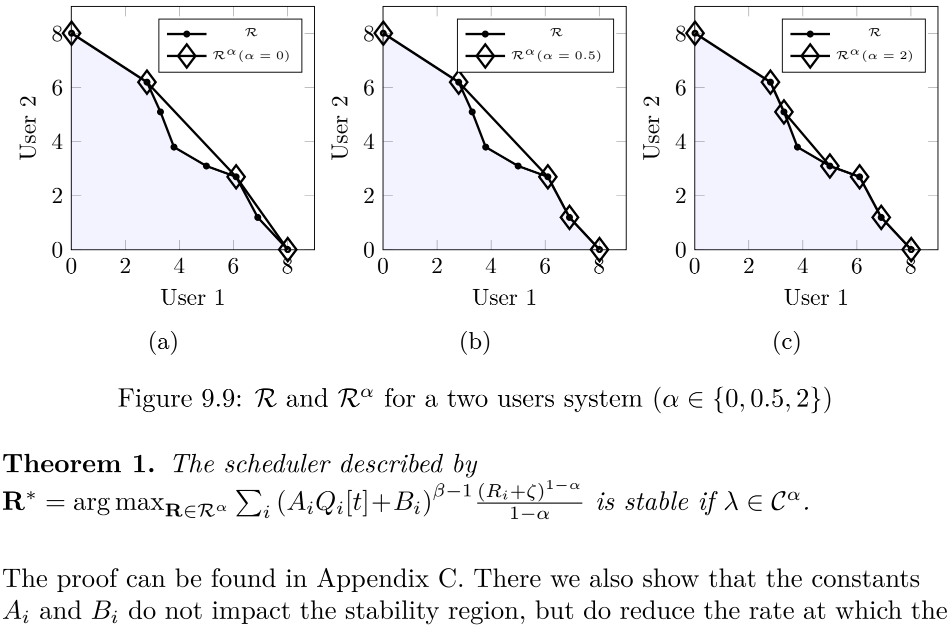


Figure 9.9: $\mathcal{R}$ and $\mathcal{R}^\alpha$ for a two users system ($\alpha \in \{0, 0.5, 2\}$)

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

The proof can be found in Appendix C. 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*

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

is stable for $\lambda \in \mathcal{C}^\alpha$ 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 D, 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 $\lambda \in \mathcal{C}^2$.*

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 (9.7) (as depicted by the dashed lines in Figure 9.10):

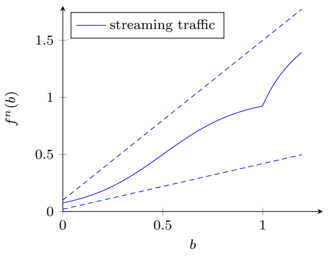


Figure 9.10: Multiplier for the streaming traffic class and bounds

$$\begin{aligned}
&g_l\tilde{\lambda}_i^n[t+1]\frac{q_i^n[t]}{(R_i^n[t]+\zeta)\cdot\hat{T}_i^n} \\
\leq &\tilde{\lambda}_i^n[t+1]c_i^n\left(\frac{q_i^n[t]}{(R_i^n[t]+\zeta)\cdot\hat{T}_i^n}+\ln_2\left(1+\frac{p_i^n[t]}{\varepsilon_i^n}\right)+\frac{\Gamma_i^n[t]}{\hat{T}_i^n}\right) \qquad (9.15)\\
\leq &g_u\tilde{\lambda}_i^n[t+1]\frac{q_i^n[t]}{(R_i^n[t]+\zeta)\cdot\hat{T}_i^n}+g_u\cdot\tilde{\lambda}_i^n[t+1]\cdot\left(\ln_2\left(1+\frac{1}{\varepsilon_i^n}\right)+\frac{\Gamma_i^n[t]}{\hat{T}_i^n}\right).
\end{aligned}$$

Equation (9.15) is the same as the weight ω_i^n from the MDV scheduler, first described in Equation (9.7). As mentioned in Section 9.4, 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 (9.8) that using only the number of delay violations will result in an unstable MDV scheduler, as the delay violation 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-based metric, such as the queue length or the HOL, to keep the scheduler stable.

Other schedulers (e.g. [164]) 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 taking the limit of $\rho \to 0$ in (9.9). However, in this case 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 λ is within $\mathcal{C}^\alpha$. An arrival rate vector outside $\mathcal{C}^\alpha$ 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 [121].

The rate region for the 4GBB DSL setting (see Section 9.5.1) is convex, and thus the MDV scheduler is throughput optimal. The rate region for 5GBB is also convex. Vectoring in 5GBB can be applied across all users and cancels out all crosstalk, and eliminates the need for a cross-layer scheduler. However, when users from different DSLAMs are grouped together, full vectoring is not possible and grouped vectoring must be applied (see Section 9.5.2). In this case, cross-talk is still present, and the rate region 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 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.

9.4.5 Stability in wireless systems

We have just shown that the MDV scheduler operating is throughput optimal when the arrival rate is within a fixed capacity region. In this section, we discuss the stability region for wireless networks. In Chapter 10 we look at the performance of the MDV scheduler in LTE and 5G.

A very important difference between a wireless system and the DSL setting is that the channel state can change from slot to slot, and thus also the instantaneous data rate available to the user. In contrast to the DSL setting and like the other opportunistic schedulers from literature, we will use $\alpha = 0$, i.e. a MW style scheduler where we solve $\arg\max_n \sum \omega_n R_n$.

We abstract the instantaneous achievable channel rate at slot t into the channel state process $\mathbf{H}[t]$, where $\mathbf{H}[t]$ is a vector with N elements, one for each user. A slot is assumed to be small enough such that the channel state can be considered constant for the duration of the slot. The set of achievable data rate vectors in a state $\mathbf{H}$ is denoted $\mathcal{C}(\mathbf{H})$.

The convex hull of this rate region can be formed by time-averaging two or more possible rate vectors:

$$\text{conv } \mathcal{C}(\mathbf{H}) = \{\phi_H(\mathbf{H}) | \forall \phi_H\},$$

where ϕ_H is a scheduling policy that depends only on the channel state $\mathbf{H}$. If we assume that the channel state process $\mathbf{H}[t]$ is ergodic, i.e. any of the states can be visited from any of the states with non-zero probability within a finite period, then we can define the ergodic capacity region as

$$\overline{\mathcal{C}} = \{\mathbb{E}[\phi_H(\mathbf{H})] | \forall \phi_H\}.$$

This ergodic capacity region is convex. This result holds only for the channel-only policies ϕ_H. Lemma 3.1 of [169] states that for an ergodic channel state process, any scheduling policy ϕ will produce an average service data rate vector that lies in the ergodic capacity region $\overline{\mathcal{C}}$:

$$\mathbb{E}[\phi(\mathbf{H})] \in \overline{\mathcal{C}}, \forall \phi.$$

The time-dependent nature of the wireless network is not important for the long-term rates: assume that $\mathbf{R}^*(\mathbf{H})$ is the solution for the channel state-dependent optimization problem

$$\mathbf{R}^*(\mathbf{H}) = \underset{\mathbf{R} \in \mathcal{C}(\mathbf{H})}{\arg\max} \sum_n \phi(\omega^n, R^n),$$

with weights ω. Then lemma 3.3 in [169] states that

$$\tilde{\mathbf{R}}^* = \mathbb{E}[\mathbf{R}^*(\mathbf{H})] = \underset{\tilde{\mathbf{R}} \in \overline{\mathcal{C}}}{\arg\max} \sum_n \phi(\omega^n, \tilde{R}^n).$$

Thus, the optimal solution for weights ω in the ergodic capacity region is the same as the expected value of the solution to the problem of the channel state-dependent problem. Hence, the proof presented in Section 9.4.4 remains valid for wireless systems, if we substitute the DSL's rate region with the ergodic capacity region.

9.4.6 Queue performance for a discrete rate region

In Section 9.4.4 we discretized a fluid rate region to discuss the stability region. This section has a note on the behavior of MD-style schedulers 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 9.11, 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 $\lambda = [0.1, 0.1]$ packets/time unit. It is clear that the arrival rate vector is well within the capacity region $\mathcal{C}$.

Figure 9.12 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.

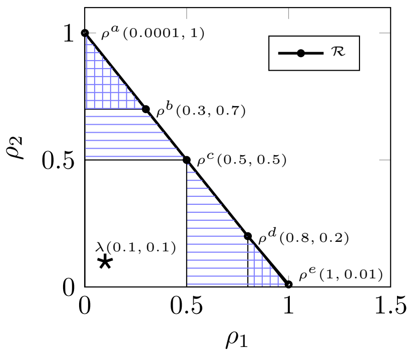


Figure 9.11: Rate regions for the example of Section 9.4.6

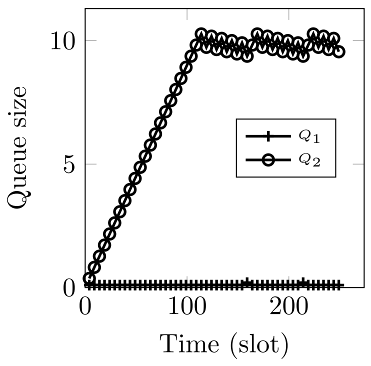


Figure 9.12: Queue evolution of a MD scheduler with two rate points

9.5 Physical layer model

So far we have discussed the upper layers only. In this section, we will look at the physical layer. The scheduler has been applied to 4GBB and its successor 5GBB, both of which are discussed in this section.

9.5.1 4GBB

For the 4GBB simulations we consider an N user DSL system. DSL employs discrete multitone (DMT) modulation in order to establish K orthogonal sub channels or tones. As spectrum coordination is considered, each of these tones k can be modeled as an interference channel.

$$\boldsymbol{y}_k = H_k \boldsymbol{x}_k + \boldsymbol{z}_k \tag{9.16}$$

In (9.16), $\boldsymbol{x}_k = \left[x_k^1, \ldots, x_k^N\right]^T$ is a vector containing the transmitted signal of all N users on tone k. Also, let $\boldsymbol{x}^n = [x_1^n, \ldots, x_K^n]^T$ and let $\boldsymbol{x} = \left[\boldsymbol{x}^{1T}, \ldots, \boldsymbol{x}^{NT}\right]^T$. Similar vector notation will be used for other signals, as well as for variables introduced later such as the bit loading, total power consumption, and data rate. Furthermore, $\boldsymbol{y}_k$ and $\boldsymbol{z}_k$ contain the received signal and noise for all N users on tone k. The average power of x_k^n is given as $s_k^n = \Delta_f \mathcal{E}\left\{|x_k^n|^2\right\}$, with $\mathcal{E}\{\cdot\}$ the expected value operator and Δ_f the tone spacing. Also, $\sigma_k^n = \Delta_f \mathcal{E}\left\{|z_k^n|^2\right\}$ is the average noise power received by user n on tone k. Finally, H_k is the $N \times N$ channel matrix, where $[H_k]_{n,m} = h_k^{n,m}$ is the transfer function between the transmitter of user m and the receiver of user n, evaluated on tone k.

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

$$b_k^n(\boldsymbol{s}_k) = \log_2\left(1 + \frac{1}{\Gamma} \frac{|h_k^{n,n}|^2 s_k^n}{\sum_{n \neq m} |h_k^{n,m}|^2 s_k^m + \sigma_k^n}\right), \tag{9.17}$$

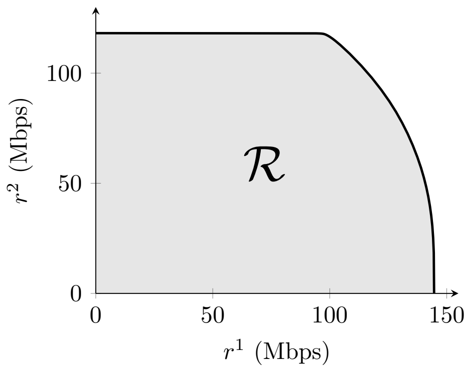


Figure 9.13: Rate region of a 2-user G.Fast system.

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

$$R^n(\boldsymbol{b}^n) = f_s \sum_{k=1}^{K} b_k^n \qquad\qquad P^n(\boldsymbol{s}^n) = \sum_{k=1}^{K} s_k^n, \tag{9.18}$$

where f_s is the symbol rate.

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

$$\mathcal{S}^n = \left\{\boldsymbol{s}^n \in \mathbb{R}_+^K \mid P^n(\boldsymbol{s}^n) \leq P^{\text{tot}} \,\text{and}\, \boldsymbol{s}^n \leq \boldsymbol{s}^{\text{mask}}\right\}. \tag{9.19}$$

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

$$\mathcal{B} = \boldsymbol{b}(\mathcal{S}) \tag{9.20}$$

Finally, we define the rate region as

$$\mathcal{R} = \left\{\boldsymbol{r} \in \mathbb{R}_+^N \mid \exists\, \boldsymbol{r}' \in \boldsymbol{R}(\mathcal{B}) : \boldsymbol{r} \leq \boldsymbol{r}'\right\}. \tag{9.21}$$

As an example, the rate region of a 2-user G.Fast system that employs spectrum coordination is depicted in Figure 9.13. Generally, there is no power allocation that simultaneously maximizes the data rate of all users.

Parameter settings for the DSL system used in the simulations are summarized in Table 9.2. Spectral mask constraints were not included.

Table 9.2: G.Fast parameter settings for the 4GBB scenarios

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

9.5.2 5GBB

In 5GBB crosstalk between users in the same group has been largely eliminated. However, crosstalk between different groups still exists, and it is in this setting that the MDV scheduler has been applied. In this section we discuss the discretization of the rate region to avoid problems when comparing the performance of the different schedulers.

In the 5GBB simulations of Section 9.6 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 [187, 165] 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 section uses zero-forcing grouped vectoring as in [187]. In such a system vectoring matrices are fixed and different rate trade-offs can be made by varying the transmit power allocation $\mathbf{s}$. The power allocation $\mathbf{s}$ that is to be employed in each time slot can be determined by solving the following NUM problem:

$$\max_{\mathbf{s}\in\mathcal{S}} \sum_{n=1}^{N} u^n(R^n(\mathbf{s})). \tag{9.22}$$

In (9.22) we have the utility function u^n from (9.1). $R^n(\mathbf{s})$ expresses the rate of user n as a function of $\mathbf{s}$, the transmit power allocation of all users, which is chosen from the set of feasible power allocations $\mathcal{S}$. The NUM problem in (9.22) contains the spectrum coordination problem from [122] as a special case, and is therefore NP-hard [122]. As such, the locally optimal solution to (9.22) 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 (9.22) 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 use a discrete set of power allocation settings $\hat{\mathcal{S}} \subset \mathcal{S}$ from which the scheduler can choose a single $\mathbf{s} \in \hat{\mathcal{S}}$. The achievable set of rate vectors will be defined as

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

Each time-slot, the scheduler then chooses a power allocation $\mathbf{s} \in \hat{\mathcal{S}}$ by evaluating the objective function of (9.22) for each $\mathbf{r} \in \hat{\mathcal{R}}$ 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 (9.22). 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.

In Section V of [184] we describe an algorithm that selects a representative set of DSL resource allocations.

Three G.fast networks are considered for the simulations of Section 9.6: 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 9.14 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 [181]. 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 9.3 (we refer to [187] for further details).

9.6 Performance 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, which has access to future arrivals and uses this information to reduce the amount of delay violations, while optimizing the throughput. In Section 9.6.1 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 9.6.2.

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 9.3: Summary of G.fast parameter settings

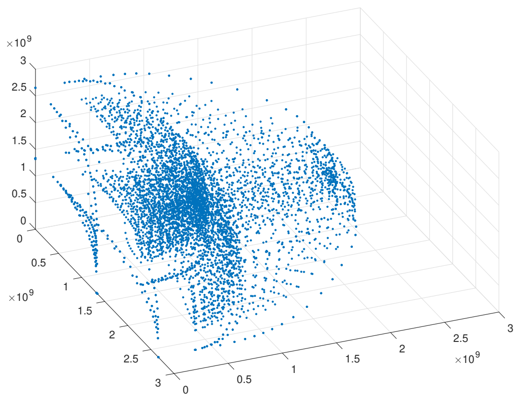


Figure 9.14: Rate region for 3 groups with 2 users (in bit/second)

9.6.1 Simulation setup

In this section, we describe how the evaluation was performed. First we give an overview of the settings and schedulers used. In Section 9.6.1.1 we briefly show the intra-user scheduler settings used in the simulations. We continue with enumerating the metrics that we have used in Section 9.6.1.2 and introducing the plot layouts in Section 9.6.1.3.

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 $5s$, 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.

Scheduler	Weight	Notes
EXP-MLWDF [19]	$\frac{\alpha_i}{\overline{\rho}_i} \exp\left(\frac{\hat{T}_i}{\hat{T}_i - \Gamma_i}\right)$	
EXP/PF [151]	$\exp\left(\frac{\alpha_i \Gamma_i - \overline{\Gamma}}{1+\sqrt{\overline{\Gamma}}}\right)\frac{1}{\overline{\rho}_i}$	$\overline{\Gamma}$ is the average HOL of all real-time flows
Lei [113]	$\frac{a_i \exp(-a_i \cdot (\Gamma_i - \hat{T}_i))}{(1+\exp(a_i \cdot (\Gamma_i - \hat{T}_i)))^2}$	$a = 1.5$
JUPS [11]	$\frac{1}{\overline{\rho}_i} \cdot \frac{-\log(\varepsilon_i)\Gamma_i}{\hat{T}_i}$	
MQS [30]	$\frac{\exp(a_i)}{\exp(a_i)+\exp(-a_i)}$	$a_i = \frac{\Gamma_i - \hat{T}_i}{\hat{T}_i}$
MPT [199]	$\Gamma_i \frac{\hat{T}_i - \Gamma_i}{\exp(\sqrt{1+\hat{T}_i - \Gamma_i})}$	
QHMLWDF [138]	$\frac{\alpha_i \Gamma_i q_i}{\overline{\rho}_i}$	
Wu (2020) [198]	q_i	Uses utility function $u_i(R) = q_i \cdot \log(1+R)$
eEXPRule [202]	$\exp\left(\frac{5\Gamma_i}{\hat{T}_i(1+\sqrt{\overline{\Gamma}})}\right)$	$\overline{\Gamma}$ is the average HOL of all real-time flows
Static		Fixed allocation, set by solving at the start $\arg\max_{\mathbf{r}\in\mathcal{R}} \{\max_{t\in\mathbb{R}}\{t \vert t \cdot \lambda \leq \mathbf{r}\}\}$
MDV	$-\tilde{\lambda}_i \cdot c_i \left(\frac{q_i}{(R_i+\zeta)\hat{T}_i} + \frac{\Gamma_i}{\hat{T}_i} + \ln_2\left(1+\frac{p_i}{\varepsilon_i}\right)\right)$	Reciprocal scheduler. $\tilde{\lambda}$ is a prediction of future arrival.
Oracle		See text and Appendix E

Table 9.4: Summary of the schedulers used in the simulations (in no particular order) and their settings

The schedulers that are used in the simulations are summarized in Table 9.4. We have used some common notation for a flow i in that table: $\overline{\rho}_i$ is the averaged service rate, $\hat{T}_i$ the delay upper bound, ε_i the delay violation upper bound, Γ_i the head-of-line, and $\alpha_i = -\ln(\varepsilon_i)/\hat{T}_i$. The complexity of these schedulers from literature are all comparable, and are negligible, compared to solving the NUM problem.

We also include two special schedulers. The first scheduler, Static, is a “no scheduling” scheduler. At the start of the simulation, the scheduler selects the configuration

$$\mathbf{r}^* = \arg\max_{\mathbf{r}\in\mathcal{R}} \left\{ \max_{t\in\mathbb{R}} \{t \vert t \cdot \lambda \leq \mathbf{r}\} \right\}, \tag{9.23}$$

where λ is a flow’s average arrival rate. It is this configuration $\mathbf{r}^*$ that is used for the duration of the simulation. This allows us to compare static allocation, that is used in current DSL installments, versus dynamic resource allocation, our improvement.

The second scheduler is an approximation to an *ideal* scheduler, called the Oracle scheduler, the pseudocode of which is given in Appendix E. 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 the 4GBB and 5GBB settings for different scenarios. The respective settings were described in the previous section. Each scenario is 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, 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.

Packets are dropped when their waiting time exceeds their delay deadline. We also ran simulations where packets were never dropped. This showed an even larger difference between the MDV and the other schedulers, especially in terms of the delay. In some scenarios, delays of over 5s were encountered. We opted for removal of packets that violate their deadline, as those packets are considered useless to the receiver, squanders capacity and does not allow for fair comparison. This mechanism, however, causes that the sole use the HOL might get wrong information about the state of the flow. If for example all the packets in a queue have just exceeded their deadline, and are removed, then to the scheduler it might seem that the flow is perfectly fine.

9.6.1.1 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 [118], which serves the flow whose HOL packet has the most stringent deadline. The

NUM problem is then reduced to

$$\arg\max_{\mathbf{R}\in\mathcal{R}} \sum_{n=1}^{N} \omega^n R^n \tag{9.24}$$

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

For the EDF scenarios, we have omitted the Oracle scheduler.

9.6.1.2 Metrics

We assess the schedulers' performance using the following metrics:

- **Packet loss ratio** $(\frac{A_p(0,T_{sim})-E_p(0,T_{sim})}{A_p(0,T_{sim})} - \varepsilon)^+$: 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, and T_{sim} the length of the simulation. We subtract ε to take the QoS into account. 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.

9.6.1.3 Plot layout

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

1. **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.
2. **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.

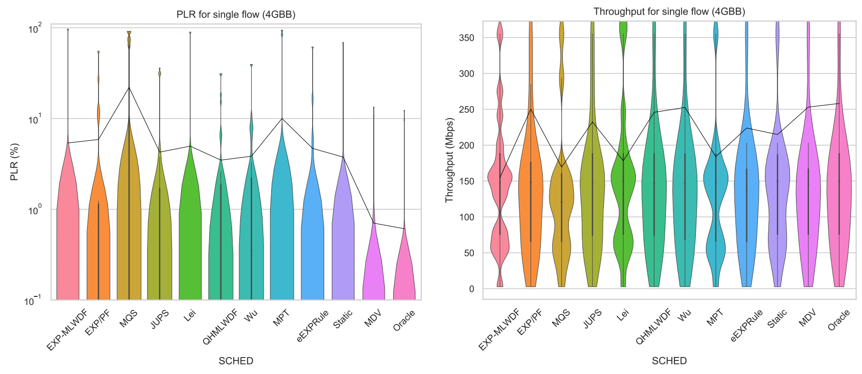
9.6.2 Results

In this section, we discuss the results of the simulations. First, in Section 9.6.2.1, we consider scenarios in which each user has exactly one flow. Then we continue with scenarios representing typical use cases in Section 9.6.2.2. In Section 9.6.2.3 we discuss the schedulers when the system load is close to 1. Scenarios that focus on self-similar traffic are shown in Section 9.6.2.4, while statistical multiplexing is discussed in Section 9.6.2.5.

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

9.6.2.1 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.

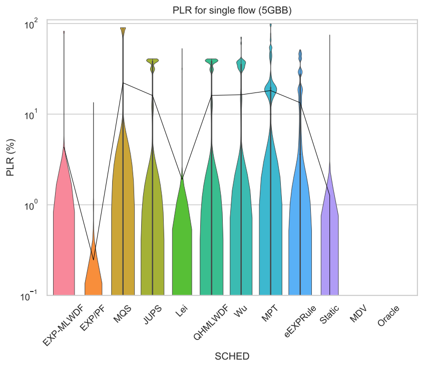


(a) PLR (b) throughput

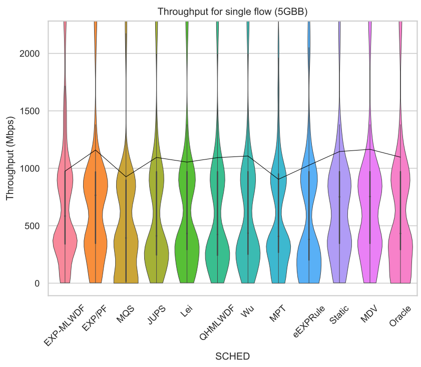
Figure 9.15: The PLR and throughput for scenarios with one flow per user (4GBB)

In Figures 9.15a and 9.16a we can see the PLR distribution for the flows of the scenarios for the two different DSL settings. 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 10%. The Static scheduler has a similar performance as the other schedulers, and in Figure 9.16a even performs better than some of the regular schedulers.

The throughput plot shown in Figure 9.16b shows that the MDV scheduler has among the highest throughput performances. The Oracle scheduler's throughput is less as it trades service to achieve a lower PLR .



(a) PLR

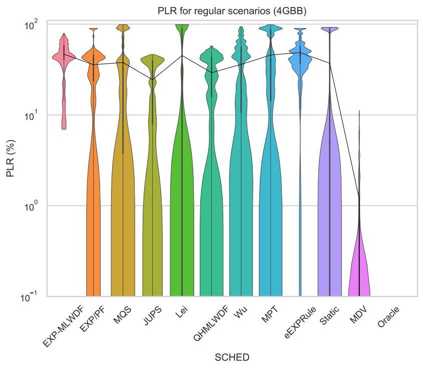


(b) throughput

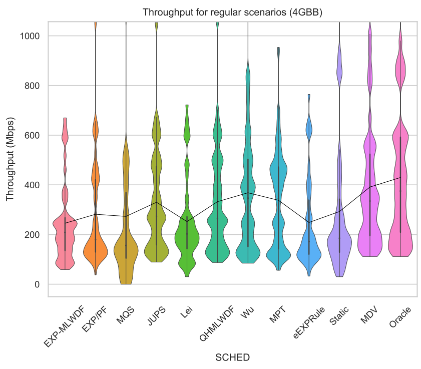
Figure 9.16: The PLR and throughput for scenarios with one flow per user (5GBB)

9.6.2.2 Regular scenarios

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



(a) PLR

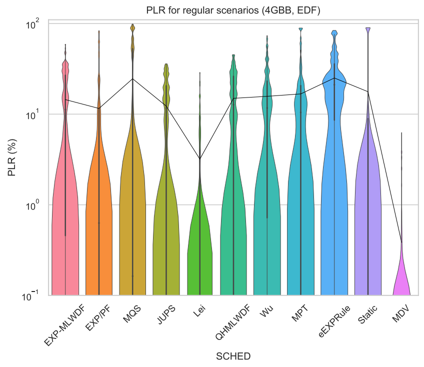


(b) throughput

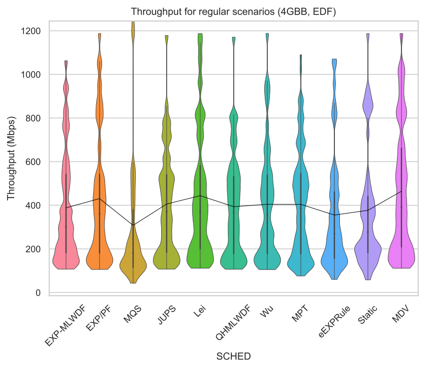
Figure 9.17: The PLR and throughput for regular scenarios (4GBB)

In the scenarios for 4GBB DSL setting, we can see that we achieve a significantly lower PLR than the other scheduler (Figure 9.17a), while also obtaining a higher throughput (Figure 9.17a). The main culprits of the violations for the other schedulers are the video flows.

In Figures 9.18a and 9.20a we show the same scenarios, but now we make use of an EDF scheduler to multiplex all the applications of a user. Unsurprisingly, the PLR decreases and the throughput increases, however, the MDV scheduler still performs better than the other schedulers.

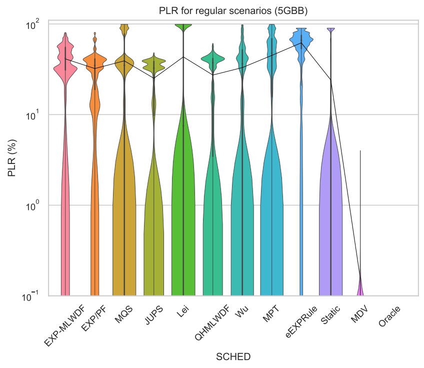


(a) PLR

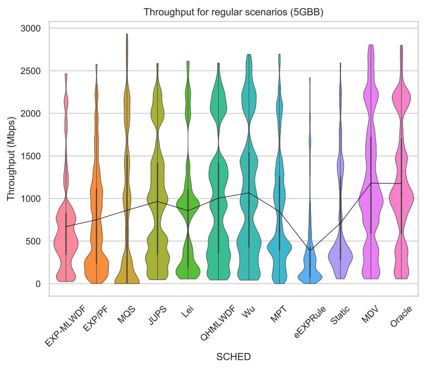


(b) throughput

Figure 9.18: The PLR and throughput for regular scenarios for the EDF scheduler (4GBB)



(a) PLR

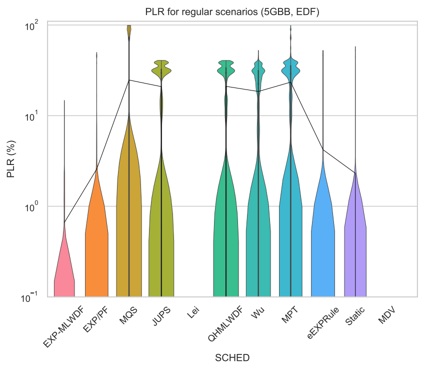


(b) throughput

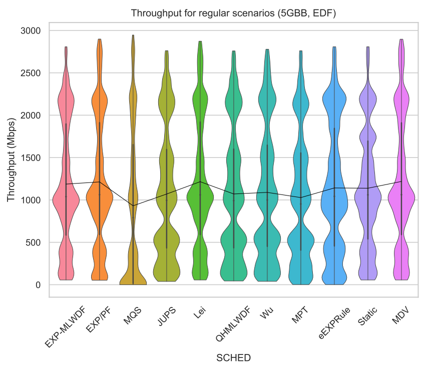
Figure 9.19: The PLR and throughput for regular scenarios (5GBB)

Figure 9.19a shows the PLR distribution for the flows for the regular 4- and 6-user scenarios for the 5GBB setting. 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



(a) PLR

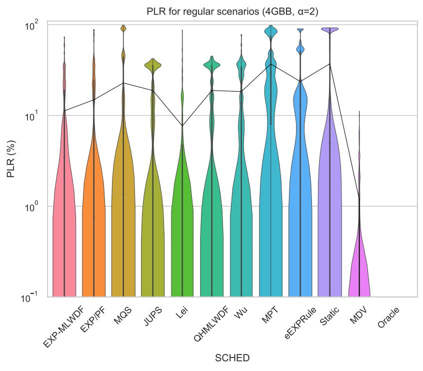


(b) throughput

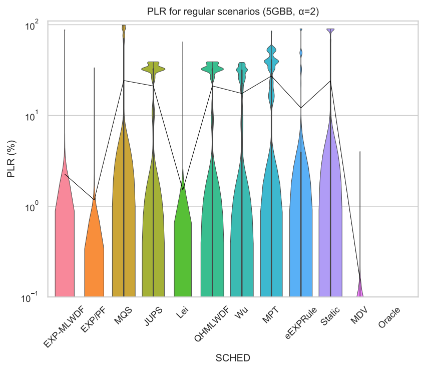
Figure 9.20: The PLR and throughput for regular scenarios for the EDF scheduler (5GBB)

PLR, we also have a higher throughput, as can be observed in Figure 9.19b.

Figure 9.20 shows the same scenarios, but now allocating resources per user, and scheduling flows with the EDF scheduler, as described in Section 9.6.1.1. It should, again, not surprise us that the PLR is lower for the linear schedulers as flows for a single user are multiplexed. Notably, the Lei scheduler now has also a PLR of 0%. This is also visible in the increased throughput.



(a) PLR



(b) PLR

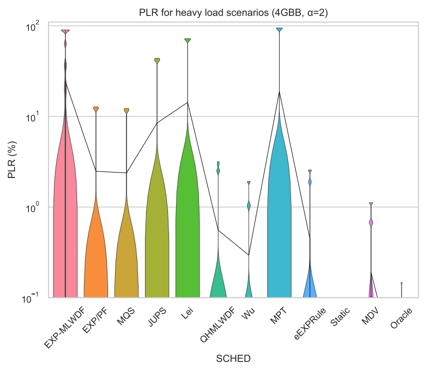
Figure 9.21: The PLR for regular scenarios using $\alpha = 2$, i.e. MD-style schedulers (4GBB and 5GBB)

The schedulers from Table 9.4 all calculate a weight per flow that is used to solve $\arg\max_R \sum \omega R$. This formulation is an instance of the family of α-fair schedulers

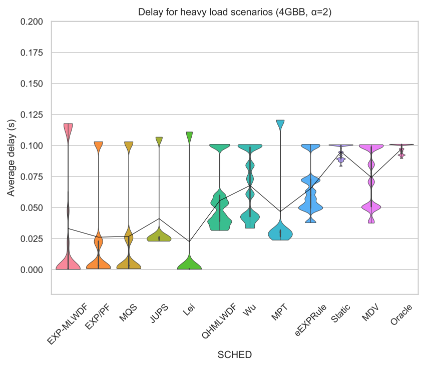
$$\arg\max_R \sum \omega \frac{R^{1-\alpha}}{1-\alpha},$$

with $\alpha = 0$. For the MDV scheduler, we have $\alpha = 2$. In Figure 9.21 we set α to 2 for all the schedulers. This allows for a better comparison of the accuracy of the weight selection. We can see that the average PLR decreases. This is mainly true for the video flows, that comprise the peaks of the histograms. The MDV scheduler, however, still has a better weight selection, with a lower PLR.

9.6.2.3 High load

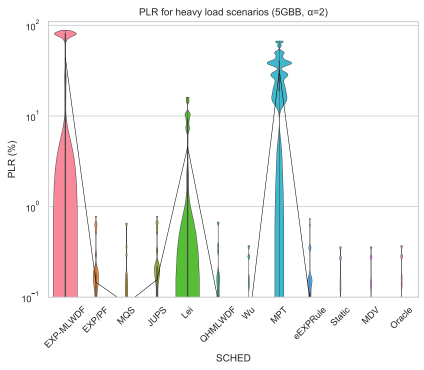


(a) PLR

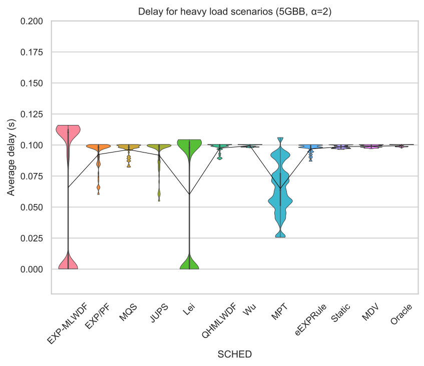


(b) average delay

Figure 9.22: The PLR and average delay for high load scenarios (4GBB, $\alpha = 2$)



(a) PLR



(b) average delay

Figure 9.23: The PLR and average delay for high load scenarios (5GBB, $\alpha = 2$)

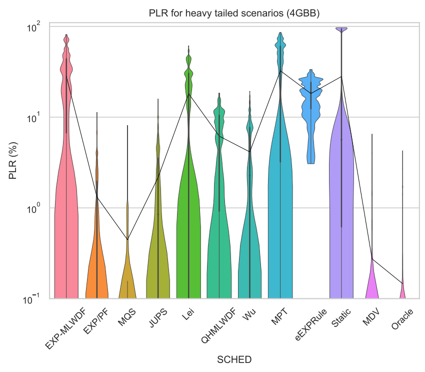
The scenarios from this subsection deal with traffic that is close to the rate region boundary, resulting in a high system load. Each user sends either one flow of type M/M/1 or CBR that averages to $0.99 \cdot \mathbf{R}^*$, where $\mathbf{R}^*$ is the solution to the NUM problem for weights $\mathbf{1}$.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

In Figure 9.22a we can see that in the 4GBB scenarios the Static scenario performs surprisingly well. This is because the traffic arrivals in an interval are quite stable. However, we can see that most schedulers have difficulty coping with the large amount of traffic.

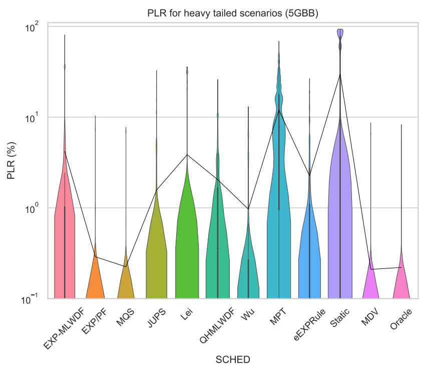
In Figure 9.23a the situation is different. Most schedulers perform well, except EXP-MLWDF, Lei and MPT. In the case of the first two schedulers this is because all service goes to all users of one group, resulting in the two distinct blobs in the delay and PLR plots. For the MPT scheduler all flows suffer many packet losses.

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 example for the EXP-MLWDF scheduler in the delay plot Figure 9.23b. A couple of 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.

9.6.2.4 Self-similar traffic



(a) PLR

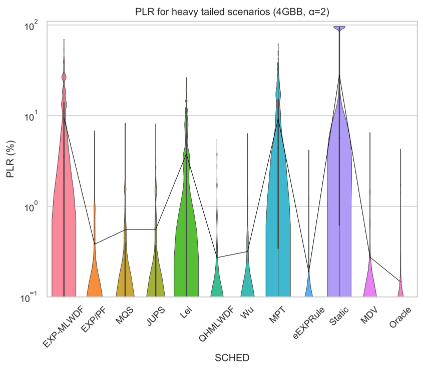


(b) PLR

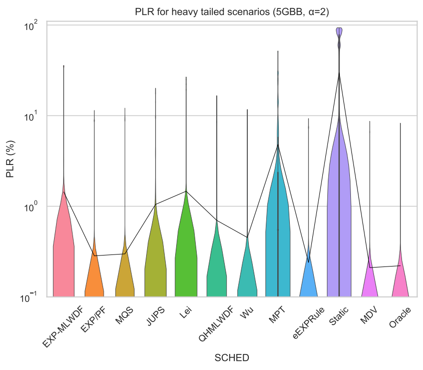
Figure 9.24: The PLR for the self-similar traffic scenarios (4GBB and 5GBB)

In the self-similar 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 self-similar traffic [179]. A self-similar process behaves the same when looking at its properties over different time scales. For example Ethernet traffic shows strong signs of self-similarity [115]. A challenge with self-similar processes is that the burstiness it exhibits can occur on various time scales, making accurate estimates on the average and variance very difficult.

The PLR in Figure 9.24 shows that, indeed, the Static allocation has difficulty



(a) PLR

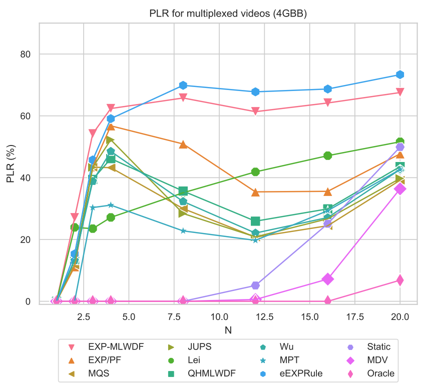


(b) PLR

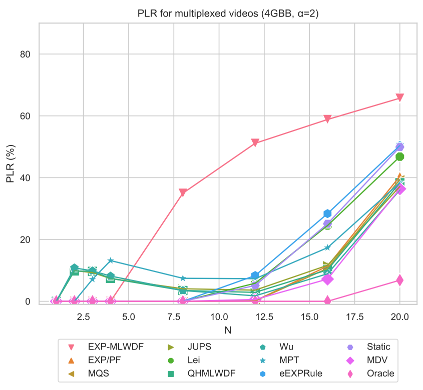
Figure 9.25: The PLR for the self-similar traffic scenarios (4GBB and 5GBB, $\alpha = 2$)

with the traffic, as its average arrival rate is usually inaccurate. However, even if it would be accurate, then a lot of capacity would be wasted as the tail, the probability of large values occurring, of a self-similar process can be quite large. The MDV scheduler has an almost similar performance as the Oracle scheduler, indicating that it can cope with the self-similar traffic, in both the 4GBB and 5GBB scenarios.

9.6.2.5 Statistical multiplexing



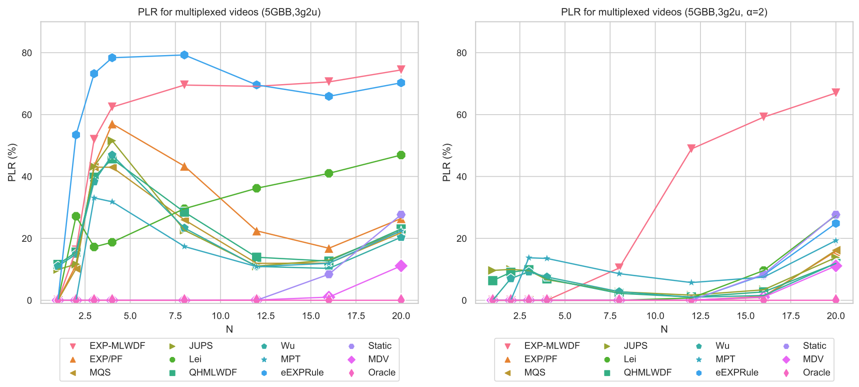
(a) PLR



(b) PLR using $\alpha = 2$

Figure 9.26: The PLR for multiplexing Starwars videos for the 4GBB rate region using regular style and MD-style scheduling

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



(a) PLR (b) PLR using $\alpha = 2$

Figure 9.27: The PLR for multiplexing Starwars videos for the 3g2u rate region using regular style and MD-style scheduling

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 subsection looks at the schedulers' performances when each user sends a number of Starwars videos [155]. The x-axis in the plots in Figure 9.26 represents the number of videos each user is transmitting. Each video starts transmitting at a random offset within the video. On the y-axis, we can see the average of the metric.

For $\alpha = 0$ we can see in figures Figures 9.26a and 9.27a that the schedulers immediately are performing badly, even for a low number of traffic sources. As the number of sources increases further, the PLR decreases a little to rise again later on. When using the MD-style schedulers in Figures 9.26b and 9.27b we can observe the PLR decrease for most schedulers.

The static allocation performs well for a low number of video sources per user, however, as of 12 sources the PLR increases quickly. The MDV scheduler remains well below it, but for many sources its performance with respect to the Oracle scheduler deteriorates. Compared to the other schedulers, the MDV schedulers always performs better.

9.7 Conclusion

In this section, we have presented a new cross-layer scheduler for DSL networks. The goal of this scheduler is to move from the static allocation that is currently the practice in DSL networks, to a dynamic approach that assigns services rate in function of the short- and long-term requirements. The scheduler combines the arrival rates, 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, and discussed some of its properties.

We have compared the PLR, throughput and delay performance to other cross-layer schedulers from literature, together with an Oracle scheduler and static allocation, and demonstrated that it performs at least as good, and usually substantially better, than the other schedulers in the tested scenarios and can improve significantly upon a static allocation. Other schedulers from literature, often developed in the context of wireless networks, have a similar PLR performance as the MDV scheduler in some scenarios. However, the MDV scheduler consistently has low PLR and high throughput, that is often also very close to an ideal scheduler. The simulations also highlight the importance of using an MD-style scheduler for use in a DSL cross-layer setting, rather than the opportunistic MW-style schedulers. But even when we apply the MD-style to the schedulers from literature, the MDV scheduler still performs better. Our 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.

Chapter 10

Cross-layer scheduling in LTE and 5G

10.1 Introduction

With the ubiquity of small and capable devices such as smartphones, LTE networks are increasingly important for e.g. VoIP and video calls. LTE is the evolution of universal mobile telecommunications system (UMTS), and improves upon it in capacity, latency and speed. A standard LTE system consists of two main parts: the evolved UMTS terrestrial radio access network (E-UTRAN) and the evolved packet core (EPC). The EPC is the LTE core network and entails the components that provide functionality such as mobility management, routing, quality of service etc. The radio access network (RAN) part comprises two types of nodes, eNodeBs (eNBs) and the user equipments (UEs). eNBs are spread over the landscape, and are the gateway between the RAN and the core network, and control radio related functionality, such as scheduling.

5G has been published by 3rd Generation Partnership Project (3GPP) in 2018, with the 3GPP standard Release 15. 5G New Radio (5G NR) improves upon LTE by increasing the data rates, decreasing the latency and reducing the bit error rate. Two frequency ranges are used, the Frequency Range 1 (FR1) which covers sub-6 GHz frequency bands, and the Frequency Range 2 (FR2) which includes the frequency bands of the millimeter wave (mmWave) range (24 GHz-100 GHz). For FR1 the download speeds are on average a little higher than for LTE, while for FR2 download speeds of up to 10 Gbps are possible. The architecture is similar to LTE, but the eNB is replaced with the gNodeB (gNB).

At the physical layer resources are divided in the time domain in frames with a size of 10 ms, each frame of which is then split in 10 subframes of 1 ms, called the transmission time interval (TTI). Each subframe is finally again split in two equal parts of 0.5 ms. In the frequency dimension, each subframe is split into subchannels of 180 kHz, called the physical resource block (PRB). Each subchannel is then split again in equal parts of 15 kHz. It is such a PRB that is assigned to a single UE. In the downlink direction, any combination of PRBs can be assigned to users. In the uplink direction, only consecutive RBs can be assigned.

The allocation of these PRBs among the UEs is a task for the scheduler, which typically uses QoS requirements, channel quality indicator (CQI) feedback and past assigned bandwidths resources in order to select an allocation that is fair in some sense. Scheduling in LTE has attracted much interest from researchers as there are many metrics and scenarios that can be considered, and no implementation of a scheduler has been prescribed by the 3GPP.

The motivation for a new scheduler usually revolves around finding a suitable balance between optimally using the achievable data rates and fairness, i.e. *opportunistically* scheduling users. On the one side, we have the max throughput (MT) scheduler which allocates RBs to the users that experience the best bit-rates, resulting in the maximal system throughput. This comes, however, at the cost of unfairness among the users. On the other side, we can find the channel-blind RR scheduler, which allocates resources sequentially to users. This algorithm is fair, but ignores channel conditions possibly resulting in a low system throughput. The proportionally fair (PF) scheduler [48] stands in the middle between both algorithms, and tries to maintain fairness and an optimal system throughput at the same time. In this algorithm flows that have received a limited amount of service in the past will have a higher probability of receiving more service, even if their channel quality is not as good. The schedulers just presented are examples of channel-aware and QoS-unaware as they do not take the traffic application's requirements into account. Many other channel-aware and QoS-unaware scheduling algorithms exist, see for example in the survey [110]. The focus of this section, however, is on QoS and channel-aware scheduling.

QoS aware strategies mainly focus on the real time (RT) service by considering the delay (M-LWDF [17], EXP/PF [151], EXPRule [159], [19], [30]) or are a hybrid of several stateful properties (FLS [148], queue-HoL-M-LWDF (QHMLWDF) [138]). The authors of [109] propose the packet prediction mechanism (PPM) scheduler which estimates the maximum delay of future queued packets using a virtual queue. In three phases (frequency domain, time domain and PPM) it schedules packets, resulting in an improved overall LTE performance. It does not consider non-real time (NRT) traffic.

The resource allocation algorithm presented in [207] uses the large deviation principle to estimate the buffer overflow probability. Using online measurement the queues are prioritized such that buffer overflow is minimized and QoS is statistically guaranteed.

The research of [8] introduces a LTE downlink scheduler based on queue monitoring. User priorities are assigned to users based on the proximity of the queue to a queue threshold limit. Then one of the three newly defined schedulers is chosen, based on the available resources, resulting in reduced loss and increases system throughput.

In [136] a taxonomy of scheduling algorithms is shown, and introduces a new hybrid LTE scheduler. An overview of LTE schedulers that focuses on content-aware scheduling for video streaming traffic is given in [137]. Another survey can be found in [110] which groups schedulers in QoS-aware and QoS-unaware. An overview that also includes energy aware schedulers is given in [40]. The authors also compare some of the schedulers using simulations. Some additional schedulers are listed in Section 9.2.

The remainder of this section is structured as follows. In Section 10.2 we look at the notation used throughout this section. Then in Section 10.3 we present the MDV scheduler, in which we look at the construction of the weights and briefly discuss the complexity. In Section 10.4 we evaluate the scheduler through simulations in an urban setting and compare its performance with other schedulers from literature. We close this work on the MDV scheduler in LTE and 5G in Section 10.5.

10.2 Notation

In this section we describe the conventions and symbols in Table 10.1 (in addition to those defined in the symbols table on Page 5) that will be used in the remainder of this section. Time is slotted, indexed by variable t, and has a duration of 1 TTI. There are N users connected to the eNB. Every user $u \in [1, N]$ has a number of active applications. The applications in the network are indexed by $i \in [1, \phi]$, with their QoS defined by $P\{d_i(k) > \hat{T}_i\} \leq \varepsilon_i$, where d_i is the application's packet delay distribution, $\hat{T}_i$ the delay upper bound and ε_i the allowed delay upper bound violation probability.

Table 10.1: Symbols

Symbol	Meaning
B	number of RBs
$R_{i,j}[t]$	Achievable data rate in slot t for application i's RB j
$\overline{\lambda}_i[t]$	EMA of the arrival rate
$p_i[t] = \frac{d_{i,p}[t-1000,t]}{a_i[t-1000,t]}$	windowed PLR
$p_i^{\mathrm{LT}}[t] = \frac{d_{i,p}[0,t]}{a_i[0,t]}$	long-time PLR
f_{RT}, f_{BE}, f_{All}	indices of the set of real time, best-effort and all flows
$\overline{(\mathbf{x})}_S = \frac{1}{\lvert S\rvert}\left(\sum_{i\in S} x_i\right)$	the average of all x in the set S

10.3 The minimal delay violation scheduler

In this section we introduce the MDV scheduler for LTE and 5G networks.

QoS-aware LTE schedulers generally assign resource blocks to the application that would benefit most of receiving that resource block. A common way to accomplish this is through a time-dependent utility function $u_i(t, R)$. This function returns the value of an application receiving a certain service rate R. Thus, resource block j is assigned to application i^* where

$$i^* = \underset{i\in[1,\phi]\wedge q_i[t]>0}{\arg\max} \{u_{i,j}(t, R_{i,j}[t]))\}. \tag{10.1}$$

Most cross-layer schedulers are linear in R, and thus (10.1) is often simplified to

$$i^* = \underset{i\in[1,\phi]\wedge q_i[t]>0}{\arg\max} \{w_i[t]\cdot R_{i,j}[t]\} \tag{10.2}$$

where $w_i[t]$ is the application's weight for slot t, and independent of RB j. It is often the calculation of this weight that comprises the cross-layer scheduler.

Here, we will reprise the MDV scheduler described in Section 9.4, and highlight the small changes. In contrast with the DSL context, we here use a linear scheduler. Such a scheduler favors the users who encounter the temporary best instantaneous data rates, rather than fair share the available service. Second, it is complex and inaccurate to characterize the ever-changing rate region over longer periods of time, which is necessary to implement an MD-style scheduler.

10.3.1 Real-time flows

For convenience, we repeat Equation (9.7) from the previous section here. The weight of application i is given by

$$w_i^{\mathrm{RT}}[t] = \underbrace{\tilde{\lambda}_i[t+1]}_{\text{(a)}} \cdot \underbrace{f_c}_{\text{(c)}} \left(\underbrace{\frac{q_i[t]}{(\bar{r}_i[t]+\zeta)\hat{T}_i} + \ln_2\left(1+\frac{p_i[t]}{\varepsilon_i}\right) + \frac{\Gamma_i[t]}{\hat{T}_i}}_{(b)} \right). \tag{10.3}$$

There are two minor changes, compared to the DSL version. The first modification is that we do not update $\tilde{\lambda}_i[t+1]$ every slot (i.e. every TTI=1 ms), but only once every 50 ms (like in the DSL context). The resolution of 1 ms is too small, as in many TTI there are no arrivals, resulting in a lot of noise and inaccurate predictions.

The second change is that for the traffic class-dependent function $f_c(\cdot)$ we have used a different γ_{VoIP} for the VoIP flows. Now, we use $\gamma_{VoIP} = 10$ instead of 1. Due to its low volume, we increase it to ensure it receives enough service. As the arrival rate of VoIP is small, compared to video flows, this should not impact the PLR of the other flows significantly.

10.3.2 Best-effort flows

The weight for BE flows remains unchanged, and uses the same values as the DSL version of the MDV scheduler, so we get

$$w_i^{\mathrm{BE}}[t] = \overline{(\overline{\lambda}[t])}_{f_{\mathrm{RT}}} \cdot \Omega \cdot \frac{\check{R}}{\overline{R}_i[t]}. \tag{10.4}$$

10.3.3 Pseudocode

In Algorithm 10.1 we show the code that assigns the RBs to the applications. The code in Algorithm 10.2 is run every $\tau = 50$ slots, and updates the average arrival rate and prediction of the arrival rate.

Algorithm 1: Assign resource blocks

```
$\check{R} := \arg\min_{j \in f_{\mathrm{BE}}}\{\overline{R}_j[t]\}$
Calculate $\overline{(\overline{\lambda}[t])}_{f_{\mathrm{RT}}}$
for $i$ in RT:
    if $q_i[t] > 0$ and $R_i[t] > 0$:
        $\overline{r}_i[t] := 0.8\overline{r}_i[t-1] + 0.2R_i[t]$
    $w_i := (10.3)$
for $i$ in $f_{\mathrm{BE}}$:
    $w_i := (10.4)$

Initialize $\mathbf{m}$ to a vector with $\phi$ entries
for $j$ in $[1, B]$:
    for $i$ in $[1, \phi]$:
        if $q_i[t] > 0$:
            m[$i$] := $w_i$
        else:
            m[$i$] := 0

    assign RB $j$ to application $\arg\max_i\{R_{i,j}[t] \cdot m[i]\}$
```

Algorithm 2: Update state

```
$\tau = 50$
for $i$ in $[1, \phi]$:
    $\tilde{\lambda}_i.addToCircularBuffer(A[t-\tau, t])$
```

10.3.4 Complexity

In this section we briefly discuss the space and runtime complexity of the MDV scheduler.

10.3.4.1 Space complexity

For every RT flow we need to store W entries for the circular buffer that is used to store the windowed average, and 10 entries for the circular buffers to store the arrivals and packet drops in the past second. Hence, the spacetime complexity is $\mathcal{O}(|f_{\mathrm{RT}}|\cdot(W+2\cdot 10)) = \mathcal{O}(|f_{\mathrm{RT}}|\cdot W)$.

10.3.4.2 Run-time complexity

Each RT flow requires updating every τ slots the state $\tilde{\lambda}_i$ and its average. This runs efficiently in $\mathcal{O}(\frac{|f_{\mathrm{RT}}|}{\tau})$. Best-effort flows require every slot the calculation of $\check{R}$ and $\overline{(\tilde{\lambda}[t])}_{f_{\mathrm{RT}}}$. These run in $\mathcal{O}(|f_{\mathrm{BE}}|)$ and $\mathcal{O}(|f_{\mathrm{RT}}|)$ respectively. This results in a run-time complexity of $\mathcal{O}(\frac{|f_{\mathrm{RT}}|}{\tau}+|f_{\mathrm{BE}}|+|f_{\mathrm{RT}}|) = \mathcal{O}(\frac{|f_{\mathrm{RT}}|}{\tau}+\phi)$. Compared to the complexity of the algorithm to assign the RBs of $\mathcal{O}(B\cdot\phi)$ this is thus negligible.

10.4 Performance evaluation

In this section we will evaluate the MDV scheduler through simulations. We also compare the results to other schedulers from literature. We first discuss the simulation setup, then list the metrics that we consider and we then close this section with the actual results.

10.4.1 Simulation setup

The LTE simulations were run in LTE-Sim [147], while the 5G simulations were performed in its successor 5G-air-simulator [130]. Both technologies were simulated with the same settings. We used the same setup as in [167]. The network topology comprises 19 cells, with each cell containing 1 eNB. The subfrequencies have been distributed to cells as in Figure 10.1, to avoid interference between neighboring cells. Each user has one video, one VoIP and one BE application, and is moving randomly, possibly crossing cells with handover. Every TTI packets that have been in the queue more than $\hat{T}$ seconds are dropped. The simulation parameters are listed in Table 10.2. The schedulers we have used from literature are listed in table Table 10.3.

The simulations we performed included more schedulers, but some were omitted as their performance was not as good and made the plots too unreadable. We also simulated the 19 cell topology where each user has exactly 1 application running. The performance was similar, and hence also not included here.

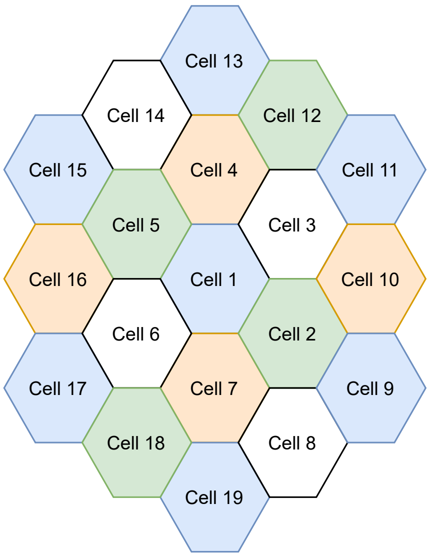


Figure 10.1: The LTE topology for the simulations

Table 10.2: Simulation parameters

Parameter	Values
Simulation length	T=30s
Repetitions	10
Cell radius	0.5 km
Frame structure	FDD
Bandwidth	10 MHz
User speed	{3,30,120} km/h
Number of users	10 .. 100
QoS: $\hat{T}$	100 ms
QoS: ε	1%

10.4.1.1 Metrics

We assess the schedulers' performance for the following metrics:

- **SYSTHR** $\frac{1}{T}\overline{\left(\mathbf{A}\left[0,T\right]\right)}_{f_{All}}$ the average service rate for all the flows. Higher is better.

Table 10.3: Summary of the schedulers used in the simulation (in no particular order) and their settings. Common symbols: $\overline{R}$ (averaged service rate), $\alpha = -\ln(\varepsilon)/\hat{T}$.

Scheduler	Real-time flow weight
EXP-MLWDF (2018) [19]	$\frac{\alpha}{\overline{R}} \exp\left(\frac{\hat{T}}{\hat{T}-\Gamma}\right)$
EF-EDDF (2020) [62]	$\frac{-\log\varepsilon}{\hat{T}} \cdot \frac{\Gamma}{\hat{T}-\Gamma} \cdot \frac{d[0,t]}{\overline{(\mathbf{d}[0,t])}_{f_{\mathrm{RT}}}} \cdot \frac{1}{\overline{R}}$
FLS (2011) [148]	$\frac{1}{\overline{R}}$, with flow selection every 10 TTI
EXPRule (2002) [163]	$\exp\left(\frac{6\Gamma/\hat{T}}{1+\sqrt{\overline{(\mathbf{\Gamma})}_{f_{\mathrm{RT}}}}}\right) \cdot \frac{1}{\overline{R}}$
MQS (2015) [30]	$\frac{\exp(a)}{\exp(a)+\exp(-a)}$
QHMLWDF (2013) [138]	$\frac{\alpha\Gamma q}{\overline{R}_i}$
MPT (2018) [199]	$\Gamma \cdot \frac{\hat{T}-\Gamma}{\exp(\sqrt{1+\hat{T}-\Gamma})}$
MDV	$\tilde{\lambda}_i[t+1] \cdot f_c\left(\frac{q_i[t]}{\overline{r}_i[t]\hat{T}_i} + \frac{\Gamma_i[t]}{\hat{T}_i} + \ln_2\left(1+\frac{p_i[t]}{\varepsilon_i}\right)\right)$

- **THR** $\frac{1}{T}\overline{(\mathbf{A}\,[0,T])}_{f_{\mathrm{BE}}}$ the average service rate for the BE flows. Higher is better.
- **PLR** $\overline{(\mathbf{p}^{\mathrm{LT}}[T])}_{f_{\mathrm{RT}}}$ the percentage of packets that did not arrive at the receiver. Lower is better.
- **Delay** $\overline{(\mathbb{E}[\mathbf{d}])}_{f_{\mathrm{RT}}}$ the average delay encountered for non-dropped packets.
- **Fairness** $\mathcal{J}(\mathbf{x}) = \frac{(\sum x_i)^2}{n \cdot \sum x_i^2} = \frac{1}{1+\sigma/\mu}$, where σ and μ are respectively the standard deviation and mean of $\mathbf{x}$. We use Jain's fairness index (JFI) [92] to assess how fairly a certain metric is distributed among users. For a metric $x_1, \ldots, x_\phi$ $\mathcal{J}(\mathbf{x})$ gives a value in the range $\frac{1}{\phi}$ (worst case) to 1 (best case, when all users receive equal treatment).

10.4.1.2 Plot layout

On the x-axis we show the number of users that were simulated. Each scheduler is shifted a little such that the data points of the different schedulers do not obstruct each other. The most-right scheduler in a group is always the MDV scheduler. The y-axis is the average of the metric over all users and repetitions of the simulations.

10.4.2 Results

We have run several scenarios, such as each user having exactly 1 or exactly three applications and for 30 km/h. In general the results were very similar, so we only show here the results for 3 km/h and 120 km/h.

10.4.2.1 LTE

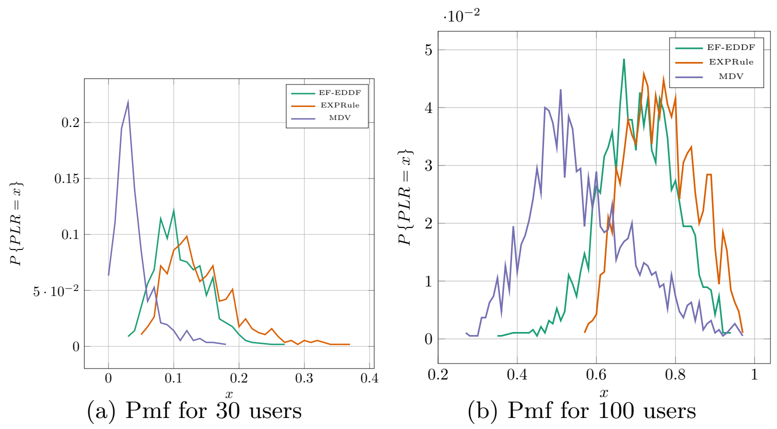
The plots in this section concern the LTE simulations. In Figures 10.4 and 10.5 we show the different metrics for users moving at 3 km/h and 120 km/h respectively.

The PLR for video and VoIP traffic are shown in respectively Figures 10.4a and 10.4b. There we can observe that the average PLR for our scheduler is consistently the lowest. For 100 users the difference between the MDV and the FLS scheduler is 8 percentage point. At the same time, the PLR for VoIP is also the lowest for our scheduler. The throughput for BE, shown in the log-plot Figure 10.4c has the highest throughput for a small number of users. However, as the number of users increases, the throughput becomes low, as the scheduler favors a low PLR. Comparing with the FLS scheduler, the closest in terms of PLR, the MDV scheduler achieves a performance of 4.15 kbps, while the FLS scheduler gets 3.84 kbps. In terms of the system throughput, shown in the log-plot Figure 10.4d, only the MQS and MPT schedulers perform better, at the cost of very high PLR for both video and VoIP for those schedulers.

The JFI fairness index for the MDV scheduler for PLR and BE throughput belongs to the lesser performing schedulers. For the PLR points for a small number of users the JFI is arguably not the ideal metric for the MDV scheduler. The JFI can be written as $\frac{1}{1+\sigma/\mu}$, where σ and μ are the standard deviation and mean, respectively and σ/μ is also called the covariance. The covariance increases quickly as μ gets closer to zero, hence, for means close to zero the JFI becomes more easily disturbed.

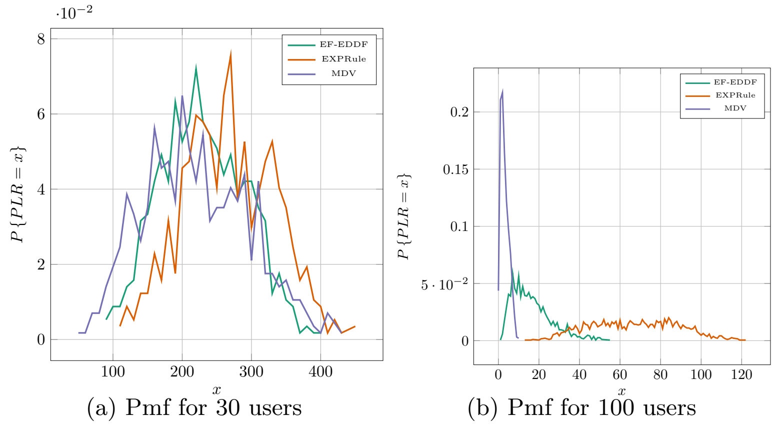
If we look at the pmf of the PLR for 30 users in Figure 10.2a, we can see that the mean for the MDV scheduler is indeed close to zero, and looks plausibly more fair than EF-EDDF and EXPRule schedulers (see also Figure 10.5a for full PLR plot). For 100 users in Figure 10.2b the fairness looks more reasonable, as the mean is relatively larger. Similarly, for the throughput for 100 users, in Figure 10.3, the JFI does not seem to represent the fairness fully.

The delay performance, finally, can be seen in Figures 10.4g and 10.4h for video and VoIP respectively. We can see there that the delay for video is the lowest (except for the MPT scheduler. However, its PLR is very high, hence the average delay is dictated by only the few packets that do arrive). Additionally, the delay of the FLS and MDV schedulers increase almost linearly over the whole range of users, whereas other schedulers show a more sigmoidal delay function. The delay



(a) Pmf for 30 users (b) Pmf for 100 users

Figure 10.2: Pmf for PLR for LTE 120 km/h scenario



(a) Pmf for 30 users (b) Pmf for 100 users

Figure 10.3: Pmf for throughput for LTE 120 km/h scenario

of VoIP also remains very constant, which is very beneficial for the QoE for VoIP calls.

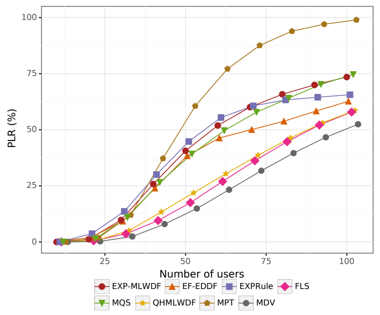
10.4.2.2 5G

The plots in this section concern the 5G simulations. In Figures 10.6 and 10.7 we show the different metrics for users moving at 3 km/h and 120 km/h respectively. As before the plots for 30 km/h were very similar and performing as expected, and have been left out.

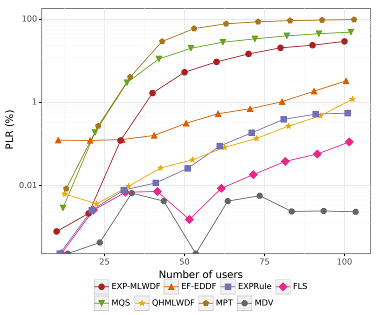
The plots differ for some metrics a little, however, the notes concerning the MDV scheduler remain valid for the 5G simulations.

10.5 Conclusion

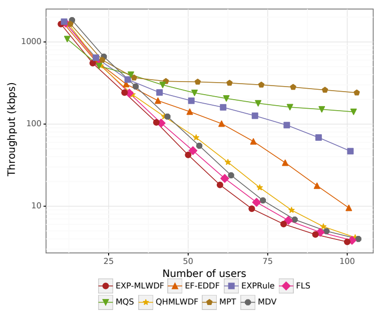
In this section we evaluated through simulations the MDV scheduler from Chapter 9. The MDV scheduler has been slightly adjusted for use in LTE and 5G context. In the simulation results we can see that the excellent performance of the MDV scheduler is not limited to the DSL context. Also in LTE and 5G we achieve a lower PLR and comparable or higher throughput than other schedulers from literature. The fairness of the MDV scheduler usually belongs to the bottom, in some cases because its mean value is close to zero, which skews the result.



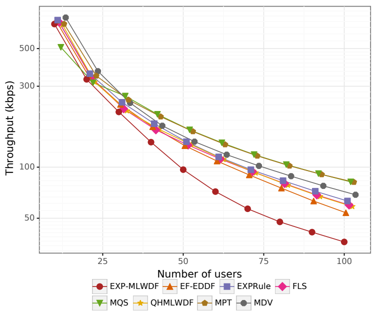
(a) Average PLR for Video



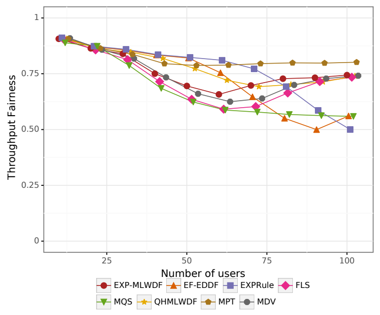
(b) Average PLR for VoIP



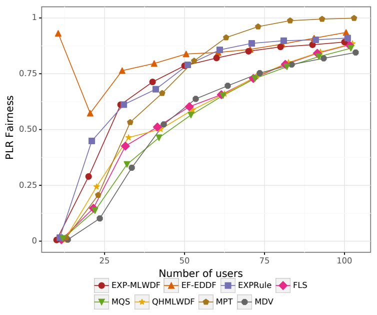
(c) Average throughput for BE



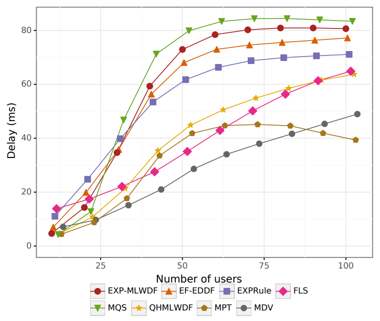
(d) Average system throughput



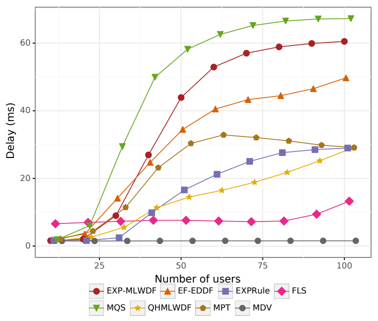
(e) JFI for throughput of BE



(f) JFI for PLR of Video

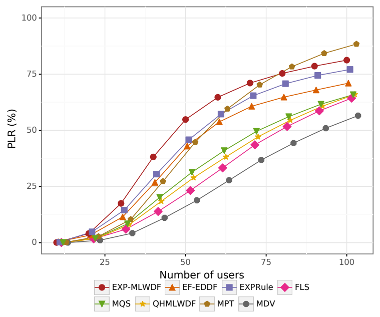


(g) Average delay for video

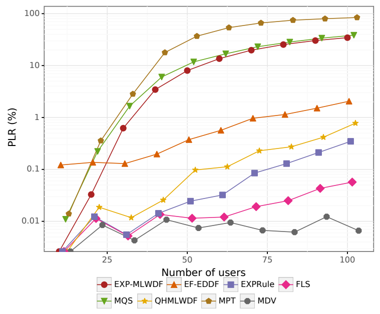


(h) Average delay for VoIP

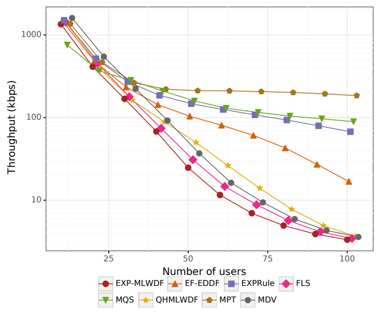
Figure 10.4: Plots for the LTE 3 km/h scenarios



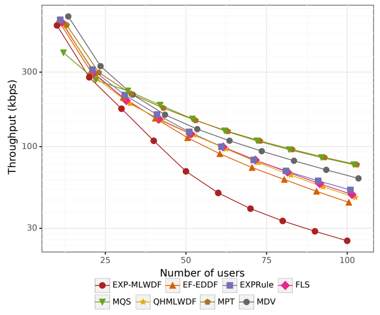
(a) Average PLR for Video



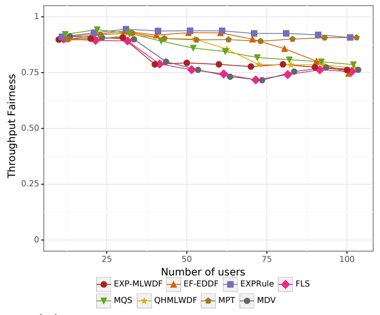
(b) Average PLR for VoIP



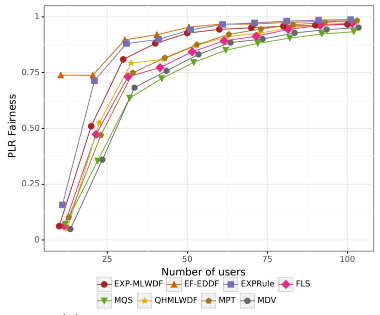
(c) Average throughput for BE



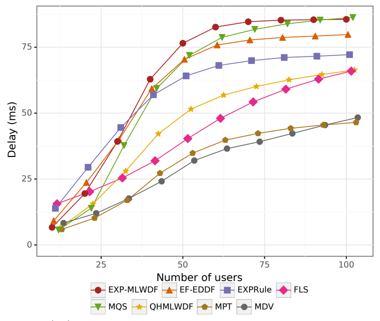
(d) Average system throughput



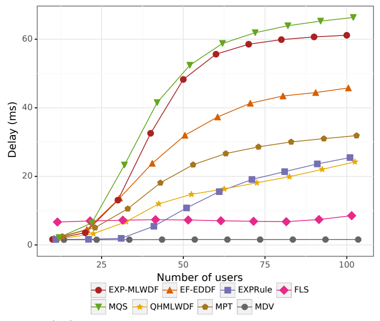
(e) JFI for throughput of BE



(f) JFI for PLR of Video

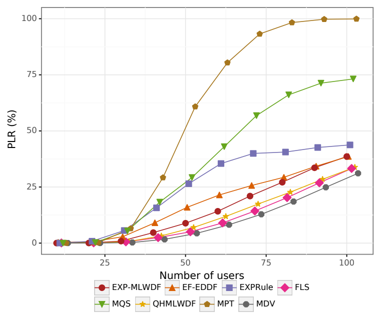


(g) Average delay for video

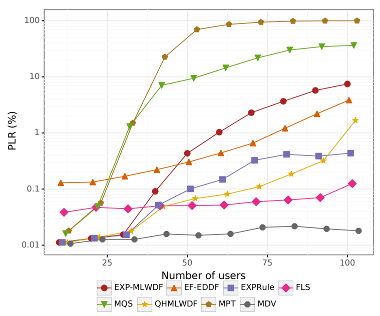


(h) Average delay for VoIP

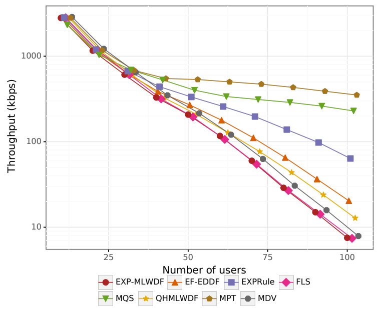
Figure 10.5: Plots for the LTE 120 km/h scenarios



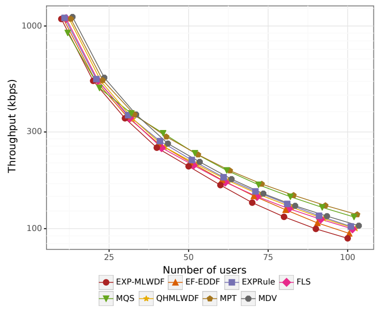
(a) Average PLR for Video



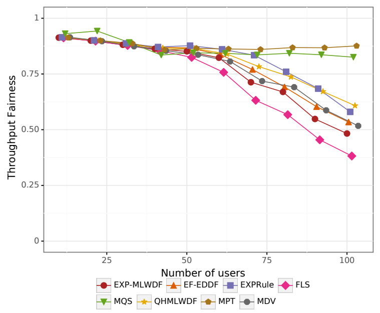
(b) Average PLR for VoIP



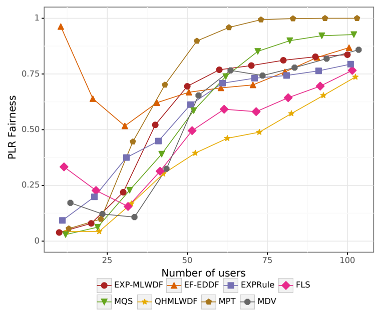
(c) Average throughput for BE



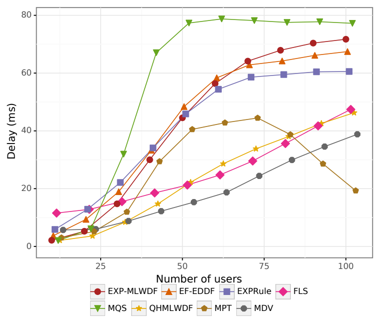
(d) Average system throughput



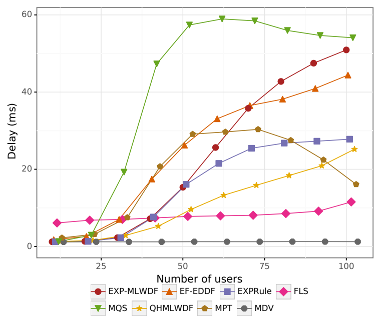
(e) JFI for throughput of BE



(f) JFI for PLR of Video

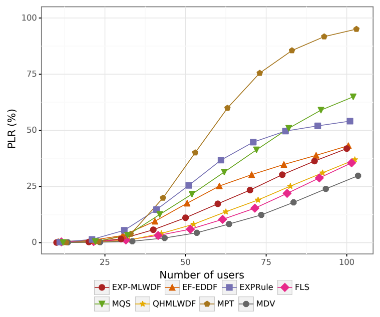


(g) Average delay for video

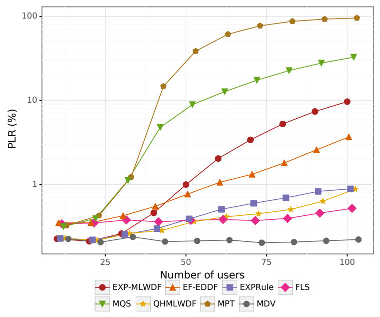


(h) Average delay for VoIP

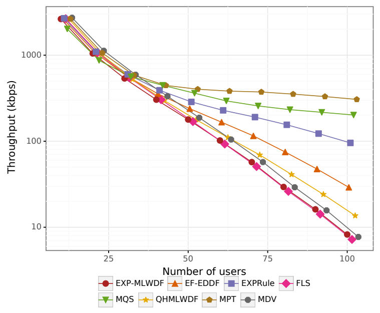
Figure 10.6: Plots for the 5G 3 km/h scenarios



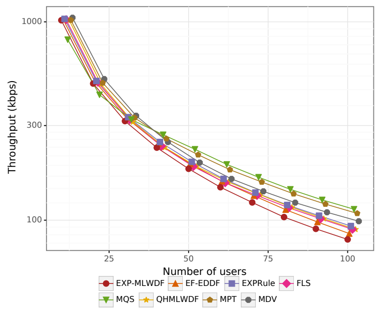
(a) Average PLR for Video



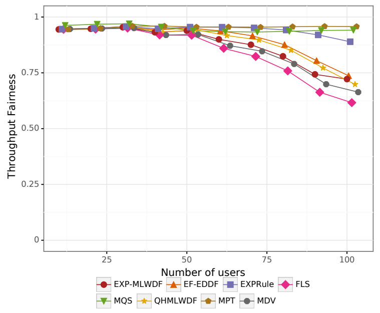
(b) Average PLR for VoIP



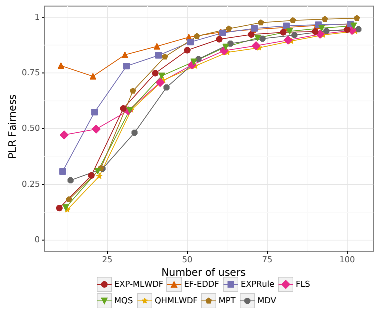
(c) Average throughput for BE



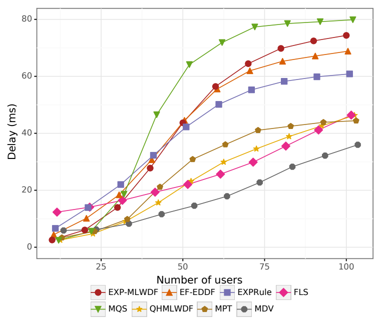
(d) Average system throughput



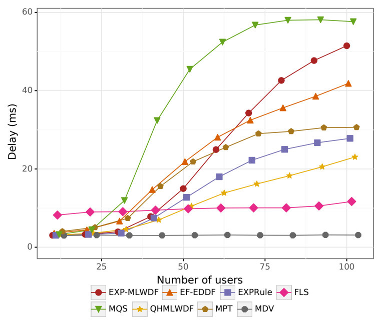
(e) JFI for throughput of BE



(f) JFI for PLR of Video



(g) Average delay for video



(h) Average delay for VoIP

Figure 10.7: Plots for the 5G 120 km/h scenarios

Chapter 11

Cross-layer resource allocation for satellite communication

Geosynchronous earth orbit (GEO) satellite networks carry a high cost-per-bit, and thus it is in the user's interest to limit the reserved capacity. A provider usually aggregates the traffic of different users to take advantage of statistical multiplexing, leading to reduced costs. However, some types of traffic, such as video and VoIP, remain bursty, even when aggregated [115]. In such case one has to dimension for the peak usage to satisfy the QoS demands of the user, squandering capacity. A solution is to periodically adapt the reserved capacity to the actual required usage. In this case a station sends a capacity request to a satellite hub, which replies with a grant, containing the allotted bandwidth. Due to the large propagation delay (of up to 500 ms [83]), it takes some time before the new capacity is in effect. Therefore, an estimate of the future necessary service rate is required at the time of the request. It is paramount that this service rate offered to its users is sufficient to satisfy the user's QoS requirements, and at the same time does not exceed the necessary rate, to save on bandwidth, and thus costs.

Also wired networks can benefit from this approach. Some methods for G.fast standards attempt to overcome crosstalk by selecting the optimal frequency and power assignment over all users. The complexity of these calculations can result in solutions that take in the order of seconds [183]. Due to this crosstalk, these networks can be seen as a shared network, where bandwidth is traded between users. This means that granting one user too much service, can have an impact on other users' service, diminishing its QoS and QoE. Thus, also here the combination of sufficient rate and efficiency is important.

When reserving capacity, a trade-off must be made between delay and throughput efficiency: a lower capacity will increase efficiency at the cost of larger delays, and vice versa.

The contributions of this section are two new methods of dynamic bandwidth allocation (DBA), called *Simple* and *MBAC*, and are described in Section 11.3.

The former is an ad hoc method, based on the arrival rate in past intervals, and is of very low complexity. The second algorithm is based on an admission control (AC) algorithm. We tuned it to work better on shorter intervals, and introduced an addition to reduce the number of parameters that have to be set. We compare these two methods with the ideal case, the *Oracle* algorithm described in Section 11.4.2.1, and one other DBA algorithm from literature. The approach presented here is not dependent on any technology, and can be used in cellular, satellite and wired networks. The focus does not lie on accurate traffic prediction, but rather on dimensioning the intensity of the aggregate, with the goal of achieving a satisfying delay percentile of the aggregate, while keeping the efficiency as high as possible.

In Section 11.1 we look at the related work. The model and notation used throughout this paper are described in Section 11.2. The simulation settings are shown in Section 11.4, while the results are discussed in Section 11.4.3, for both UDP and TCP protocols. We close this section with a conclusion in Section 11.5.

11.1 Related work

Much of the DBA literature on satellite networks targets single flows of media traffic, e.g. [79, 106, 107]. However, we discuss in this work an aggregate of many flows, for which it is more difficult to construct a traffic model.

Likewise, [43] applies a traffic model to predict future arrivals. It is well known that the accuracy of model-based approaches depends on the underlying source models. These are often simplified, sometimes ignoring deeper patterns, such as long-range dependency, and thus not applicable in a wide range of scenarios.

Some literature deals with the allocation of subchannels of a satellite system. However, in the current work, we use an abstraction of the channel, and only work with the rate. For example, [94] addresses allocation in satellite systems while keeping the delay of real-time streams bounded. For each subchannel it calculates the optimal power setting, modelling the channel. Also, [144] optimizes the bandwidth allocation in function of the power. In [142] they also optimize bandwidth allocation as a function of power, using particle swarm optimization.

The algorithm in [201] uses queue statistics and the arrivals in a simple algorithm to predict the demand in the next slot. In [72] the authors propose a predictive approach, using a normalized linear mean square predictor. Other algorithms focus on improving the throughput or achieving a low delay bound, but do not allow for a stochastic bound, making it more difficult to provide different service levels. In [208] two algorithms are proposed, one focusing on obtaining the maximum throughput while the other minimizes the demand-supply variance. The algorithms in [63, 47] also provides a dynamic bandwidth allocation methods. However, they assume that the time between bandwidth requests is less than the round trip delay, whereas we assume bandwidth requests that occur less frequent.

In [206] an algorithm is proposed that solves a corresponding convex optimization problem, resulting in a fair allocation for the different users. Also, [174] tries to optimize the allocation in function of user fairness. However, for our algorithm we have only one stream, which is the aggregate of multiple users.

A lot of work also has been done on traffic prediction e.g. [23, 86]. There are various approaches, such as employing the frequency domain in [49], or through neural networks and wavelet decomposition [26]. In general traffic prediction requires upfront construction of models, or are of a great runtime complexity. Additionally, these prediction models usually lack a confidence level, might be difficult to apply to an aggregate of traffic and often do not take the queue size into account.

11.2 System model

We consider a system where several flows are aggregated over a single channel, from a client to a server. The server is responsible for granting bandwidth requests to the client.

At time $r_i = U \cdot i$, where U is the system slot size and i is the time slot, the client sends a capacity request $G'(r_i)$ to the server. This request $G'(r_i)$ is based on the system state up to that point. At time $g_i = r_i + d$, with $0 < d \leq U$ being the round trip time for a request-grant application, a grant $G(r_i) \in \left[\check{R}, \hat{R}\right]$ is received, and the new service rate is applied, during the interval $[r_i + d, r_i + d + U]$. Figure 11.1 shows a summary of above description.

In some cases, geostationary satellites have a $U = 1s$ and $d = 250$ ms, while for some implementations of G.fast $U = d$ can be in the order of 5s.

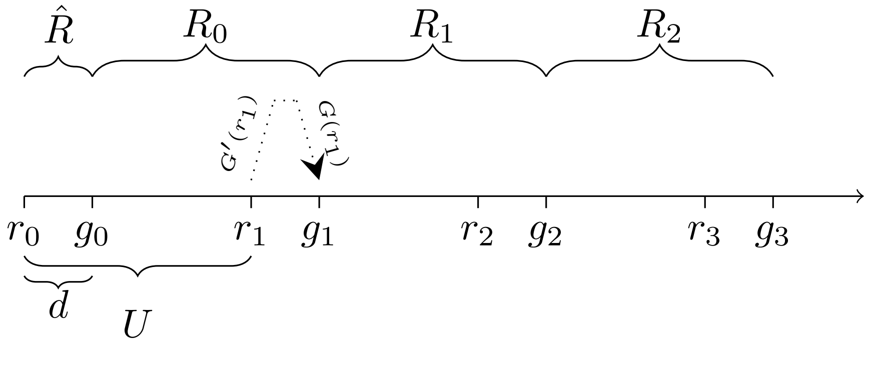


Figure 11.1: Request - grant timeline

We impose the QoS restriction $P\{D(k) > \hat{T}\} \leq \varepsilon$ where D is the delay distribution of the traffic aggregate, $\hat{T}$ the upper bound on the delay and ε the maximal percentage of packets that can have a delay larger than $\hat{T}$.

The number of bits in the queue and the HOL at time t are defined as $q[t]$ and $\Gamma[t]$, respectively. The total number of arrivals and departures in an interval $[s,t]$ are denoted by $A\,[s,t]$ and $E\,[s,t]$. To count packets instead of bits we notate them as $a\,[s,t]$ and $e\,[s,t]$.

The symbols are summarized in Table 11.1 (and extend those defined in the symbols table on Page 5).

Table 11.1: Symbols

Symbol	Meaning
U	system slot size
d	round trip time for a request-grant application
τ	slot size for the measurement based admission control (MBAC) algorithm
T	number of slots in a measurement window
C_k^m	the peak aggregate rate in the m-th most recent measurement window
M	number of measurement windows
ρ_k	smoothing factor for the EMA
K	the set of slots for which we track peak rates
d_p	propagation delay

11.3 Algorithms

11.3.1 The *Simple* algorithm

The pseudocode for the ad hoc *Simple* algorithm is shown in Algorithm 11.1. It captures the arrivals on four timescales (function M on line 2), ranging from U to $\frac{U}{20}$, assuming that more recent arrivals have a larger impact on the arrivals in the future slot. Additionally, in function MQ on line 3 the queue size is taken into account to ensure that the current queue will be emptied within $\hat{T}$ seconds. To ensure that small outliers in the last window do not result in a low rate, we take the maximum over the past three rates (line 9).

To account for variation in traffic over longer time scales, we also track the prediction error in the Err object, which is an exponentially moving average. We set the smoothing factor $\rho = 1-10^{-\varepsilon/0.5}$, where $\varepsilon/0.5$ is the time constant, indicating after how many system slots 10% of the original signal remains. The call to Err.collect will record a new value, as described by equations (11.10) and (11.11) (line 6). We assume a normal distributed prediction error, and multiply by the $1-\varepsilon$ quantile, if it is larger than 1 (line 7, 10).

Algorithm 1: The ad hoc Simple algorithm

```
1   Constant: $U$, $\hat{T}$, $\varepsilon$
2   func M($A$, $s$) = $\max_{i\in\{1,5,10,20\}}\{\frac{A(s-U/i,s)}{U/i}\}$
3   func Mq($A, q, s, \alpha = 1$) = $\max\{\frac{q[t]}{\hat{T}}, M(A,s)\cdot\alpha\}$
4
5   func Simple($A$, $q$, $V$, Err) =
6       Err.collect( $\frac{M[t]}{M[t-1])}$ )
7       $\alpha := normalQuantile(Err.\mu, Err.\sigma, 1-\varepsilon)$
8       return max{
9           M($A, t-2U$), M($A, t-1U$), M($A, t$),
10          Mq($A, q, t, \max\{1, \alpha\}$)
11          }
```

The complexity of this algorithm is very low. For storage we require $N+M$ numbers, where $N = 4$ is the number of intervals we track in the current slot, and $M = 3$ the past assigned rates.

Calculating the rate requires taking the maximum of $M+N$ values, resulting also in a very low computational complexity.

11.3.2 The *MBAC* algorithm

The algorithm is based on an AC algorithm to update the channel capacity. As an AC algorithm assesses if adding a new flow to a channel with capacity R will not violate any requirements, it needs to have a prediction ability of the current traffic aggregate.

Assume a function $\mathrm{AC}(R, \eta, f, \hat{T}, \varepsilon)$ that returns true if a capacity R and traffic aggregate η can support a new flow f without violating the delay constraint $P\{D > \hat{T}\} \leq \varepsilon$, and that returns false otherwise. Then, calling $\mathrm{AC}(R, \eta, f_0, \hat{T}, \varepsilon)$, where f_0 is a flow with no traffic at all returns true if the current aggregate will not violate the requirements for the current service rate. Obtaining the optimal $R^*[t+1]$ is then equal to solving

$$R^*[t+1] = \arg\min_{r\in[\check{R},\hat{R}]}\{\mathrm{AC}(r, \eta, f_0, \hat{T}, \varepsilon)\}.$$

The MBAC algorithm described in [150] provides such a function AC. It considers the average and variation of the peak rate of the traffic aggregate envelopes over multiple interval lengths. This allows us to determine in what interval a violation is most likely to occur, and dimension our system accordingly. In this paper, we base our MBAC algorithm on this admission control algorithm, but it should be noted that any MBAC algorithm can be used for this approach. The authors of [35] mention that a large family of admission control algorithms, to which [150] also belongs, have a similar performance. Hence, the same approach can probably be applied to these algorithms, with some tweaks.

11.3.2.1 The original MBAC algorithm

The original algorithm is described here for reference. Time is divided into slots of size τ. It should be noted that τ is not correlated with U, which we will refer to as the system slot size to avoid confusion (though, as we will see later, ideally U is an integer multiple of τ). C_k^m is defined as the peak aggregate rate in an interval of $k\tau$ in the m-th most recent measurement window of size T slots.

Thus,

$$C_k^1 = \frac{1}{k\tau} \cdot \max_{t-T\tau \leq s \leq t-k\tau} A[s, s+k\tau] \tag{11.1}$$

is the peak aggregate rate of the $k\tau$-sized window in the last T slots.

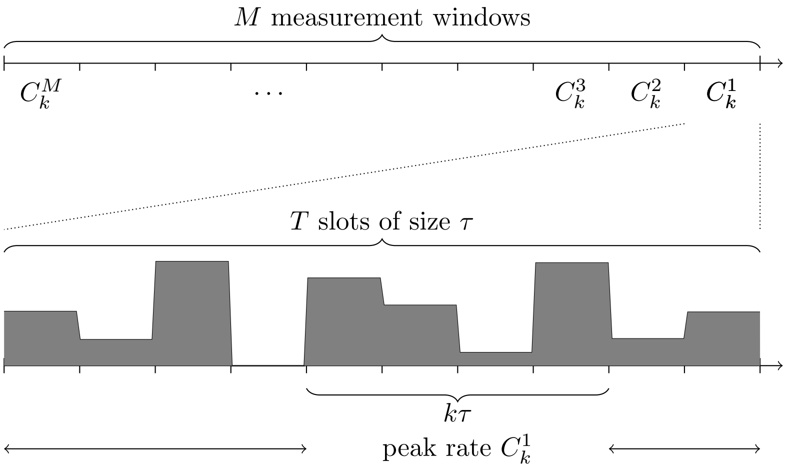


Figure 11.2: Overview of the original MBAC model

From these observations, we calculate the empirical mean and variance respectively as

$$\bar{C}_k = \sum_{m=1}^{M} \frac{C_k^m}{M} \tag{11.2}$$

$$\sigma_k^2 = \frac{1}{M-1} \sum_{m=1}^{M} (C_k^m - \bar{C}_k)^2 \tag{11.3}$$

Assuming stationary arrivals, the aggregate flow satisfies

$$P\left\{\max_s \frac{A[s,s+k\tau]}{k\tau} \leq \bar{C}_k+\alpha\sigma_k\right\} = \Phi(\alpha) \tag{11.4}$$

Thus the peak rate over an interval of k slots is less than $\bar{C}_k+\alpha\sigma_k$ with probability $\Phi(\alpha)$. Denoting the distribution of peak rates C_k by $F_k(\cdot)$, we can approximate $F_k(\cdot)$ by the Gumbel distribution

$$P\{F_k \leq x\} = \exp\left[-\exp\left(-\frac{x-\mu}{\beta}\right)\right]. \tag{11.5}$$

The Gumbel distribution is used in extreme value theory where it describes the asymptotic distribution of the extremes of a series of random variables, and is applicable to a large class of distributions, including Gaussian, Gamma, exponential, log-normal and Gumbel distributions [41]. The distribution has two parameters, the mean μ and the scale β.

The mean and variance of the Gumbel distribution are respectively $\mu+\beta\gamma$ and $\pi^2\beta^2/6$ (where γ is the Euler-Mascheroni constant). As we have measured the mean $\bar{C}_k$ and variance σ_k^2, we can infer the μ and β parameters for the Gumbel distribution. Now, we can approximate

$$\Phi(\alpha) \approx \exp\left[-\exp\left(-\frac{(\bar{C}_k+\alpha\sigma_k)-\mu}{\beta}\right)\right]. \tag{11.6}$$

From this $\alpha = -\frac{\sqrt{6}}{\pi}(\ln(-\ln(\varepsilon))+\gamma)$ can be obtained, for the required confidence level ε. The minimal $R \in \left[\check{R}, \hat{R}\right]$ for which the conditions (11.7)-(11.8) hold, is then a service rate satisfying our delay requirement, with confidence level $\Phi(\alpha)$.

$$\forall k : k\tau(\bar{C}_k+\alpha\sigma_k-R) \leq R\cdot\hat{T} \tag{11.7}$$

$$\bar{C}_T+\alpha\sigma_T \leq R \tag{11.8}$$

The first condition ensures that the peak rate in any $k\tau$ sized-interval will not lead to delays larger than $\hat{T}$. The k for which this equation does not hold indicate the dominant time scales under which delay bound violations are most likely to occur. The stability condition (11.8) ensures that the mean rate over a window is less than R, and the busy period is less than T, with confidence level $\Phi(\alpha)$.

11.3.2.2 The new algorithm

The above algorithm is designed to accommodate new flows into a long-running system, and guarantee long-term stability. As we have a dynamic system, we modify the above algorithm in two places. First, we expand the conditions

(11.7)-(11.8) with condition (11.9), with which we include the queue size and delay build-up into calculating the required service rate:

$$q[t]+(\Gamma[t]-d)\cdot R[t]+R^1_{d/\tau} < R\cdot\hat{T} \tag{11.9}$$

It states that the estimated queue at time $t+d$ must be serviced away in a timely manner. In an admission control algorithm, it does not make sense to use the queue, as it is a volatile metric. A moment after the admission has been performed, the queue size might have changed dramatically. In our system, updates are performed frequently, and as such the queue size provides important information about QoS. The ideal R is now the minimal rate that satisfies conditions (11.7)-(11.9).

The second modification covers the calculation of the average and variation of the peak rates. The original AC takes the long-term average into account, discarding timing information. As the timing of the occurrence of peak rates is more important when the service rate changes more frequently, this is not an ideal metric to use. Therefore, we use an exponentially moving empirical mean and variance, rather than the moving empirical mean and variance of the peak rates, in which we can choose the importance of older observations. Hence, for each $k \in K = \{1, 2, \ldots, T\}$ we calculate the mean and variance as

$$\bar{C}_k = \rho_k C^1_k+(1-\rho_k)\bar{C}_k, \tag{11.10}$$

and

$$\sigma^2_k = \bar{\bar{C}}_k-\bar{C}^2_k, \tag{11.11}$$

where we use $Var(X) = E[X^2]-E[X]^2$ to calculate the variance, and

$$\bar{\bar{C}}_k = \rho_k (C^1_k)^2+(1-\rho_k)\bar{\bar{C}}_k. \tag{11.12}$$

The simulations use a numerically stable version [67], because the version listed here suffers from catastrophic cancellation. The parameter $\rho_k \in]0, 1[$ is the smoothing factor of the EMA.

The parameters that are used in this new algorithm are listed here:

τ determines the minimal interval over which we calculate peak rates. A smaller τ, captures the peak rates better. Ideally, this is set to the smallest inter-arrival time that could occur. A smaller τ would not capture any more information. However, making τ smaller, implies increasing T. As we will see in Section 11.3.2.4, this will increase computational complexity.

T is the maximum number of consecutive τ-sized slots we track in calculating peak rates. In the original MBAC algorithm this variable can be chosen freely. In our algorithm it is more interesting to ensure that $T = U/\tau$, meaning that the number of slots we track equals the length of the system slot U.

ρ_k is a relative of the original MBAC's M parameter, the number of past peak rates to track. This parameter determines the smoothing factor of the EMA. A value close to 1 places more emphasis on recent values, i.e. it will react faster to changes. A ρ_k close to 0 will emphasize older values. This increases the variance, and results in a more conservative rate prediction. Choosing the optimal ρ_k is impossible, as it depends on the traffic, the delay upper bound $\hat{T}$, the interval T etc. Therefore, in the next section we introduce an addition to the algorithm that dynamically updates ρ_k.

11.3.2.3 Dynamic ρ_k

The following addition dynamically adapts ρ_k using the delay performance. The k for which the left side of (11.7) is the largest is the timescale on which delay violations are most likely to occur. Denote the cumulative distribution function of the delay distribution up until time t by $F_{D_t}(x) = P\{D_t \leq x\}$. Then $F_{D_t}(\hat{T})$ is the QoS violation probability, which should be less than ε. If $F_{D_t}(\hat{T}) > \varepsilon$ and increasing, then we assume ρ_k was not adequate and we decrease ρ_k, as implemented in Algorithm 11.2 on lines 3-4. The closer we are to ρ_{min} the slower the decrease. We only update if $F_{D_t}(\hat{T})$ is increasing with respect to the previous system slot, to avoid that a single burst would cause ρ_k to continue decreasing. Lines 5-6 describe the case where $F_{D_t}(\hat{T}) < 0.9\varepsilon$. In this situation we consider the algorithm being too conservative, and we increase ρ_k very slowly. Modifying ρ_k is not retroactive, therefore the ρ_k parameter is updated slowly to avoid overshooting its target.

Algorithm 2: Updating ρ_k

```
1  constant: $U, \hat{T}, \rho_{min}, \rho_{max}$
2  func SetRho($d$, $\rho_k$) =
3      if $F_{D_t}(\hat{T}) > \varepsilon$ and $F_{D_t}(\hat{T}) > F_{D_{t-1}}(\hat{T})$:
4          $\rho_k := \rho_k + (\rho_{min} - \rho_k)/10$
5      elif $F_{D_t}(\hat{T}) < 0.9\varepsilon$:
6          $\rho_k := +(\rho_{max} - \rho_k)/100$
```

Rather than measuring the cumulative delay distribution $F_{D_t}(\hat{T})$, we keep track of the number of packets whose delay exceeds $\hat{T}$, notated with $e_t(\hat{T})$. As such, we can calculate $F_{D_t}(\hat{T}) = \frac{e_t(\hat{T})}{a[0,t]}$.

11.3.2.4 Complexity

There are two complexities to consider here: the space complexity, and time complexity when updating and searching for a suitable rate, which happens every U seconds.

11.3.2.4.1 Space complexity We need to track $|K|$ values of $\bar{C}_k$, $\bar{\bar{C}}_k$ and ρ_k. Furthermore, we require the arrivals in the past T slots, resulting in a space complexity of $\mathcal{O}(|K|+T)$. An additional advantage of using our new method is that this comprises a big reduction in complexity, compared to the original algorithm's $\mathcal{O}(TM \ln(T))$ due to its need to track $R_k^m, m \in [1, M]$.

11.3.2.4.2 Time complexity Every U seconds the parameters are updated, and a suitable R is searched for. Setting a new ρ_k is a constant operation. Updating the parameters $\bar{C}_k$ and $\bar{\bar{C}}_k$ involves taking the maximum over the sum of an interval of k slots, for every $k \in K$. This can be done in $\mathcal{O}(|K|T)$. Selecting the optimal service rate, requires solving $|K|+2$ linear equations, giving a final time complexity $\mathcal{O}(|K|(T+1))$.

Thus, a suitable selection of K (or τ) can reduce time complexity, especially as T grows: selecting half of the elements of K reduces computation by half. This reduction, however, comes at the cost of a decrease in prediction accuracy and thus an increase of the delay error.

Figure 11.3 shows the behavior of the delay error in the DSL setting (Figure 11.3a) and the average time to calculate a suitable rate (Figure 11.3b), as a function of $|K|$. In the results below, the different $k \in K$ are evenly distributed between $[1, T]$. For example, $|K| = 3$ means $K = \{1, T/2, T\}$

Increasing $|K|$ thus expands the number of measurements, resulting in a better delay error metric, as can be seen in Figure 11.3a. Very small values of $|K|$ perform badly, as the dominant time interval over which violations happen is not captured correctly. It is not shown here, but making $|K|$ larger decreases the efficiency, but only very slightly: changing from $|K| = 1$ to $|K| = 100$ results in 2% less efficiency.

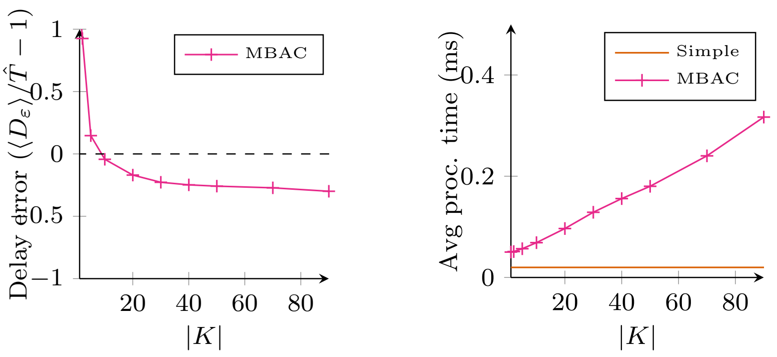
The trade-off involved with $|K|$ is shown in Figure 11.3b: the processing time grows linearly with $|K|$. For reference, we also displayed the average processing time of the *Simple* algorithm as a horizontal line.

In the remainder of this section, we use $|K| = 40$, which is a nice trade-off between delay error and processing time.

11.4 Performance evaluation

11.4.1 Simulation setup

The simulations were run using the INET and OMNeT++ frameworks. A client continuously sends traffic to a server over an abstract, lossless channel with a propagation delay of d_p seconds. Every U seconds the client sends a request over the channel, the reply of which arrives $2d_p < d \leq U$ seconds later, thus assuming



(a) $|K|$ vs delay error (b) $|K|$ vs processing time

Figure 11.3: $|K|$ vs delay error and processing time (DSL setting)

a processing time of $d - 2d_p$ at the server. A fluid system is simulated, implying that any change in capacity is immediate. For example, a packet that is being transmitted while the channel's capacity is lowered, will have a longer transmission delay.

The delay of a packet is measured as the difference in time between a packet arriving at the client, and its last bit arriving at the server, minus the propagation delay d_p. During the warm-up period of 30s no delays were recorded, and the service rate was kept constant to allow TCP streams to settle.

We ran the scenarios for the parameters listed in Table 11.2. The traffic sources are described in Table 11.3. Each scenario was run for 10 minutes, and repeated 20 times, the average of which was used.

Table 11.2: Simulation parameters

Parameter	Values
Protocol	UDP, TCP
Traffic	Poisson, Video, VoIP, Mix, Self-Similar, Sine
(U, d, d_p)	Sat (1s, 240 ms, 120 ms), DSL (5s, 5s, 0 ms)
T	U/τ
$\hat{T}$	100 ms, 500 ms, 1s
ε	0.1%, 1%, 5%, 10%
τ	10 ms

Table 11.3: Traffic sources

Traffic	Contents
Poisson	Poisson traffic models (12)
video	video traces, including Starwars [156] (12)
VoIP	on-off model [37] (30)
Self-Similar	superposition of Pareto [179] (12)
Mix	A mix of all the above traffic sources (17)
Sine	A mix of Self-Similar traffic and sources that send oscillating traffic at various frequencies (9)

11.4.2 Comparison algorithms

To assess the performance of the algorithms described above, we introduce the oracle bandwidth predictor, and a traffic prediction algorithm from the literature designed for stochastic resource allocation.

11.4.2.1 Oracle

The major hurdle in adequate resource allocation is the fact that the arrivals in the next window are not known. Prediction techniques try to alleviate this. Failure to predict sudden peaks, however, usually result in an increased load and subsequent increase of delays. The oracle algorithm we present here has access to the traffic arriving in the future $U+d$ seconds. This (simplified) algorithm is displayed in Algorithm 11.3. It simulates the queue evolution (function starting at line 23), which contains the timestamp and size of packets, and calculates the corresponding delay distribution D_{t+1}. By performing a binary search-like operation (lines 8-20), the smallest R_{t+1} is found, for which the predicted delay distribution does not violate the QoS requirements (line 13). If the delay is violated using the maximal capacity $\hat{R}$ (pirOK becomes false, line 5), then we try to find the smallest capacity that does not increase the delay percentile (lines 15-17).

As the TCP protocol is quite complex, and more difficult to simulate, due to its statefullness and mechanisms such as acknowledgments, window scaling etc., we only use the *Oracle* algorithm for UDP.

Algorithm 1: The Oracle algorithm

```
Constants: $\hat{R}$, $\check{R}$, $d$, $\hat{T}$, $\varepsilon$
func Oracle($D_t$, Q, $\rho$) =
  dPIR := simulate($D_t$, Q, $\rho$, $t+d$, $\hat{R}$)
  bestDelay := cdf(dPIR, $\varepsilon$)
  pirOK := bestDelay $\leq \hat{T}$
  lo := $\check{R}$; hi := $\hat{R}$; bestRate := $\hat{R}$

  for i in 1 .. 20:
    mid := (lo+hi)/2
    dMid := simulate($D_t$, Q, $\rho$, $t+d$, mid)

    if pirOK:
      OK := cdf(dMid, $\varepsilon$) $\leq \hat{T}$
    else:
      OK := (cdf(dMid, $\varepsilon$) $\leq$ bestDelay) and \newline
        ($\max_{\mathrm{dMID}} \leq \max_{\mathrm{dPIR}} * 1.01$)
      bestDelay := min(bestDelay, dMid)

    if OK: hi := mid; bestRate := mid
    else: lo := mid
  return bestRate

func simulate($D_t$, Q, cur$-\rho$, tChange, new$-\rho$) =
  simulate packets from Q, updating $D_{t+1}$.
  At time tChange the service rate changes to new$-\rho$.
  return $D_{t+1}$
```

11.4.2.2 SDB

In [70], the authors use a NLMS predictor on the traffic arrivals, and apply the Chebyshev inequality to stochastically bound future arrivals. A NLMS of order p is a linear predictor with weights $\boldsymbol{h}$. In such a linear predictor, a prediction is made by

$$\tilde{x}[(t+1)] = \sum_{i=0}^{p} x(t-i) \cdot h_t(i) \tag{11.13}$$

The weights are updated according to

$$\boldsymbol{h}_{t+1} = \boldsymbol{h}_t + \mu \frac{\epsilon(t)\boldsymbol{x}_t}{||\boldsymbol{x}_t||^2} \tag{11.14}$$

Here $\mu \in]0,2[$ is the rate of convergence, and $\epsilon(t)$ is the prediction error.

The Chebyshev inequality states that for a random variable X with mean μ_X and variance σ_X^2holds that

$$P\{|X-\mu_X| \geq k\sigma_X\} \leq \frac{1}{k^2}. \tag{11.15}$$

Hence, by changing k we can choose the violation probability. The requested rate is then calculated as

$$R = \frac{\tilde{x}(t+1) + \sqrt{1/(2\varepsilon)}\hat{\sigma}_\epsilon}{\hat{T}}, \tag{11.16}$$

with

$$\hat{\sigma}_\epsilon(t) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\epsilon^2(t-i)} \tag{11.17}$$

the sample variance of the error.

11.4.3 Results

We begin this section by a short introduction to the plot layout, followed by a comparison of the different algorithms for the UDP and TCP scenarios in respectively Section 11.4.3.2 and Section 11.4.3.7.

The plots described in this section follow the same pattern: on the x-axis we display the average over all selected simulation runs of $\langle D_\varepsilon \rangle / \hat{T} - 1$, where $\langle D_\varepsilon \rangle$ is the average over all simulations of the ε percentile of the delay distribution. A value in the range $[-1, 0]$ means that ε percent of the packets have a delay less than $\hat{T}$. A value larger than 0 indicates that too many packets (i.e. more than ε percent) had the delay constraint $\hat{T}$ violated. For example, a value 1 means that ε percent of the packets have a delay that is more than $2\hat{T}$. A dashed vertical line is drawn at 0 to indicate this cut-off value.

On the y-axis we show the average efficiency, calculated as $100 \cdot E(0, T_{sim}) / \sum_{t=0}^{T_{sim}} R[t])$. The closer the efficiency is to 100%, the better.

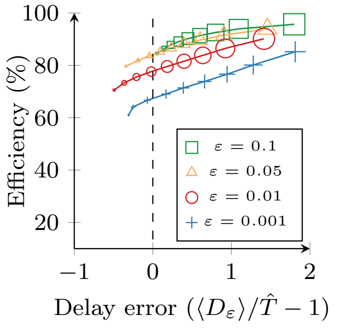
Each algorithm has four data points, one for each $\varepsilon \in \{0.1\%, 1\%, 5\%, 10\%\}$, sized accordingly, and connected for easier recognition. Ideally, this curve should be vertical, as then it means that we can maintain the same delay for various ε.

Simulation results have also been run for $\hat{T} = 500$ ms, but have been left out as they were in general behaving as expected. Any different behavior is indicated in the text.

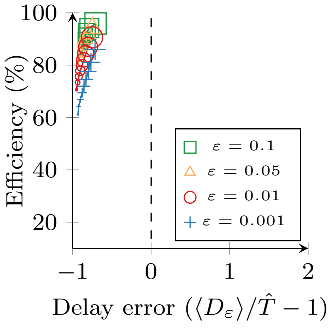
11.4.3.1 Fixed ρ_k

In this subsection we discuss how the results change in function of a constant ρ_k. The plot of Figure 11.4 shows the averages of the delay error versus the efficiency over all simulation runs. Each trace represents a different ε, while the individual data points are different $\rho \in [0.1, 0.9]$, the largest ρ having the largest symbol, on the right-hand side.

The trend is that ρ incurs a trade-off between the delay error and the efficiency: decreasing ρ, i.e. favoring older arrivals, results in decreased delays, but at the cost of efficiency. A small ρ removes much of the high frequency noise, which can attribute to excessive delays. This behavior is visible in all different settings.



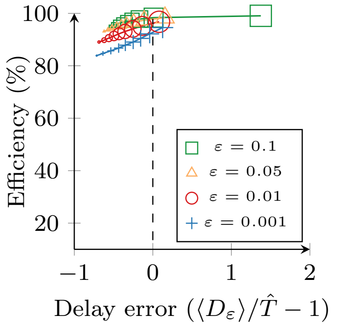
(a) $\hat{T} = 0.1s$



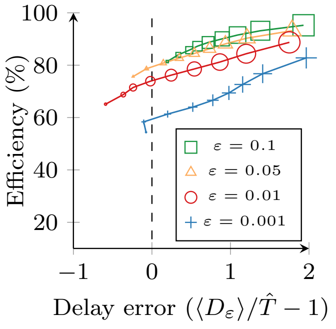
(b) $\hat{T} = 1s$

Figure 11.4: vs efficiency for varying ρ and ε

Another thing to notice is that the difference with respect to the different ε for efficiency and delay error is quite constant. This means that the measurement based approach is able to take the required delay violation quite well into account.



(a) Poisson



(b) Mix

Figure 11.5: Delay error vs efficiency for varying ρ and ε Poisson vs Mix for $\hat{T} = 0.1s$

However, Figure 11.4a is slightly misleading because there are quite some differences in the delay error metric between different traffic types. For example, in Figure 11.5a and Figure 11.5b we can see the same plot, but now highlighting the traffic types Poisson and Mix, respectively. There it clearly shows that for predictably behaving traffic, such as Poisson, the choice of ρ is less critical, as the domain is much smaller. The outlier in Figure 11.5a is for $\rho = 0.9$ and $\varepsilon = 0.1$ where the algorithm underestimated on various occasions the traffic intensity, as the Gumbel distribution is not ideal for well-behaved traffic such as Poisson. The bursty traffic of Figure 11.5b shows that a smaller ρ is necessary. The plots of the metrics, however, behave very similar.

Returning to Figure 11.4, we can see that for a small $\hat{T}$ (Figure 11.4a) there are many delay violations, while for larger $\hat{T}$ (Figure 11.4b) the opposite is true. The reasoning behind this decrease is as follows: in [21] it is shown that for $GI/G/1$ queues $P\{D > \hat{T}\} \leq \varepsilon$ can be approximated by $P\{D > \hat{T}\} \approx \exp\{-\frac{\hat{T}}{E[D]}\}$. Here, we can see that if the nominator $\hat{T}$ increases faster than the denominator $E[D]$, the delay violations will decrease. Using Kingman's formula [101], we can approximate

$$\begin{aligned} E[D] &\approx E[D_{M/M/1}] \cdot \frac{c_a^2 + c_s^2}{2} \\ &= \frac{1}{\mu(\mu/\lambda - 1)} \cdot \frac{c_a^2 + c_s^2}{2} \end{aligned}$$

Here λ is the mean arrival rate, μ the mean service rate and c_a and c_s the coefficients of variation for the arrival and service times respectively.

The average rate μ is dependent on $R_k(\hat{T}) = (\bar{C}_k + \alpha\sigma_k)\frac{k\tau}{\hat{T}+k\tau}$, obtained after rewriting (11.7). If we assume for a moment that we always pick the same k, then μ is only influenced by $\hat{T}$, since in our simulations $\bar{C}_k$ and σ_k are identical for different parameters. Thus, we can write, with $\mu' = E[\bar{C}_k[t]) + \alpha\sigma_k[t])]$:

$$\begin{aligned} \frac{\hat{T}}{E[D]} &= \frac{\hat{T}}{\frac{1}{\mu(\mu/\lambda-1)} \cdot \frac{c_a^2+c_s^2}{2}} \\ &= \frac{2\hat{T}\mu(\mu/\lambda - 1)}{c_a^2 + c_s^2} \\ &= \frac{2\hat{T}\mu' \frac{k\tau}{\hat{T}+k\tau}(\mu' \frac{k\tau}{\hat{T}+k\tau}/\lambda - 1)}{c_a^2 + c_s^2} \end{aligned} \tag{11.18}$$

To remove the dependence of c_s on the actual rates, we can upper bound $c_s \leq \frac{L-U}{2\mu}$, where $L = \min_s\{R[s]\}$ and $U = \max_s\{R[s]\}$. This upper bound only occurs in the extreme case where in one slot R reaches the maximum U, and in all other slots R equals the minimal rate L. Over all the simulation runs we found that c_s is bounded above by 2.

We plotted Equation (11.18) for particular values in Figure 11.6. Here we can observe that $\hat{T}/E[D]$ keeps rising, until $\hat{T}$ becomes $\hat{T}_M = \frac{k(\mu'-\lambda)}{\mu'+\lambda}$, after which the ratio will go down and become 0 at $\hat{T} = \hat{T}_0 = k(\mu'/\lambda - 1)$. Thus, increasing $\hat{T}$ in the range $\left[0, \hat{T}_M\right]$ will result in a delay decrease. For $\hat{T} > \hat{T}_M$ the delay will increase. As $\hat{T}$ exceeds $\hat{T}_0$ the delay increases exponentially because the average service rate μ will become less than the average arrival rate λ, resulting in a load greater than 1. Equation (11.8) prevents this from happening by putting a lower bound on the service rate.

A large $\hat{T}$ implies a larger efficiency: as mentioned before, $R_k(\cdot) \sim \frac{k\tau}{\hat{T}+k\tau}$. We can observe that $R_k(\hat{T})$ decreases as $\hat{T}$ increases due to the factor $\frac{k\tau}{\hat{T}+k\tau}$. The final

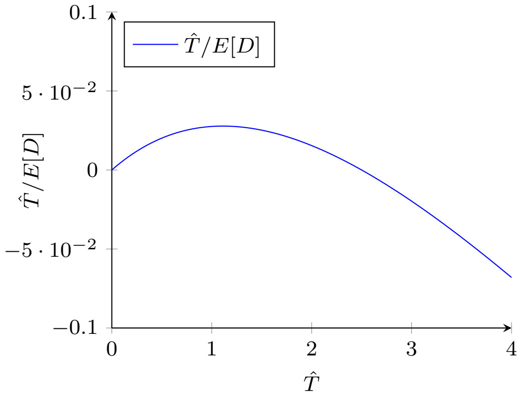


Figure 11.6: $\hat{T}/E[D]$, for $\mu' = 0.5$, $\lambda = 0.4$, $c_a^2 = c_s^2 = 1.5^2$, $k\tau = 10$

$R = \arg\max_{k \in K} R_k(\hat{T})$, thus also decreases. As the number of arrivals remain the same for our scenarios, this means the efficiency goes up. In the simulations this effect is very small though, being on average about 1% more efficient.

The remainder of the simulations use the dynamical ρ_k.

11.4.3.2 UDP comparison

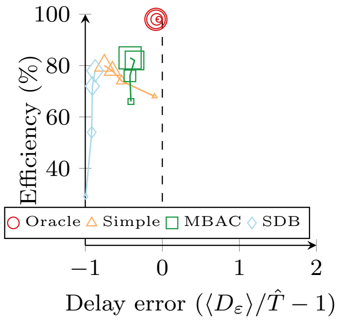
In this section we compare the different algorithms for the UDP protocol in the Figures 11.7 and 11.8.

11.4.3.3 Oracle

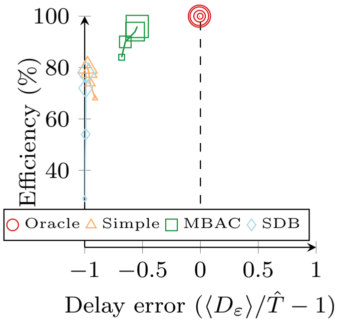
The *Oracle* algorithm (red circle) has a close to optimal behavior, as can be observed in Figure 11.7 and Figure 11.8: the delay error is close to zero and the efficiency near 100% for all ε. It is only for DSL and a small $\hat{T}$ that the efficiency is a bit lower. This is explained by the fact that the *Oracle* algorithm has a look-ahead of only $U+d$ seconds. If a lot of traffic arrives closely together in time, then the service rate is adapted to accommodate for these peak arrivals as it wants to maintain a good delay error at all times. If this peak is short-lived and large compared to the rest of the slot, then a lot of capacity remains unused. This is apparent for $\varepsilon = 10^{-3}$: there some traffic types, such as Mix, have an efficiency of only 70%.

11.4.3.4 SDB

The predictive approach of the SDB algorithm (blue diamond) is able to maintain a very low delay error. For large ε the efficiency is on par with the

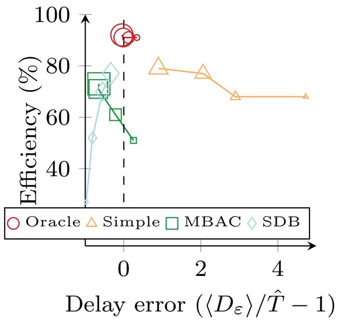


(a) $\hat{T} = 0.1s$

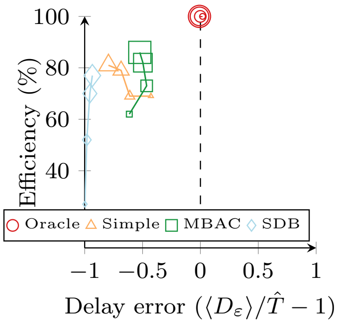


(b) $\hat{T} = 1s$

Figure 11.7: Summary for UDP for the Sat scenario



(a) $\hat{T} = 0.1s$



(b) $\hat{T} = 1s$

Figure 11.8: Summary for UDP for the DSL scenario

other schedulers. However, when we require a low delay violation, the efficiency plummets. This is mainly due to the prediction error, which raises the variance quickly causing the algorithm to be too prudent.

11.4.3.5 Simple

The *Simple* scheduler (orange triangle) is in general able to satisfy the QoS requirements. The efficiency is relatively good, but sometimes at the cost of delays.

It is only for the most difficult scenario tested here, a large system window and small allowed delay in Figure 11.8a, that the average delay error is quite large. This seems sensible, as picking a wrong service rate will have a much larger impact, since it cannot be changed for a longer time. The biggest contributor to the large deviation is the Mix traffic type: none of the simulation runs have a satisfying delay error.

The average efficiency is about 80% for large ε to 70% for small ε, performing about 20% to 30% worse than the *Oracle*. If one looks at the underlying data, then one can see the average is a good estimator of the individual performance.

11.4.3.6 MBAC

The *MBAC* scheduler (green square) is able to remain below a zero delay error, while the efficiency in general is similar or better than *Simple* for large ε, and doing worse for small ε, except for $\hat{T} = 1s$ in the Satellite scenarios, where it performs much better.

Looking at the data, the average delay error is in general quite constant, with outliers caused by video traffic, which is inherently difficult to predict. As the interval between such scenes can be in the order of tens of seconds – enough to have been forgotten by the average $\bar{C}_k$ and variance σ_k – the *MBAC* algorithm sometimes has troubles maintaining the delay error. This is apparent in Figure 11.8a for a small ε.

The efficiency is good when the delay upper bound is large. For the Sat scenario, the efficiency is close to optimal for large ε.

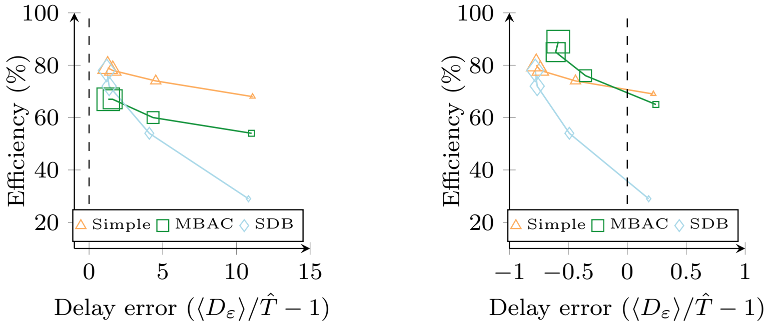
Looking at these plots, and underlying data, we can conclude that the *MBAC* algorithm has an overall better performance than the two other methods. It can deal quite well with larger system slot sizes, due to its better understanding of the flow's behavior on different time scales. Compared to the ideal *Oracle* algorithm, we can see, however, that the *MBAC*'s efficiency is not in the same range.

11.4.3.7 TCP comparison

In this section we run the simulations for TCP. TCP is, contrary to UDP, a connection oriented protocol that provides reliable, ordered and error-checked delivery of data. Additionally, it tries to determine an optimal service rate, such that packet loss and congestion in the network are minimal. However, on links with a large bandwidth-delay product, such as satellite links, these mechanisms deteriorate the offered service (e.g. [166, 46]).

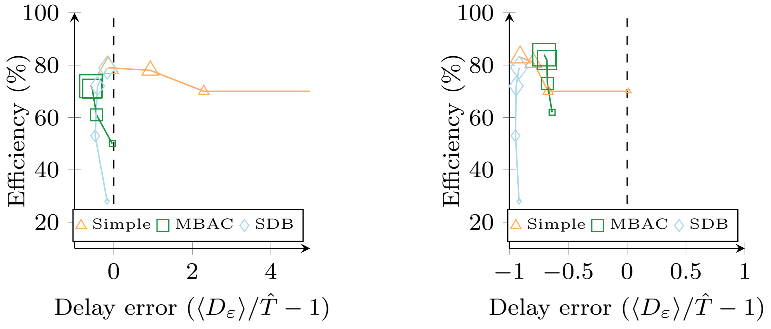
The main problem is that TCP was designed for wired networks, where the service is reliable, constant and low-latency. This assumption causes problems in satellite networks. For example, during the slow-start phase, large bursts of packets are sent over the network. The ensuing round-trip time of packets is very volatile, influencing the congestion control algorithm heavily.

For the simulations we used TCP Reno and tuned the TCP parameters for better performance [104, 105]. Delayed ACKs, selective ACKs and window scaling were enabled. For the Satellite scenario we also set the maximum segment size (MSS) to 9180 bytes, and increased the advertised window size to 100 MSS. In the simulations, each of the flows has its own TCP connection, rather than one TCP connection for the aggregate.



(a) $\hat{T} = 0.1s$ (b) $\hat{T} = 1s$

Figure 11.9: Summary for TCP for the Sat scenario



(a) $\hat{T} = 0.1s$ (b) $\hat{T} = 1s$

Figure 11.10: Summary for TCP for the DSL scenario

Looking at the TCP simulations, we can observe the average efficiency is similar to the UDP simulations, while the delay error differs. The most notable difference is the delay error in the Sat scenario for $\hat{T} = 100$ ms (Figure 11.10a): it is quite large for all algorithms. One reason for this is that the on-off model used to generate VoIP traffic, has an average OFF state of 640 ms, a relatively large period, causing TCP to initiate slow start [91].

Another reason is the large propagation delay in satellite networks. For example, the increased delay variances affect adversely the TCP timer mechanisms, resulting in premature timeouts and incorrect window sizes [14]. Another source of delay errors in this particular case can be attributed to TCP's congestion control. If we want to increase the service rate this is either due to our prediction, or due to the queue growing larger. A growing queue indicates larger delays, and a more volatile average round-trip (especially since we try to update the service rate every second), which leads to TCP decreasing its congestion window, thus reducing the service rate.

The DSL scenario in Figure 11.10 shows a similar performance to the UDP counterpart, both delay error- and efficiency-wise. The *Simple* scheduler, on the other hand, has problems allocating sufficient bandwidth when the delay upper bound is small, and also behaves more unpredictable for a larger delay upper bound.

11.5 Conclusion

In communication systems where traffic is variable, and unused capacity can carry a large cost, such as in satellite networks, it is important to be efficient with resources, while still adhering to QoS requirements. We presented in this section two algorithms that dynamically allocate capacity for a traffic aggregate. First, we discussed the system model in Section 11.2. The details of the ad hoc *Simple* and *MBAC* algorithms were described in Section 11.3. The simulation results were detailed in Section 11.4.3. There we compared our results to an ideal *Oracle* algorithm and one from the literature in a satellite and DSL setting, for both the UDP and TCP protocol and various QoS settings.

The *Simple* algorithm can be a bit unpredictable, and cannot deal with all QoS situations appropriately. Nonetheless, its simplicity can make it suitable in scenarios where a very low complexity is required. The *MBAC* scheduler is well-suited for most tasks and traffic types, and performs in general similar to or better than the two other algorithms. The efficiency depends heavily on the required violation probability, and is in the simulations worst-case 40% less than the ideal case.

Chapter 12

Throughput Constraining in Cross-layer Schedulers

12.1 Introduction

In the previous sections, we discussed cross-layer schedulers for DSL, LTE and 5G. These schedulers try to fairly distribute service rates over the users in a shared medium, by generating weights that are then used to solve the NUM problem

$$\mathbf{R}^* = \arg\max_{\mathbf{R}\in\mathcal{R}} \sum_{n=1}^{N} u^n(R^n, \mathscr{S}). \tag{12.1}$$

In this system, a user n's service rate depends on the weight in relation to all other users. This implies that it is challenging to control a single user's service rate, since there is no correspondence between one user's weight and its received rate. Nonetheless, sometimes we want to be able to constrain the short-term (and long-term) average rates, for example to ensure the QoS of the users. We give more reasons for implementing rate constraints in Section 12.1.1.

Most cross-layer schedulers, such as the ones listed in Section 9.6.1, Section 10.4.1 and Section 12.4.1 do not offer the option to confine the service rate. Therefore, after reviewing the system model in Section 12.2 we describe our contribution, the token bucket rate modifier (TBRM) algorithm, in Section 12.3. It is a generic low-complexity algorithm that constrains the short- and long-term average service rates of all users in a utility function-based cross-layer scheduler.

Each time slot t, the NUM problem is solved, but each user's weight is replaced by

$$\theta^n \exp(\frac{k_g^n}{\sigma_g^n} + \frac{k_M^n}{\sigma_M^n}).$$

Two counters, k_g^n and k_M^n, track the deficiency and excess in service, respectively. If a user n has received less service than ρ_g^n, the amount of tokens will accumulate, and the user's weight will increase, therefore raising the probability of receiving more data rate. The parameters σ_g^n and σ_M^n are a measure for the slowness to react to deficiency and excess respectively. The variable θ^n accounts for non-positive weights. The algorithm is very easy to incorporate into any scheduler, as it does not require manipulating the original scheduler's weight function.

In Section 12.4 we evaluate our algorithm through multiple simulations. We compare our results with the unbounded scenarios, and look at the influence of the slot size τ and parameter σ. The metrics indicate that we can bound the average service rate for all users within a limited amount of time. Schedulers whose weight fluctuate heavily are more difficult to constrain.

We close this section with related works in Section 12.5 and a conclusion in Section 12.6.

12.1.1 Motivation for service rate constraints

In a multi-user environment, it can be useful to ensure that average service rate of a flow is upper and/or lower bounded. For example, a provider might offer different QoS guarantees to different users, depending on the subscribed model. This might include a guaranteed and/or maximal rate.

There are other reasons to ensure a minimal rate. For example applications like audio and video need a minimal rate for a satisfying QoE. Additionally, the authors of [42] observe that TCP-based applications can lead to large queues when the throughput is too small. Finally, a guaranteed rate ensures that misbehaving competing flows, cannot smother flows from receiving their fair share.

Applying an upper bound on a user's service rate is also useful. For example, to accommodate a new flow into a network, admission control algorithms often require an upper bound on the arrival rates [150]. If a flow disrespects this rate, other applications in the network can suffer deteriorated QoS. By limiting the maximal data rate, a misbehaving flow is isolated and cannot negatively impact the other applications, but will only punish itself. In addition, it can also be useful to provide different service levels to users, where an operator may choose to cap the data rate for cheap data services, and remove this limit for the more expensive premium services.

Although rate constraints can be implemented into the physical layer, it might be interesting to handle it at higher layers. First, it reduces the degrees of cross-layer freedom, and limits the communication necessary. Second, this allows a more flexible approach, allowing for temporary violations. Finally, it might not always be possible to introduce the rate constraints into the physical layer. For example, the scheduler is implemented in hardware or closed-source and cannot be modified, or the corresponding NUM problem's complexity might increase too much due to the additional constraints.

Imposing upper bound constraints can be easily accomplished using a token bucket counter on a user's output stream. This is wasteful though: the token bucket would be applied to the service rate, as reserved by the cross-layer scheduler. Any excess service rate then remains unused by the user itself, but can neither be used by any of the other users, as would be the case when modifying the flows' weights in the scheduler.

12.2 System model

In this section we describe the conventions and symbols in Table 12.1 (in addition to those defined in the symbols table on Page 5) that will be used in the remainder of this section. Time in our model is divided into slots of size τ seconds. There are N users, indexed by $n \in [1, N]$, each of which can send $\tau \cdot R^n[t]$ bits during slot $t \in \mathbb{N}$, where $0 \leq R^n[t] \leq \hat{R^n}$ is the service rate for user n.

The service rates $\mathbf{R}[t] \in \mathcal{R}$ are determined by a scheduler, based on weights $\omega[t]$: at the start of slot t, a request is made to the scheduler, the reply of which is applied at the start of slot $t+1$. There is thus a delay of τ seconds between a request and application of the rates.

Table 12.1: Symbols

Symbol	Meaning
θ	weight modifier
x_g	value related to the guaranteed rate token bucket
x_M	value related to the maximum rate token bucket
k	number of tokens
ρ	rate of tokens
σ	slowness parameter

12.3 The Token Bucket Rate Modifier algorithm

12.3.1 Token buckets

Token buckets are applied in various situations. For example in [177, 54, 111] they are used to describe traffic flows (such as in Section 4.3 on Page 45, where we used it to model an aggregate of VoIP flows), while in [173] the authors employ token buckets to check conformance of incoming or outgoing traffic (policing and shaping), traffic marking in DiffServ [82, 81]. In [204] token buckets are used for rate estimation.

Conceptually, a token bucket $TB(\rho, \sigma)$ consists of a bucket holding k tokens (e.g. bits). Tokens are added at a constant rate ρ to the bucket, which is capped at σ tokens. Whenever a packet of L bit passes and there are sufficient tokens, L tokens are removed from the bucket and the packet continues its journey. If $k < L$, the packet is considered non-conforming and an appropriate action is taken, such as being color-marked non-conforming, queued (shaping) or dropped (policing). Such a token bucket will limit the long-term average outgoing rate to ρ. On a short-term scale, bursts of up to σ bits can be served.

12.3.2 Algorithm

In the following algorithm, this token bucket principle is used to lower and upper bound the service rate in a cross-layer scheduler setting. But, in contrast to a regular token bucket, we now do not cap the tokens to σ. Rather, they are used to indicate the severity of the excess. In the algorithm, instead of solving

$$\arg\max_{\mathbf{R}\in\mathcal{R}} \sum_n f(R^n)\omega^n \tag{12.2}$$

where $f(R) = R$ for the linear, and $f(R) = R^{-1}$ for the reciprocal variant, the NUM problem is modified to

$$\arg\max_{\mathbf{R}\in\mathcal{R}} \sum_n f(R^n)\theta^n \exp(\frac{k_g^n}{\sigma_g^n} + \frac{k_M^n}{\sigma_M^n}) \tag{12.3}$$

Here, $k_g^n \in [0, \infty[$ and $k_M^n \in]\text{-}\infty, 0]$ are the tokens for the guaranteed and maximal token buckets, respectively. The guaranteed token bucket for a user n handles the lower bound on the long-term average service a user receives, while the maximal token bucket manages the upper bound on the user's long-term average service rate. Every slot, the tokens are updated according to the following rules:

$$k_g^n(t+1) = \max\{0, k_g^n[t] + (\rho_g^n - R^n[t])\tau\} \tag{12.4}$$

$$k_M^n(t+1) = \min\{0, k_M^n[t] + (\rho_M^n - R^n[t])\tau\} \tag{12.5}$$

When the received service rate for a user n in the past slots is less than the guaranteed rate ρ_g^n, the virtual token counter k_g^n will continue to increase as long as there is a deficit in received service, and hence the weight will exponentially increase. Likewise, if a user n has received more than ρ_M^n service, the virtual token counter k_M^n will have a negative drift, as long as more data rate is assigned to the user. This will reduce the user's weight exponentially. When the service rate is less than ρ_M^n, the token counter will return to 0.

We introduce

$$\theta^n = \begin{cases} \overline{\omega}, & \text{if } \omega^n \leq \epsilon \text{ and } \frac{k_g^n}{\sigma_g^n} + \frac{k_M^n}{\sigma_M^n} \neq 0 \\ \omega^n, & \text{else} \end{cases} \tag{12.6}$$

to account for non-positive weights ω^n. Here $\overline{\omega}$ is the EMA of all the positive weights, and ϵ a small number, which would result in a user receiving a rate close to zero if the user's weight would be less than ϵ (for the simulations we used $\epsilon = \max_n\{\omega^n\}\cdot 10^{-5}$). If $\omega^n \in]0, \epsilon]$ it becomes difficult to increase the bandwidth reliably, and in the case of $\omega^n = 0$, it is even impossible, since the weight will remain zero.

For a negative weight, which can occur for example for best effort flows in the EXP/PF scheduler, multiplying by θ^n would result in an even lower weight, and would also inhibit us from receiving service. The variable $\overline{\omega}$ is used to approximate a valid weight that is reasonably stable. This weight is scheduler and traffic dependent, and thus must be calculated at run time.

Note that if $\rho_g^n = 0$ or $\rho_M^n \geq \hat{R}^n$, then respectively the first and second exponent will always be 1, and the respective bound is disabled.

It can be seen that if a flow stays within the bounds, then the tokens k_g^n and k_M^n will remain zero, and $\theta^n = \omega^n$, resulting in the unmodified weight. Only if some rate guarantee will not be met, weights will be adapted.

12.3.3 Discussion

12.3.3.1 Parameters ρ_g^n and ρ_M^n

The choice of ρ_g and ρ_M influences the speed at which the rate can adapt. For example, if the guaranteed rate $\rho_g^n = 0.75\hat{R}^n$, then in each slot the tokens can increase by at most $0.75\hat{R}^n$, and the negative drift is at most $0.25\hat{R}^n$. Thus, if this flow has been receiving no service, then the tokens - and thus the weight too - will increase quickly. If it is receiving service at a rate $\hat{R}^n$, then the tokens will decrease more slowly. A small ρ_g^n thus also implies a small positive and large negative drift. A similar reasoning can be applied to the maximal rate ρ_M^n.

12.3.3.2 Parameters σ_g^n and σ_M^n

In the traditional token bucket algorithms, σ is a measure for the burstiness of a flow. For example, large values of σ mean that large bursts are allowed. In our algorithm, the σ can be interpreted as a measure for slowness to react. A large value of σ_M^n means that longer periods of above-guaranteed service rates are possible, because our weight will decrease more slowly. Small values of σ_M^n will react quicker and can lead to an overreaction. The two token buckets can also influence each other: in case of an overreaction, the other token bucket will also have a sudden excess, and in turn have a fiercer reaction. This can be observed for small values of σ in the simulations of Section 12.4.2.2

12.3.3.3 Slot size

In our system, the slot size implies a delay between a request for and subsequent assignment of the service rate. A larger slot size means that changes will be slower, and that predicting future traffic becomes more important. This also matters to the rate constraint algorithm, since the scheduler's response to weights becomes more unpredictable, hence modifying the weights. The simulations of Section 12.4.2.3 briefly look at increasing slot sizes.

12.3.3.3.1 exp The function exp is chosen to modify the token fractions, but any continuous, strictly increasing function $\alpha(\cdot)$ for which holds that $\alpha(0) = 1$, $\lim\limits_{x\to-\infty} \alpha(x) = 0$ and $\lim\limits_{x\to\infty} \alpha(x) = \infty$ will give rate guarantees, albeit with different bounds. Tests with different functions resulted in more short-time erratic behavior.

12.3.3.4 Additive form

Instead of using a product, it is also possible to use an additive form,

$$\arg\max_{\mathbf{R}\in\mathcal{R}} \sum_n f(R^n)\omega^n + (\alpha(\frac{k_g^n}{\sigma_g^n}) + \alpha(\frac{k_M^n}{\sigma_M^n}))\beta$$

Here α is a continuous, strictly increasing function with the properties $\alpha(0) = 0$, $\lim\limits_{x\to-\infty} \alpha(x) = -\infty$ and $\lim\limits_{x\to\infty} \alpha(x) = \infty$. An additional factor β must be introduced to account for the fact that ω^n is usually not unitless.

We ran some simulations for $\alpha(x) = x$ and $\alpha(x) = x^3$, and $\beta = \overline{\omega}$. The simulations showed that this approach is also possible, and avoids the non-positive weight problem which forced us to introduce the factor θ^n. However, in the NUM problem, the relative weights are important, rather than the absolute difference, which the additive form expresses. Even though on larger timescales the additive form leads to nicely averaged data rates, on short timescales the behavior is very extreme, where $R^n[t]$ alternates between 0 and rates close to $\hat{R^n}$ in successive slots.

12.3.3.5 Complexity

The space and time complexity of the TBRM algorithm is very low. Every slot we update the N users' token counters k_g^n and k_M^n (Equations (12.4) and (12.5)). Additionally, we have to select a suitable θ^n for all n. The exponentially weighted $\overline{\omega}$ is a constant time operation $\mathcal{O}(1)$. The resulting time complexity is thus $\mathcal{O}(3N+1) = \mathcal{O}(N)$.
Likewise, the space requirements are equally low: we track the $2N$ counters, and a single EMA $\overline{\omega}$. The space complexity is in this case $\mathcal{O}(2N+1) = \mathcal{O}(N)$.

12.3.3.6 Other considerations

Applying rate guarantees transforms a work-conserving scheduler into a non-work conserving scheduler. I.e. the scheduler might have service assigned, even though there are no jobs available. For example, the MW scheduler is served based on its queue size. The larger the queue-rate product, the higher its service rate will be. It is clear that when the queue size is zero, without the algorithm would not receive service, but with minimal rate guarantees will have a weight larger than zero, and hence receive service.

Additionally, imposing throughput constraints reduces the stability region of a scheduler. Enforcing a maximal data rate inside the stability region, clearly decreases this region. However, supporting a minimal throughput constraint also modifies the stability region: a minimal throughput constraint can be rewritten as a (more complex) maximal throughput constraint on the other users.

12.4 Performance evaluation

12.4.1 Simulation setup

We evaluated the TBRM algorithm using simulations for the schedulers listed in Table 12.2. We ran simulations in OMNeT++ using the INET framework. Every $\tau = 50$ms the original weights and weight modifiers were computed. The resulting NUM problem was then solved with the help of the nlopt [172] library, by first applying the local variant of the DIviding RECTangles algorithm [69], followed by the COBYLA algorithm [149], to obtain the final rate, applying them in the next slot.

12.4.1.1 Scenarios

The scenarios listed in Table 12.3 show the different types of traffic and the applied constraints.

Table 12.2: Summary of the schedulers used in the simulation and their settings. Common symbols: $\overline{R}$ (averaged service rate), $\overline{\lambda}$ (averaged arrival rate), $\overline{\Gamma}$ is the average HOL of all real-time flows, and $\alpha = -\ln(\varepsilon)/\hat{T}$.

Scheduler	Real-time flow weight	Notes
MW [180]	$q[t]$	
M-LWDF [17]	$\frac{\alpha}{\overline{R}} \cdot \Gamma$	
EXP/PF [151]	$\exp\left(\frac{\alpha\Gamma - \overline{\Gamma}}{1+\sqrt{\overline{\Gamma}}}\right)\frac{1}{\overline{\rho}}$	
MDU [170]	$\overline{w}^{0.6} + \begin{cases} 0, \text{if } \overline{w} \leq \hat{T} \\ \overline{w} - \hat{T}, \text{otherwise} \end{cases}$	$\overline{w} = \overline{q}/\overline{\lambda}$
MD	$-q[t]$	reciprocal scheduler
MDV	$\tilde{\lambda}_i[t+1] \cdot f_c\left(\frac{q^n[t]}{\overline{r}_i[t]\hat{T}^n} + \frac{\Gamma^n[t]}{\hat{T}^n} + \ln_2\left(1 + \frac{p_i[t]}{\varepsilon^n}\right)\right)$	Reciprocal scheduler

The traffic types behave differently on short and large timescales. The first type of traffic consists of a sine-wave, superimposed with a faster oscillating sine-wave. Some flows will oscillate slowly (Sine2VS) while others oscillate fast (Sine2F). The second type of traffic is the heavy tail traffic, which is either a trace file of a video file, such as Starwars, or a self-similar flow, generated by a superposition of Pareto-distributed sources [25]. The last class of traffic, SAT, tries to send as much traffic as possible, by ensuring the queue is always backlogged.

These scenarios are run for $\tau = 0.05s$ in Section 12.4.2.1 In Section 12.4.2.2 we vary σ, and in Section 12.4.2.3 we vary τ.

12.4.1.2 Metrics

We examined three different metrics. The m2 and m3 metrics are defined on windows of size G, which groups G consecutive slots.

m1 The percentage of slots that would be marked non-conforming by a token bucket process $TB(\rho_g, \rho_g \tau x)$ and $TB(\rho_M, \rho_M \tau x)$ for respectively the guaranteed and maximal rate. $x \in \mathbb{R}^+$ is a variable indicating the allowed burstiness. Increasing x allows for more burstiness, and will result in a smaller percentage of non-conforming slots.

m2 The average amount of excess bits per window G. If we define the amount of bit reserved in the w-th window as $R^G(w) = \sum_{t=wG}^{(w+1)G} R[t]\tau$, and W as the total number of windows, then the m2 metric for respectively the guaranteed and maximal rate can be formally described as $\mathbb{E}[\max\{\rho_g G\tau - R^G(w), 0\} | w = 0..W-1]$ and $\mathbb{E}[\max\{R^G(w) - \rho_M G\tau, 0\} | w = 0..W-1]$. This is a representation of the

Scenario	User 1	User 2	User 3	User 4	User 5
1	SAT [150,250]	SAT [250,350]	SAT [350,400]	SAT [150,350]	SAT [50,100]
2	Starwars [50,150]	Alice [250,350]	Self-Similar [150,350]	SAT [150,350]	Sine2VS [50,120]
3	Starwars [50,150]	Sine2F [250,350]	Self-Similar [150,350]	SAT [150,350]	Sine2VS [50,120]
4	Sine2VS [150,250]	Sine2VS [150,250]	Sine2VS [250,300]	Sine2VS [150,350]	Sine2VS [50,400]
5	Sine2VS [150,250]	Sine2VS [150,250]	Sine2VS [250,300]	Sine2VS [150,350]	Self-Similar [0,0]

Table 12.3: Summary of scenarios. Listed for each user are traffic type, and [ρ_g, ρ_M] in Mbps.

severity of the average violation. A larger number indicates more severe violations. The metric can be visualized by imagining the surface above or below the required rate. Increasing G decreases the m2 metric as we smooth out excess bits over a larger window.

m3 $\mathbb{E}[B^G]$: where B^G is the set of consecutive violating G-sized windows. This metric gives an idea of how grouped violations are. For example, if this number is large, it means that a violation is resolved slowly.

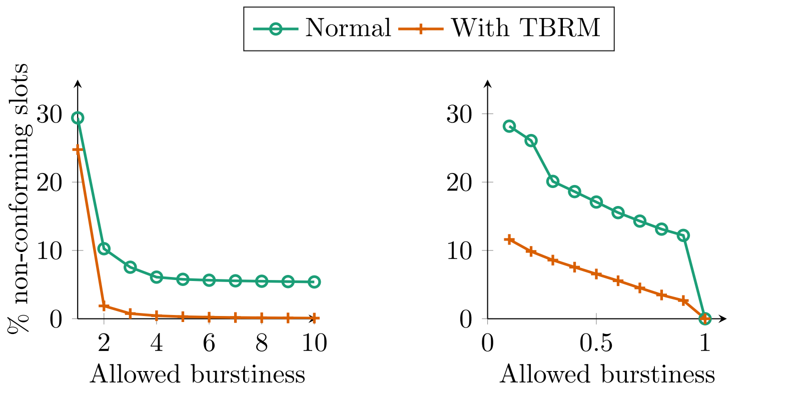
12.4.2 Results

12.4.2.1 Regular scenarios

In the following plots, we averaged over all schedulers and scenarios, as showing the individual schedulers would result in a cluttered plot. Important discrepancies between schedulers will be discussed in the text. Each plot has two curves, one of which displays the results for which no rate constraints were applied, as a base case, and the other has our TBRM algorithm applied.

12.4.2.1.1 m1 The m1 metric is shown in Figure 12.1, which displays on the x-axis the allowed burstiness, and on the y-axis the percentage of non-conforming slots.

If we examine Figure 12.1a, which shows the m1 metric for the upper bound, then we can see that for $x = 1$, the number of violations is close to the results of the unconstrained simulations. When we increase x, the allowed burstiness,



(a) m1: upper bound on rate (b) m1: lower bound on rate

Figure 12.1: m1 for the regular scenarios

however, we can observe that the violation probability quickly drops for our TBRM algorithm, and becomes almost 0 when the allowed burst size is $5\rho_M\tau$. This indicates that the violations occur irregularly spread. The unconstrained results remain fixed around 6% for a long time.

The underlying data show that for the constrained scenarios all the schedulers inhibit the same behavior: there is a steep decline in violations, going from $x = 1$ to $x = 2$, and then they gradually go to almost 0 for $x = 5$.

This behavior is the same for all schedulers over all traffic classes. However, the initial violation probability for SAT class is slightly lower than the video, self-similar and sine classes. The SAT traffic is easier to correct due to its queue based nature.

In the m1 plot of the guaranteed rate in Figure 12.1b, our domain is limited to $]0, 1]$: if $x = 1$, then it means that approximately in every slot we allow a deficit of $\rho_g\tau$ bit, which is obviously the maximum deficit we can attain per slot. In the plot, one can see that there is a much wider gap between the constrained and unconstrained scenarios, confirming the efficacy of our algorithm.

The data show here that the majority of the violations come from the MW and M-LWDF schedulers, and more specifically for the video streams. For example, in the TBRM scenarios, for $x = 0.1$, both schedulers have a violation probability of about 20%, while the other schedulers are closer to 6%.

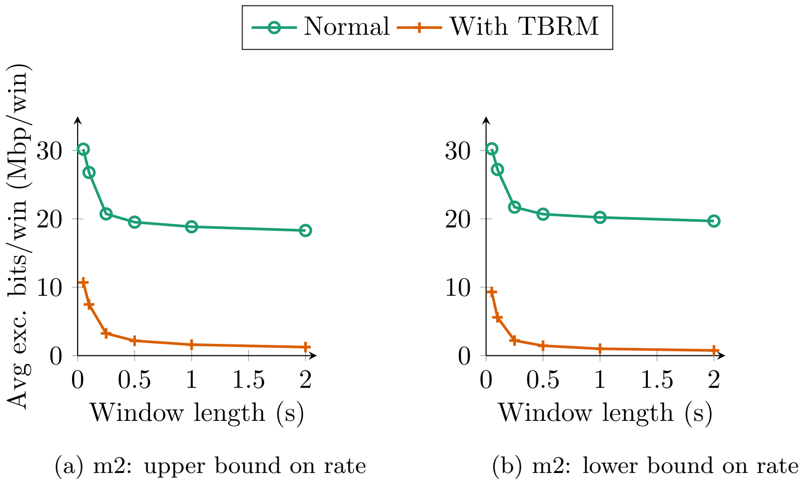
This difference can be explained by the fact that in those linear schedulers the queue length is used as a weight. This number is immediate, which causes a more unpredictable weight (especially in combination with a linear scheduler), making it more difficult to estimate a suitable weight modifier. Additionally, the weight can become 0 very easily. This requires the use of the additional θ^n modifier. Even though $\overline{\omega}$ is smoother, the switch between $\overline{\omega}$ and ω^n can be disruptive.

However, this extra factor is necessary, as simulations without this correction θ^n, result in a much higher violation probability.

The curve looks quite linear. This can be explained by the fact that the guaranteed rate violations are more evenly spread out.

12.4.2.1.2 m2 Ideally, we can limit the rate immediately. However, there is an inherent delay of 1 slot, and an elasticity in the form of a burst factor. Therefore, we study the rate, when we group $\frac{G}{\tau}$ slots into windows of size G.

The m2 metric in Figure 12.2, displays the average amount of violated bits per window, for increasing window sizes G.



(a) m2: upper bound on rate (b) m2: lower bound on rate

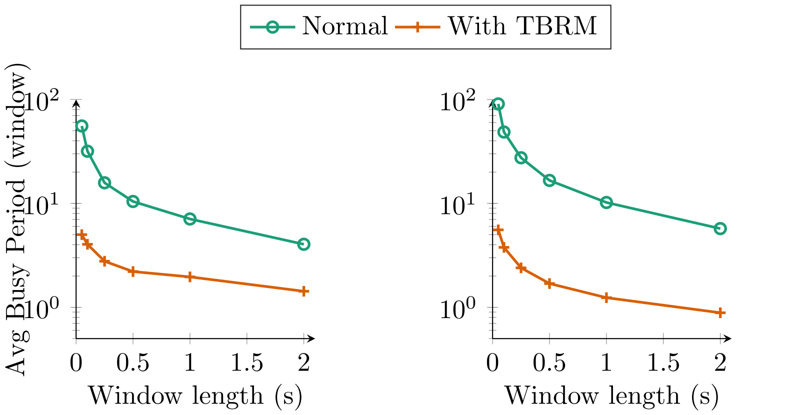
Figure 12.2: m2 for the regular scenarios

It can be observed that for a window size of $G = 0.05s$, for the unconstrained scenarios there is an average of 30 Mbit/window in excess of the target rate. When we constrain it using our algorithm, this drops to about 10 Mbit/window.

Increasing the window size G, averages out bursts. Like the results for m1, there is a steep decline until $G = 0.25s$, which coincides with 5 slots, after which the bit violations remains stable in the upper bound case. Though less pronounced, also here do the MW, M-LWDF and MD schedulers fare the worst for small window sizes, mainly for the Self-Similar traffic.

The decline implies that bursts are usually short-lived: overflow and good windows are usually close together, as they don't violate the constraints when merged. This is also confirmed in the m3 metric, below. The rate of decline is similar for the scenarios with and without TBRM applied.

12.4.2.1.3 m3 The last metric discusses the average length of a violation streak. Figure 12.3 shows the average number of successive windows that violate their constraints in a log-plot. The m3 metric, like the m2 metric, initially decreases quickly as the window size increases, and then slowly decreases. It can be clearly seen that, regardless of the unconstrained behavior, the TBRM algorithm limits the bursts to 5 windows, for $G = \tau$, dropping to 2 windows for $G = 5\tau$. These short bursts confirm that the algorithm is able to fix excesses within about 5 slots.



(a) m3: upper bound on rate (b) m3: lower bound on rate

Figure 12.3: m3 for the regular scenarios

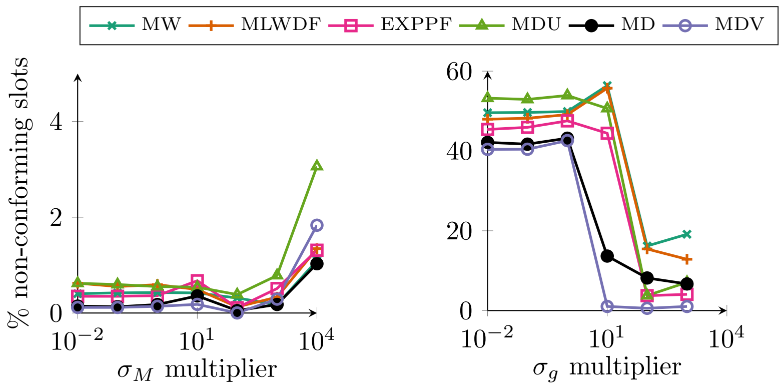
The main contributor to the average in this metric is the MDU scheduler. The data show that this is because whereas other schedulers consist of many smaller busy periods, the MDU scheduler has only one or two large busy periods, increasing the average significantly.

Without the MDU data, the average for 5 windows is about 1, for the minimal rate constraint, and 2 for the maximal rate constraint.

12.4.2.2 Study of parameter σ

In this section we look at the results for different values of the burst parameter σ. We let $\sigma_g = i\tau\rho_g$ and $\sigma_M = i\tau\rho_M$, for $i \in \left[10^{-2}, 10^4\right]$, and look at the effect on the m1 metric in Figure 12.4, which shows the results for the individual schedulers.

The plot shows that for the upper bound constraints, the violation probability is always very low (about 4% at most for $i = 10^4$). The lower bound, however, starts around 50% violation probability, and suddenly drops to smaller probabilities for $i = 10^2$.

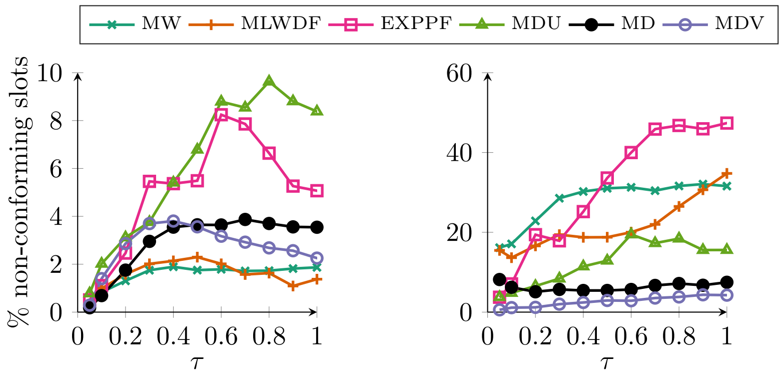


(a) m1: upper bound on rate (b) m1: lower bound on rate

Figure 12.4: m1 for the σ scenarios

Indeed, a small σ_M will overflow quickly, which causes its weight to be reduced swiftly, hence there will be fewer violations. As σ_M grows, the weight modifier will decrease much more slowly, leaving more room for violations. For the guaranteed rate, on the other hand, k_g cannot build up a deficit as fast as k_M, as discussed before. It is only when the growths of the deficits are balanced that the m1 metric can lower, which is around $i = 10^2$ and upwards.

The schedulers that perform the worst are, unsurprisingly, the MW and M-LWDF schedulers.



(a) m1: upper bound on rate (b) m1: lower bound on rate

Figure 12.5: m1 for the τ scenarios ($\sigma = 5\tau\rho$)

12.4.2.3 Study of parameter τ

In the previous simulations, we assumed a slot length of $\tau = 0.05s$. This study observes how the TBRM algorithm changes in function of τ.

In Figure 12.5 the m1 metric for $\sigma = 5\tau\rho$ is plotted. It can be observed that mainly the lower bound in Figure 12.5b is sensitive to an increasing slot length. This might be because with an increasing τ also grows the probability of a larger delay: if in a slot a low service rate was assigned erroneously, the delays or queues will increase and additionally it takes longer to correct, causing larger queues. Especially the EXP/PF (EXP/PF) scheduler suffers from this, as it is of the form exp(Γ), where Γ is the head-of-line. As the non-linear MD and MDV schedulers try to minimize the delay, they suffer less from an increase of slot size.

For the upper bound in Figure 12.5a only the MDU and EXP/PF schedulers seem to suffer from the increased slot size. This is probably due to the fact that it is easier to receive a service lower than ρ_M.

12.5 Related work

In [17, 162] the authors use virtual tokens as a measure for the average waiting time, and incorporate it with the M-LWDF [17] and EXP/PF [28] scheduling algorithm to warrant a minimal rate. It is, however, not transferable to other schedulers. In other schedulers, the guaranteed rate constraint is built into the scheduler itself [134, 196], but they are all scheduler-specific and don't allow enforcing a maximal data rate. The authors of [120] consider utility based throughput allocation subject to certain properties, but is only valid for linear utility functions. In [34] a related problem of maintaining an optimal service rate is proposed. In [18], the authors consider a generic algorithm with minimum and maximum rate constraints. It is, however, only applicable to schedulers that operate in function of an average rate. As such, it excludes for example the MW [180], MD and M-LWDF schedulers. Other schedulers, such as, MDU [169] and MDV [185] have a more elaborate utility function and are more difficult to characterize. The authors of [203] also employ a token system, but assume that users lie about their demands to strategically maximize their utility. In [128] constraints are applied to network slices of traffic aggregates in a 5G context, using an additive approach.

12.6 Conclusion

In this section we looked at restricting the service rates given to users in a cross-layer scheduler setting. This is useful in for example admission control or providing different service levels to users.

We implemented this using a low-complexity algorithm that modifies the weights in a network utility maximization problem, using the concept of token buckets. We first discussed cross-layer scheduling, and the need to both upper and lower limit data rates assigned to users. Then we proposed the TBRM algorithm, and applied the algorithm to simulations. We ran these simulations for six different schedulers, and multiple scenarios demonstrating that using our approach it is possible to limit the service rate, within error, after about five slots for the maximal and guaranteed service rate for most schedulers. Schedulers that progress smoothly are easier to constrain than schedulers that can behave wildly, such as EXP/PF for long slot times, MW or M-LWDF. These are more difficult to restrain with respect to guaranteeing a lower bound on the service rate.

Chapter 13
Conclusion

In this thesis we looked at two different subjects. In Part I, we analyzed the performance of a SP scheduler, which has a number of high priority CBR queues, medium priority VoIP queues, low priority video queues and background traffic. This particular kind of setup is important in industrial networks, in use by for example railroad or power companies, who want to replace old, expensive in maintenance, and dedicated hardware with modern IP networks. The applications on these networks often have very stringent requirements, and the analysis we performed helps in determining whether a network can support these applications securely. Key in this analysis is the characterization of the busy period of an aggregate of CBR sources. The HP queues can undergo a vacation period, which influences the busy period and delay distributions. We then extended the results from single hop to multi-hop, in particular for the E2E delay and IPDV. We considered the case where the through-traffic has a low load, leading to a simple solution. When the load of the through-traffic increases, a more complex algorithm is necessary, as low priority and high priority through-traffic can lead to extended delays. The analytical results were compared to multiple different simulated scenarios, and were found to provide a good bound in most cases, for all the traffic classes considered.

In the second part of this thesis, Part II, we developed algorithms for use in cross-layer contexts. We developed resource allocation algorithms for two distinct use cases. In the first use case we developed an algorithm that was designed for the context of a shared medium, such as recent technologies of DSL, and LTE and 5G settings. Using a well-defined interface between the physical and upper layers, a new cross-layer scheduler, called the MDV scheduler, has been shown through simulations to offer an excellent performance with respect to delay, delay violations and throughput for various communication technologies. The scheduler makes use of the different metrics (arrival rate, queue size, HOL, and the short-term PLR) to obtain weights that reflect the current requirements of users' applications. We have discussed some properties of the MDV scheduler in the DSL context. For example, for convex rate regions the scheduler is throughput optimal, i.e. the number of packets in the system can always be bounded, whenever any other cross-layer scheduler can bound the number of packets. For non-convex rate regions, which occur for example in the 5GBB scenarios we consider, the number of packets is not necessarily bounded. However, this rarely occurs in practice.

The simulations for the DSL scenarios looked at the PLR, throughput and average delay for a variety of scenarios and a number of schedulers from literature. The MDV's performance was often similar or significantly better than those other schedulers, regardless of usage of an intra-user EDF scheduler or each flow receiving its own "channel".

The simulations for the LTE and 5G scenarios also show the excellent performance of the MDV scheduler. The PLR and throughput are comparable or better than other schedulers from literature. The fairness, however, is overall less than the other schedulers. In some cases this is because the mean value is close to zero, skewing the result.

The second use case deals with resource allocation for an aggregate of sources, where there is a significant delay between requesting a service rate, and acquiring this service rate. This can be of use in satellite communications, where many users are grouped together, and resources are allocated for this group of users. A simple ad hoc algorithm and a more complex algorithm based on an admission control mechanism was developed. In the simulations we compare the algorithms with an ideal one, which has access to future traffic arrivals, for the UDP and TCP protocols. We look at the delay and the efficiency. The simple algorithm's performance is in some cases good, while in other cases it cannot allocate resources as required. The more complex *MBAC* algorithm can handle most traffic types reasonably well, and usually performing better than the other algorithms. Compared to the ideal allocation, the efficiency is worst-case 40% less.

We finally developed the TBRM algorithm for cross-layer schedulers, which constrains the service rate between an upper- and a lower bound by manipulating the weights of the flow's. This is useful to e.g. limit ill-behaving flows or enforce some types of service level agreements. The algorithm is based on token buckets. In the simulations we applied the algorithm to six different cross-layer schedulers. We found that service rates are usually restored within their bounds after about five slots. Some schedulers are more difficult to constrain due to their erratic weight changes.

13.1 Future work

For the cross-layering, it might be interesting to see how machine learning could be incorporated [194] For example, at the DSL physical layer, deep learning has shown to improve conventional methods of resource allocation, with respect to computation of speed [29]. Likewise, deep learning has been successfully applied to the cross-layer scheduling component in wireless networks. For example, in

[146], the authors applied reinforcement learning techniques to the cross-layer UE of an LTE network, and in [13], a deep reinforcement learning agent, called LEASCH, has learned to schedule from scratch in 5G networks. Similar techniques might be applied in the context of DSL networks to increase the throughput and reduce the packet loss even more.

Obtaining bounds on the delay is quite difficult. Deriving bounds using Lyapunov techniques result in delays that are very loose, and not usable in practice. Implementing the system as a Markov chain, results in an explosion of states, due to the stateful nature of the MDV scheduler. In [114] a method is developed using Petri nets for stochastic wireless networks, that tries to reduce the computational complexity. A similar approach might be applied to the MDV scheduler. Approximations to these delay bounds could then be used to implement an admission control algorithm.

Appendix A

Constructing matrix H

This appendix shows an algorithm to compute the matrix **H** in equations (3.16) and (3.17). Note that, in contrast to Section 3.6, we use zero-based numbering for vectors to make it more suitable for implementation in a programming language like python or c++.

Algorithm 1: Constructing the **H** matrix

```
func H(i: int, Q: int): Matrix[int] =
    let M = N.len
    var curRow = Vector[int](M)

    proc helper(j: int): Matrix[int] =
        var result: Matrix[int]
        if j $\geq$ M: return result

        for k in 0 .. N[j]:
            curRow[j] = k
            curRow[j+1 .. M−1] = 0

            let len = curRow$^T$*L
            if len > Q: break
            elif len = Q:
                if curRow[i] > 0:
                    result.addRow(curRow)
                endif
                break
            endif

            result.extend(helper(j+1))
        return result

    return helper(0, result)
```

Appendix B

Calculating the vacation pmf

Algorithm 1: Calculating the vacation pmf

```
var vacation:Pmf

for scenario in permutations(['0','A','B','C','D','E'], nLinks):
    var ps:seq[P]
    var vs:seq[Pmf]

    for i in 0 .. nLinks−1:
        let $V_i^{HP+LP}$=merged([
            $V_i^{HP}$.scaledP($\rho_i^{HP}(1+\rho_i^{LP})/\rho_i$),
            $V_i^{LP}$.scaledP($\rho_i^{LP}\cdot((1-\rho_i)+\rho_i^{LP})/\rho_i$)
            ])

        case (if i>0: scenario[i−1] else: '0')+scenario[i]
            of "00","A0": ps.add(($1-\rho_i$)); vs.add(0)
            of "0A","AA": ps.add($\rho_i^{CT}$); vs.add($V_i^{HP+LP}$)
            of "0B","AB": ps.add(0);       vs.add(0)
            of "0C","AC": ps.add($\rho_i^{LP,TT}$); vs.add($V_i^{LP,TT}$)
            of "0D","AD": ps.add(0);       vs.add(0)
            of "0E","AE": ps.add($\rho_i^{HP,TT}$); vs.add($V_i^{HP}$)

            of "B0","C0": ps.add(0);       vs.add(0)
            of "BA","CA": ps.add($\rho_i^{CT}$); vs.add($V_i^{HP+LP}$)
            of "BB","CB": ps.add(($1-\rho_i$)); vs.add($L'^{LP,TT}$)
            of "BC","CC": ps.add($\rho_i^{LP,TT}$); vs.add($V_i^{LP,TT}$)
            of "BD","CD": ps.add(0);       vs.add(0)
            of "BE","CE": ps.add($\rho_i^{HP,TT}$); vs.add($V_i^{HP}$)

            of "D0","E0": ps.add(0);       vs.add(0)
            of "DA","EA": ps.add($\rho_i^{CT}$); vs.add($V_i^{HP}$)
            of "DB","EB": ps.add(0);       vs.add(0)
            of "DC","EC": ps.add($\rho_i^{LP,TT}$); vs.add($V_i^{HP}$)
            of "DD","ED": ps.add(($1-\rho_i$)); vs.add($\mathcal{L}(()V_i^{HP})$)
            of "DE","EE": ps.add($\rho_i^{HP,TT}$); vs.add($V_i^{HP}$)

        if $\Pi_{p\in \text{ps}}(p)$ == 0: break

    vacation.addWeighted(vs.convoluted(), $\Pi_{p\in \text{ps}}(p)$)

```

Appendix C

Proof of stability for constant A and B

For the proof of Theorem 1 on page 121 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 $\lambda = [\lambda_1, \dots, \lambda_n] = [\frac{\nu_1}{\mu_1}, \dots, \frac{\nu_n}{\mu_n}]$. By including the residual inter-arrival and service times we can extend this result to renewal arrival processes and generally distributed packet sizes [55], and obtain results for general distributions.

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

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

where R_i^* is given by Equation (9.14).

In the proof we look at the fluid system corresponding to the Markov process Q_i. In [55] it is shown that if a fluid limit model eventually reaches zero, regardless of its initial configuration, and remains there, then the original queuing network is positive Harris recurrent, and we consider the system stable. Positive Harris recurrent means that in the Markov chain every state will be visited an unbounded number of times, with probability 1.

Proof. We look at the fluid system, where the sum of the initial queues grow to infinity:

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

If this limit exists, we have that $\sum_i X_i(0) = 1$. Define $\mathbf{X}[t] = [X_1[t], \dots, X_n[t]]$. Given an initial distribution of $\mathbf{X}(0)$, it follows from the strong law of large numbers that the evolution of the fluid $\mathbf{X}[t]$ is defined by

$$\frac{d}{dt} X_i = \nu_i - \mu_i R_i[t]$$

for all i,t such that $X_i[t] > 0$. $\mathbf{R}[t]$ is the solution to Equation (9.14). If the traffic conditions

$$\lambda \in \mathcal{C}^\alpha \tag{C.1}$$

are satisfied, then we show that there exists a constant $T > 0$, such that $\mathbf{X}[t] = \mathbf{0}, \forall t \geq T$, which, according to [55], implies stability.

For this, we define Lyapunov function F and scheduler G as

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

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

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 $F(\mathbf{X})$ with respect to t we get the Lyapunov drift:

$$\frac{d}{dt}F(\mathbf{X}) = \sum_i \frac{(A_i X_i + B_i)^{\beta-1}}{(\lambda_i + \zeta)^\alpha}(\lambda_i - R_i), \tag{C.2}$$

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

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

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

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

$$\nabla G(\mathbf{u}) \cdot (\mathbf{u} - \mathbf{R}) \leq 0. \tag{C.3}$$

Under the stability condition (C.1), we can find an $\epsilon > 0$ such that $\mathbf{u} = (1+\epsilon)\lambda \in \mathcal{C}^\alpha$. Applying $\mathbf{u}$ to (C.3) results in

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

which can be reduced to

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

This can be rewritten using (C.2) to obtain

$$
\begin{aligned}
\frac{d}{dt}F(\mathbf{X}) &\le -\epsilon \sum_i \mu_i^{-1}(\lambda_i+\zeta)^{-\alpha}(A_iX_i+B_i)^{\beta-1} \\
&\le -\epsilon \sqrt[\beta]{\min_i(\mu_i^{-1}(\lambda_i+\zeta)^{-\alpha})}\cdot \\
&\quad \left(\sum \mu_i^{-1}(\lambda_i+\zeta)^{-\alpha}(A_iX_i+B_i)^{\beta}\right)^{\frac{\beta-1}{\beta}} \qquad \text{(C.4)} \\
&\le -\epsilon \sqrt[\beta]{\min_i(\mu_i^{-1}(\lambda_i+\zeta)^{-\alpha})}(\beta\min(\mathbf{A}))^{\frac{\beta-1}{\beta}}\cdot \\
&\quad \left(\sum \frac{\mu_i^{-1}(\lambda_i+\zeta)^{-\alpha}}{\beta A_i}(A_iX_i+B_i)^{\beta}\right)^{\frac{\beta-1}{\beta}} \qquad \text{(C.5)} \\
&= -\theta F(\mathbf{X})^{\frac{\beta-1}{\beta}}
\end{aligned}
$$

In step (C.4) we employed the well-known inequality $||\mathbf{a}||_q \le ||\mathbf{a}||_p \le n^{1/p-1/q}||\mathbf{a}||_q$, where $||\mathbf{a}||_p$ 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 $\mathbf{A} = [A_1, \ldots, A_n]$ then

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

Step (C.5) uses Abel's inequality.

Now, if there exists $T > 0$ for which $F(\mathbf{X}(T)) = 0$, then it is clear that $F(\mathbf{X}[t])$ will always be able return to 0, $\forall t \ge T$.

Furthermore,

$$
\begin{aligned}
&\frac{d}{dt}F(\mathbf{X}) \le -\theta F(\mathbf{X})^{\frac{\beta-1}{\beta}} \\
\iff& \frac{d}{dt}\ln(F(\mathbf{X}))F(\mathbf{X})^{\frac{1}{\beta}} \le -\theta.
\end{aligned}
$$

Integrating both sides results in

$$\begin{aligned}
&\int_0^t \frac{d}{dt}\ln(F(\mathbf{X}(s)))F(\mathbf{X}(s))^{\frac{1}{\beta}}ds \leq \int_0^t -\theta ds \\
\iff\quad & \beta F(\mathbf{X}(s))^{\frac{1}{\beta}}|_0^t \leq -\theta t \\
\iff\quad & F(\mathbf{X}[t]) \leq \left(F(\mathbf{X}(0))^{\frac{1}{\beta}} - \frac{\theta}{\beta}t\right)^{\beta}.
\end{aligned}$$

This implies that $F(\mathbf{X}[t]) = 0$, and thus also $\mathbf{X}[t] = 0$ for all $t \geq T$, with

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

□

The inclusion of a constant B_i does not impact the stability region (but does increase T). Also, multiplying $\mathbf{A}$ 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 $\beta = 2$, $\alpha = 2$, $\mathbf{A} = \mathbf{1}$ and $\mathbf{B} = \mathbf{0}$, 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 $\lambda \in \mathcal{C}^{\alpha}$ 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 [33] 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 D

Proof of stability for time-dependent A and B

We now show for Corollary 1.1 on page 121 that the fluid limit model corresponding to the non-homogeneous Markov process, i.e. for time-dependent A and B, also reaches zero, regardless of its initial configuration, and remains there, with probability 1.

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

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

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

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

$$\frac{d}{dt}F(\mathbf{X}) = \sum_i (\lambda_i+\zeta)^{-\alpha}(A_iX_i+B_i)^{\beta-1}\cdot(\lambda_i - R_i) + a + b$$

where

$$a = \sum_i (\lambda_i+\zeta)^{-\alpha}\mu_i^{-1}\frac{d}{dt}A_i \cdot (A_iX_i+B_i)^{\beta-1}\cdot\left(\frac{X_i}{A_i} - \frac{A_iX_i+B_i}{A_i^2\beta}\right)$$

and

$$b = \sum_i (\lambda_i+\zeta)^{-\alpha}\mu_i^{-1}\frac{d}{dt}B_i \cdot \frac{(A_iX_i+B_i)^{\beta-1}}{A_i}.$$

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

$$\frac{d}{dt}F(\mathbf{X}) \le -\theta F(\mathbf{X})^{\frac{\beta-1}{\beta}} + a + b.$$

In the fluid system considered here, a and b depend on $\frac{d}{dt}A_i$ and $\frac{d}{dt}B_i$ 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 $F(\mathbf{X}[t])$ possibly makes a jump, due to the weights changing. However, this does not impact the queue sizes themselves. We have thus that $\frac{d}{dt}F(\mathbf{X}) \leq 0$.

As before, we can rewrite the equation to obtain

$$F(\mathbf{X}[t]) \leq \left(F(\mathbf{X}(0))^{\frac{1}{\beta}} - \eta \sum_{s=0}^{t} \min(\mathbf{A}(s))^{\frac{\beta-1}{\beta}} \right)^{\beta} \tag{D.1}$$

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

From (D.1) we can again obtain T. $\square$

Appendix E

DSL Oracle scheduler

In this appendix, we describe the algorithm to determine the ideal rate for the DSL simulations of Section 9.6. The algorithm runs in two steps. First, we calculate for each flow the minimal required service rate for the next slot. This first step does not depend on the rate region. Then, in the second step, we assign the service rates. We first check if each flow can have its minimal required rate, if this is possible, we maximize the system throughput. If some flows cannot have their minimum required service rate, then service rates are assigned such that the number of delay violations is minimized. We now describe these two steps in more detail.

Step 1: calculate the minimal required rate We define $\check{\rho}_{i,t+1}$ to be the minimal required service rate in slot $t+1$ to satisfy flow i's delay requirement. It is based on the observed past delays, and the arrivals in the next M slots. Assume function $sim(\rho_t, \rho_{t+1}, q_t, D_t, A_{[t,t+M[})$ simulates the queue evolution for the arrivals in the next M slots given rates ρ_t and ρ_{t+1} for respective slots, and returns the resulting delay distribution and the maximal delay encountered during $[t, t+M]$. Then we can find an approximation of the smallest service $\check{\rho}_{i,t+1}$ for which the ε-percentile of this distribution is less than $\hat{T}$, as described in (Listing E.1).

Lines 1-3 define the constants for the algorithm. A flow will always receive a minimal service rate $\check{R}$. The function starting at line 5 implements the search procedure. This algorithm is based on a binary search. A distinction is made between a flow that can satisfy its QoS or not. If $\hat{R}$ cannot satisfy the QoS (i.e. $isPirOk$=false), then the minimal service rate that does not worsen the metric is used (see lines 21-23). If it is possible to satisfy the QoS ($isPirOk$=true) in the next slot, then the minimal service rate that will satisfy the requirements is selected (line 20). The lines 25-29 perform the binary search.

The function defined at line 33 returns true if the delay distribution is suitable for the QoS requirements. To avoid excessive packet delays, the maximal delay is bounded by M.

Algorithm 1: Finding $\check{\rho}_{t+1}$

```
const $\hat{T}$, $\varepsilon$
const $\check{R}$ := minimal service rate, $\hat{R}$
const $M := 2\hat{T}$

proc find_minimal_rate($\rho_t$, $Q_t$, $D_t$, $A_{[t,t+M[}$): Rate =
    var $lo := \check{R}$, $hi := \hat{R}$
    if $\#Q_t$ = 0: return $\check{R}$

    var $\check{\rho}_{t+1}$: Rate
    const $D_{pir}$, $D_{max,pir}$ := sim($\rho_t$, $\hat{R}$, $D_t$, $A_{[t,t+M[}$)
    const $isPirOk$ := delay_distr_is_ok($D_{pir}$, $D_{max,pir}$)
    var $D_{best}$ := $D_{pir}$

    for i in 1 .. 20:
        const $mid$ := (lo+hi)/2
        const $D$, $D_{max}$ := sim($\rho_t$, $mid$, $D_t$, $A_{[t,t+M[}$)

        var $isOk$: bool
        if $isPirOk$:
            $isOk$ := delay_distr_is_ok(D, $D_{max}$)
        else:
            $isOk$ := ($D \leq D_{best}$)
            $D_{best}$ := $\min(D, D_{best})$

        if $isOk$:
            $hi$ := $mid$
            $\check{\rho}_{t+1}$ := $mid$
        else:
            $lo$ := $mid$

    return $\check{\rho}_{t+1}$

proc delay_distr_is_ok(D, $D_{max}$): bool =
    const $p$ := percentile($D$, 1−$\varepsilon$)
    return ($p \leq \hat{T}$) and ($D_{max} \leq M$)
```

Step 2: distribute the service rates After having calculated $\check{\rho}_{i,t+1}$ for all flows in step 1, the operating point $\mathbf{R}$ that satisfies the users' requirements best is now selected and the service rates are distributed over the flows.

Denote the set of configurations that satisfy all $\check{\rho}_{t+1}$ by $\mathbf{R}_{ok} \subset \hat{\mathcal{R}}$. Then there are two cases to consider: $|\mathbf{R}_{ok}| > 0$, i.e. at least one configuration is suitable, or $|\mathbf{R}_{ok}| = 0$ indicating that there is at least one flow that will not receive its required minimal rate.

When $|\mathbf{R}_{ok}| > 0$, the optimal operating point is chosen as

$$\mathbf{R}^* = \arg\max_{\mathbf{R}\in\mathbf{R}_{ok}} \sum_{n=1}^{N} R,$$

i.e. optimizing the system throughput. A flow then receives a rate relative to its weight:

$$\rho^n = \check{\rho}_{i,t+1} + (R^{*n} - \sum_{j=1}^{\phi} \check{\rho}_{j,t+1}) \cdot \frac{\check{\rho}_{i,t+1}}{\sum_{j=1}^{\phi} \check{\rho}_{j,t+1}}. \tag{E.1}$$

When $|\mathbf{R}_{ok}| = 0$, the aim will be to limit the delay violations. To accomplish this, the optimal operating point

$$\mathbf{R}^* = \arg\min_{\mathbf{R}\in\hat{\mathcal{R}}} \sum_{n=1}^{N} \sum_{i=1}^{\phi} (\check{\rho}_{i,t+1} - \frac{\check{\rho}_{i,t+1}}{\sum_{j=1}^{\phi} \check{\rho}_{j,t+1}} R)^+,$$

is selected, i.e. we minimize the average service rate deficit. The rate assignment of the flows is the same as for the first case, as described in Equation (E.1).

Calculating the ideal rate is computationally intensive, hence we use two shortcuts. First, we only consider the arrivals in the next $M = 2$ slots. Increasing M would allow for better handling of bursts and a more accurate calculation of the required rate, but at the cost of increased time complexity. Second, we use the minimal required service rate rather than the delay distribution to calculate the required rate. This approximation may not always be valid. However, evaluating the delay distribution for every rate tuple would increase the time complexity considerably. Hence, this Oracle scheduler gives an approximation to the ideal scheduler with respect to the delay violation metric. It fails mainly in high load situations: in such cases often the minimal required service rate cannot be satisfied, and the wrong flow is sacrificed to give up service rate. Without evaluating the delay distribution, it is impossible to know the effects of this sacrifice.

References

[1] Alcatel-Lucent to provide communications system to connect railway lines in two Spanish cities. https://www.telecomlead.com/latest-news/alcatel-lucent-to-provide-communications-system-to-connect-railway-lines-in-two-spanish-cities-19601. Accessed: 2021-07-01. 17

[2] INET Framework. https://inet.omnetpp.org/. 72

[3] Next stop Berlin – IRJ's full InnoTrans preview. https://www.railjournal.com/news/next-stop-berlin/. Accessed: 2021-07-01. 17

[4] OMNeT++ Network Simulation Framework. http://www.omnetpp.org/. 72

[5] Sweden's national rail administration Banverket relies on innovations from Alcatel-Lucent to help ensure the safety and smooth operation of rail traffic. https://www.webwire.com/ViewPressRel.asp?aId=94695. Accessed: 2021-07-01. 17

[6] TRANSPOWER NEW ZEALAND Providing Modern Communications for New Zeland's National Electricity Grid. http://www.pexx.net/pdfs/casestudies/alcatel_lucent/mpr9500/CS_Transpower_6_010408.pdf. Accessed: 2021-07-01. 17

[7] Using Moxa Ethernet Solutions to Create a Reliable Substation Automation System. https://www.moxa.com.tw/applications/Substation_Automation_System.htm. Accessed: 2021-07-01. 17

[8] N. Adesh and A. Renuka. Adaptive downlink packet scheduling in lte networks based on queue monitoring. *Wireless Networks*, 25(6):3149–3166, 2019. 144

[9] M. Aguado, E. Jacob, J. Matias, C. Conde, and M. Berbineau. Deploying cctv as an ethernet service over the wimax mobile network in the public transport scenario. In *2009 IEEE International Conference on Communications Workshops*, pages 1–5. IEEE, 2009. 60

[10] A. Aguiar, A. Wolisz, and H. Lederer. Utility-based packet scheduler for wireless communications. *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, pages 863–870, 2006. 118

[11] 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)*, pages 24–29. IEEE, 2018. 131

[12] M. Akerele, I. Al-Anbagi, and M. Erol-Kantarci. A fiber-wireless sensor networks qos mechanism for smart grid applications. *IEEE Access*, 7:37601–37610, 2019. 25

[13] F. Al-Tam, N. Correia, and J. Rodriguez. Learn to schedule (leasch): A deep reinforcement learning approach for radio resource scheduling in the 5g mac layer. *IEEE Access*, 8:108088–108101, 2020. 199

[14] M. Allman, S. Dawkins, D. Glover, J. Griner, D. Tran, T. Henderson, J. Heidemann, J. Touch, H. Kruse, S. Ostermann, et al. Ongoing tcp research related to satellites. Technical report, IETF, 2000. 178

[15] S. Andreev, A. Anisimov, Y. Koucheryavy, and A. Turlikov. Practical traffic generation model for wireless networks. In *Fourth ERCIM Workshop on Emobility*, page 61. Citeseer, 2010. 46

[16] 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*, 18(02):191—217, 2004. 111

[17] 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*, 39(2):150–154, 2001. 114, 144, 188, 194

[18] M. Andrews, L. Qian, and A. Stolyar. Optimal utility based multi-user throughput allocation subject to throughput constraints. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 4, page 2415—2424. IEEE, 2005. 194

[19] 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*, 8(3):1618, 2018. 131, 144, 150

[20] 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*, 68(2):259–275, 2018. 111

[21] S. Asmussen. *Applied probability and queues*, volume 51. Springer Science & Business Media, 2008. 174

[22] S. Azodolmolky, R. Nejabati, M. Pazouki, P. Wieder, R. Yahyapour, and D. Simeonidou. An analytical model for software defined networking: A

network calculus-based approach. In *2013 IEEE Global Communications Conference (GLOBECOM)*, pages 1397–1402. IEEE, IEEE, 2013. 35

[23] A. Azzouni and G. Pujolle. A long short-term memory recurrent neural network framework for network traffic matrix prediction. *arXiv preprint arXiv:1705.05690*, 2017. 161

[24] J. W. Baek, H. W. Lee, S. Ahn, and Y. H. Bae. Exact time-dependent solutions for the m/d/1 queue. *Operations Research Letters*, 44(5):692–695, 2016. 56

[25] X. Bai and A. Shami. Modeling self-similar traffic for network simulation. *arXiv preprint arXiv:1308.3842*, 2013. 188

[26] M. Barabas, G. Boanea, A. B. Rus, V. Dobrota, and J. Domingo-Pascual. Evaluation of network traffic prediction based on neural networks with multi-task learning and multiresolution decomposition. In *Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on*, pages 95–102. IEEE, 2011. 161

[27] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *Journal of the ACM (JACM)*, 22(2):248–260, 1975. 20

[28] 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*, pages 1–5. IEEE, 2009. 111, 194

[29] P. Behmandpoor, J. Verdyck, and M. Moonen. Deep learning-based cross-layer resource allocation for wired communication systems. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4120–4124. IEEE, 2021. 198

[30] 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)*, pages 201–206. IEEE, 2015. 131, 144, 150

[31] N. Benammar, F. Ridouard, H. Bauer, and P. Richard. Forward end-to-end delay analysis extension for fp/fifo policy in afdx networks. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2017. 19

[32] T. Bonald and L. Massoulié. Impact of fairness on internet performance. In *ACM SIGMETRICS Performance Evaluation Review*, volume 29, pages 82–91. ACM, 2001. 120

[33] 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*, pages 327–332. IEEE, 2006. 121, 208

[34] S. Borst and P. Whiting. Dynamic channel-sensitive scheduling algorithms for wireless data throughput optimization. *IEEE Transactions on Vehicular Technology*, 52(3):569–586, 2003. 194

[35] L. Breslau, S. Jamin, and S. Shenker. Comments on the performance of measurement-based admission control algorithms. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1233–1242. IEEE, 2000. 164

[36] O. Brun and J.-M. Garcia. Analytical solution of finite capacity m/d/1 queues. *Journal of Applied Probability*, pages 1092–1098, 2000. 56

[37] R. Bruno, R. Garroppo, and S. Giordano. Token bucket dimensioning for aggregate voip sources. In *Proceedings of IEEE ATM Workshop 2000*, 2000. 54, 115, 170

[38] M. Buchli, D. De Vleeschauwer, J. Janssen, and G. H. Petit. Policing aggregates of voice traffic with the token bucket algorithm. In *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No. 02CH37333)*, volume 4, pages 2547–2551. IEEE, 2002. 47, 48

[39] A. Burchard, J. Liebeherr, and S. D. Patek. A min-plus calculus for end-to-end statistical service guarantees. *IEEE Transactions on Information Theory*, 52(9):4105–4114, 2006. 21

[40] 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*, 15:678–700, 2013. 118, 145

[41] E. Castillo. *Extreme value theory in engineering.* Elsevier, 2012. 165

[42] R. Chakravorty, S. Katti, J. Crowcroft, and I. Pratt. Flow aggregation for enhanced tcp over wide-area wireless. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1754–1764. IEEE, 2003. 182

[43] Y. H. Chan, T. Randhawa, and S. Hardy. Traffic prediction based access control using different video traffic models in 3g cdma high speed data networks. In *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pages 227–232. ACM, 2006. 160

[44] Z. Chen, N. Pappas, M. Kountouris, and V. Angelakis. Throughput with delay constraints in a shared access network with priorities. *IEEE Transactions on Wireless Communications*, 17(9):5885–5899, 2018. 19

[45] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, 2007. 102

[46] P. Chini, G. Giambene, D. Bartolini, M. Luglio, and C. Roseti. Dynamic resource allocation based on a tcp-mac cross-layer approach for dvb-rcs satellite networks. *International Journal of Satellite Communications and Networking*, 24(5):367–385, 2006. 177

[47] L. Chisci, R. Fantacci, and T. Pecorella. Predictive bandwidth control for geo satellite networks. In *2004 IEEE International Conference on Communications (IEEE Cat. No. 04CH37577)*, volume 7, pages 3958–3962. IEEE, 2004. 160

[48] J.-G. Choi and S. Bahk. Cell-throughput analysis of the proportional fair scheduler in the single-cell environment. *IEEE Transactions on Vehicular Technology*, 56(2):766–778, 2007. 144

[49] S. Chong, S.-q. Li, and J. Ghosh. Predictive dynamic bandwidth allocation for efficient transport of real-time vbr video over atm. *IEEE Journal on Selected Areas in Communications*, 13(1):12–23, 1995. 161

[50] F. Ciucu. Network calculus delay bounds in queueing networks with exact solutions. In *International Teletraffic Congress*, pages 495–506. Springer, 2007. 21

[51] F. Ciucu. End-to-end delay analysis for networks with partial assumptions of statistical independence. In *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, pages 1–11. Citeseer, 2009. 21

[52] F. Ciucu, A. Burchard, and J. Liebeherr. Scaling properties of statistical end-to-end bounds in the network calculus. *IEEE Transactions on Information Theory*, 52(6):2300–2312, 2006. 20

[53] R. Coelho, G. Fohler, and J.-L. Scharbarg. Dimensioning buffers for afdx networks with multiple priorities virtual links. In *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, pages 10A5–1. IEEE, 2015. 22

[54] R. L. Cruz. A calculus for network delay. i. network elements in isolation. *Information Theory, IEEE Transactions on*, 37(1):114–131, 1991. 183

[55] 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*, 5(1):49–77, 1995. 205, 206

[56] J. Daigle and J. Langford. Models for analysis of packet voice communications systems. *IEEE Journal on selected areas in communications*, 4(6):847–855, 1986. 46

[57] C. Demichelis and P. Chimento. IP Packet Delay Variation metric for IP performance metrics (IPPM). Technical report, RFC 3393, November, 2002. 33

[58] Demichelis, Carlo and Chimento, Philip. Ip packet delay variation metric for ip performance metrics (ippm). 2002. 33

[59] T. Demoor, J. Walraevens, D. Fiems, and H. Bruneel. Performance analysis of a priority queue: Expedited forwarding phb in diffserv. *AEU-International Journal of Electronics and Communications*, 65(3):190–197, 2011. 20

[60] H. Dong, Y. Lin, Y. Zhang, Z. Zhou, and Z. Zhang. Using static priority queueing to optimize the avionics full duplex switched ethernet. In *2013 Ninth International Conference on Natural Computation (ICNC)*, pages 1610–1616. IEEE, 2013. 22

[61] 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*, 78(11):15507–15525, 2019. 112

[62] M. I. Elhadad, W. El-Shafai, E.-S. M. El-Rabaie, M. Abd-Elnaby, and F. E. Abd El-Samie. Enhanced fair earliest due date first scheduling strategy for multimedia applications in lte downlink framework. *International Journal of Communication Systems*, 33(6):e4190, 2020. 150

[63] A. Fiaschetti, A. Pietrabissa, and L. Pimpinella. A cross-layer approach to dynamic bandwidth allocation in satellite networks. In *International Conference on Personal Satellite Services*, pages 114–129. Springer, 2010. 160

[64] M. Fidler. An end-to-end probabilistic network calculus with moment generating functions. In *200614th IEEE International Workshop on Quality of Service*, pages 261–270. IEEE, 2006. 21

[65] M. Fidler. Survey of deterministic and stochastic service curve models in the network calculus. *Communications Surveys & Tutorials, IEEE*, 12(1):59–86, 2010. 21

[66] M. Fidler and R. Persaud. M| g| 1 priority scheduling with discrete pre-emption points: on the impacts of fragmentation on ip qos. *Computer Communications*, 27(12):1183–1196, 2004. 20

[67] T. Finch. Incremental calculation of weighted mean and variance. *University of Cambridge*, 4(11-5):41–42, 2009. 166

[68] B. Fu, Y. Xiao, H. Deng, and H. Zeng. A survey of cross-layer designs in wireless networks. *IEEE Communications Surveys & Tutorials*, 16(1):110–126, 2013. 102

[69] J. M. Gablonsky and C. T. Kelley. A locally-biased form of the direct algorithm. *Journal of Global Optimization*, 21(1):27–37, 2001. 187

[70] R. G. Garroppo and C. Callegari. Prediction of mobile networks traffic: enhancement of the nmls technique. In *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–6. IEEE, 2020. 171

[71] R. G. Garroppo, S. Giordano, and M. Pagano. Estimation of token bucket parameters for aggregated voip sources. *International Journal of Communication Systems*, 15(10):851–866, 2002. 47, 48

[72] R. G. Garroppo, S. Giordano, M. Pagano, and G. Procissi. On traffic prediction for resource allocation: A chebyshev bound based allocation scheme. *Computer Communications*, 31(16):3741–3751, 2008. 160

[73] L. Georgiadis, M. J. Neely, L. Tassiulas, et al. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends® in Networking*, 1(1):1–144, 2006. 114

[74] Y. Ghiassi-Farrokhfal, J. Liebeherr, and A. Burchard. The impact of link scheduling on long paths: Statistical analysis and optimal bounds. In *2011 Proceedings IEEE INFOCOM*, pages 1242–1250. IEEE, 2011. 21

[75] M. Głabowski, S. Hanczewski, M. Stasiak, M. Weissenberg, P. Zwierzykowski, and V. Bai. Traffic modeling in industrial ethernet networks. *International Journal of Electronics and Telecommunications*, 2020. 46

[76] G. Goodwin and K. Sin. Adaptive filtering prediction and control. 1984. *Englewood Clifs: Prentice Ha lI*, 1984. 116

[77] S. Guo, D. Wu, H. Zhang, and D. Yuan. Resource modeling and scheduling for mobile edge computing: A service provider's perspective. *IEEE Access*, 6:35611–35623, 2018. 19

[78] V. Gupta, T. N. Joshi, and S. Tiwari. M/d/1 multiple vacation queueing systems with deterministic service time. *IOSR Journal of Mathematics*, 12:75–80, 2016. 56

[79] R. J. Haddad, M. P. McGarry, and P. Seeling. Video bandwidth forecasting. *Communications Surveys & Tutorials, IEEE*, 15(4):1803–1818, 2013. 160

[80] H. Hassan, J.-M. Garcia, and O. Brun. Generic modeling of multimedia traffic sources. In *3rd International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-Nets' 05), Ilkley (Great-Britain)*, 2005. 115

[81] J. Heinanen and R. Guerin. Ietf rfc 2698.". *A Single Rate Three Colour Marker*, 1999. 184

[82] J. Heinanen and R. Guerin. Rfc 2697–a single rate three color marker. *IETF, September*, 1999. 184

[83] T. R. Henderson and R. H. Katz. Transport protocols for internet-compatible satellite networks. *Selected Areas in Communications, IEEE Journal on*, 17(2):326–344, 1999. 159

[84] P. Humblet, A. Bhargava, and M. G. Hluchyj. Ballot theorems applied to the transient analysis of nd/d/1 queues. *IEEE/ACM Transactions on Networking*, 1:81–95, 1992. 28

[85] IEEE. Ieee standard for local and metropolitan area networks – bridges and bridged networks – amendment 26: Frame preemption. *IEEE Std 802.1Qbu-2016 (Amendment to IEEE Std 802.1Q-2014)*, pages 1–52, 2016. 26

[86] M. F. Iqbal, M. Zahid, D. Habib, and L. K. John. Efficient prediction of network traffic for real-time applications. *Journal of Computer Networks and Communications*, 2019, 2019. 161

[87] ITU-T. Fast access to subscriber terminals (g.fast) – power spectral density specification, 2014. 106

[88] ITU-T. Recommendation itu-t g.9701 - fast access to subscriber terminals (g.fast) - physical layer specification, 2014. 106

[89] V. B. Iversen and L. Staalhagen. Waiting time distribution in m/d/1 queueing systems. *Electronics Letters*, 35(25):2184–2185, 1999. 56

[90] J. R. Jackson. Jobshop-like queueing systems. *Management science*, 10(1):131–142, 1963. 20

[91] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM computer communication review*, volume 18, pages 314–329. ACM, 1988. 178

[92] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe. A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 1984. 150

[93] 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*, volume 3, pages 1854–1858. IEEE, 2000. 114

[94] Z. Ji, Y. Wang, W. Feng, and J. Lu. Delay-aware power and bandwidth allocation for multiuser satellite downlinks. *IEEE Communications Letters*, 18(11):1951–1954, 2014. 160

[95] P. Kansal and A. Bose. Bandwidth and latency requirements for smart transmission grid applications. *IEEE Transactions on Smart Grid*, 3(3):1344–1352, 2012. 25

[96] 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*, 8(1):66–71, 2009. 111

[97] V. Kawadia and P. Kumar. A cautionary perspective on cross-layer design. *Wireless Communications, IEEE*, 12(1):3–11, 2005. 102

[98] O. Kella and U. Yechiali. Waiting times in the non-preemptive priority m/m/c queue. *Stochastic Models*, 1(2):257–262, 1985. 20

[99] F. P. Kelly. Networks of queues with customers of different types. *Journal of applied probability*, 12(3):542–554, 1975. 20

[100] M. M. Khan. Cross-layer designs: a survey. *International Journal of Computer Applications*, 53(8), 2012. 102

[101] J. Kingman. The single server queue in heavy traffic. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 57, pages 902–904. Cambridge University Press, 1961. 174

[102] M. G. Konovalov and R. V. Razumchik. Comparison of two active queue management schemes through the m/d/1/n queue. *Informatics and Applications*, 12(4):9–15, 2018. 56

[103] R. E. Kooij, O. Østerbø, and J. Van der Wal. Calculating end-to-end queuing delay for real-time services on an ip network. In *International Workshop on Architectures for Quality of Service in the Internet*, pages 115–126. Springer, 2003. 21

[104] S. Kota, M. Goyal, R. Goyal, and R. Jain. Broadband satellite network: Tcp/ip performance analysis. In *Broadband communications*, pages 273–282. Springer, 2000. 177

[105] S. Kota, M. Goyal, R. Goyal, and R. Jain. Multimedia satellite networks and tcp/ip traffic transport. *arXiv preprint arXiv:1603.08020*, 2016. 177

[106] P. Koutsakis. Using traffic prediction and estimation of provider revenue for a joint geo satellite mac/cac scheme. *Wireless Networks*, 17(3):797–815, 2011. 160

[107] P. Koutsakis, D. Vasileiadou, and C. Stamos. Performance evaluation of the fprra framework for geo satellites in the absence of accurate multimedia traffic prediction. *International Journal on Communications Antenna and Propagation (I. Re. CAP)*, 1(1), 2011. 160

[108] A. Kumar, A. Abdelhadi, and T. C. Clancy. Delay-efficient multiclass packet scheduler. In *Design and Implementation of Practical Schedulers for M2M Uplink Networks*, pages 15–80. Springer, 2018. 118

[109] W. K. Lai and C.-L. Tang. Qos-aware downlink packet scheduling for lte networks. *Computer Networks*, 57(7):1689–1698, 2013. 144

[110] M. A. Lawal, I. Saidu, A. Mohammed, and Y. A. Sade. Downlink scheduling algorithms in lte networks: A survey. *IOSR J Mob Comput Appl*, 4(3):1–12, 2017. 112, 144, 145

[111] J.-Y. Le Boudec and P. Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*, volume 2050. Springer Science & Business Media, 2001. 41, 47, 183

[112] H. Lee. Anatomy of delay performance for the strict priority scheduling scheme in multi-service internet. *Computer Communications*, 29(1):69–76, 2005. 20

[113] 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*, pages 1664–1668. IEEE, 2007. 131

[114] L. Lei, C. Lin, and Z. Zhong. *Stochastic Petri nets for wireless networks.* Springer, 2019. 199

[115] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking (ToN)*, 2(1):1–15, 1994. 139, 159

[116] C. Li, A. Burchard, and J. Liebeherr. A network calculus with effective bandwidth. *IEEE/ACM Transactions on Networking (TON)*, 15(6):1442–1453, 2007. 21

[117] X. Lin, N. B. Shroff, and R. Srikant. A tutorial on cross-layer optimization in wireless networks. *Selected Areas in Communications, IEEE Journal on*, 24(8):1452–1463, 2006. 102

[118] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973. 26, 132

[119] 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*, pages 239–250, 2007. 121

[120] X. Liu, E. K. Chong, and N. B. Shroff. A framework for opportunistic scheduling in wireless networks. *Computer networks*, 41(4):451–474, 2003. 194

[121] W. Luo and A. Ephremides. Stability of n interacting queues in random-access systems. *IEEE Transactions on Information Theory*, 45(5):1579–1587, 1999. 123

[122] Z.-Q. Luo and S. Zhang. Dynamic spectrum management: Complexity and duality. *IEEE journal of selected topics in signal processing*, 2(1):57–73, 2008. 128

[123] Z. Ma, W. Wang, and L. Hu. Performance evaluation and analysis of a discrete queue system with multiple working vacations and non-preemptive priority. *Journal of Industrial & Management Optimization*, 16(3):1135, 2020. 20

[124] K. C. Madan et al. A non-preemptive priority queueing system with a single server serving two queues m/g/1 and m/d/1 with optional server vacations based on exhaustive service of the priority units. *Applied Mathematics*, 2(06):791, 2011. 20, 56

[125] K. C. Madan and M. F. Saleh. On m/d/1 queue with general server vacations. *International journal of information and management sciences*, 12(2):25–38, 2001. 56

[126] K. C. Madan and M. F. Saleh. On single server vacation queues with deterministic service or deterministic vacations. *Calcutta Statistical Association Bulletin*, 51(3-4):225–242, 2001. 56

[127] S. Madhu, M. B. Raju, and P. C. Reddy. A survey of cross layer design in wireless networks for joint optimization of multimedia transmission. *International Journal of Advanced Research in Computer Science*, 8(3), 2017. 102

[128] S. Mandelli, M. Andrews, S. C. Borst, and S. Klein. Satisfying network slicing constraints via 5g mac scheduling. *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 2332–2340, 2019. 194

[129] S. Mao and S. S. Panwar. A survey of envelope processes and their applications in quality of service provisioning. *IEEE Communications Surveys and Tutorials*, 8(1-4):2–20, 2006. 42

[130] S. Martiradonna, A. Grassi, G. Piro, and G. Boggia. 5g-air-simulator: An open-source tool modeling the 5g air interface. *Computer Networks*, 173:107151, 2020. 112, 148

[131] R. McEliece, J. Murphy, M. Jennings, and Z. Yu. On simplified modelling of the leaky bucket. In *Proc. Of the IEE 13th UK IEE Teletraffic Symposium, Strathclyde, UK*, pages 18–20, 1996. 48

[132] 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. 114

[133] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on networking*, 8(5):556–567, 2000. 115, 120, 121

[134] M. Mohseni, R. Zhang, and J. M. Cioffi. Optimized transmission for fading multiple-access and broadcast channels with multiple antennas. *IEEE Journal on Selected Areas in Communications*, 24(8):1627–1639, 2006. 194

[135] S. Nananukul. Multiplexing of periodic arrival processes with different packet sizes. *IEEE Transactions on Communications*, 50(7):1055–1057, 2002. 30

[136] M. M. Nasralla. A hybrid downlink scheduling approach for multi-traffic classes in lte wireless systems. *IEEE Access*, 8:82173–82186, 2020. 145

[137] 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*, 130:78–100, 2018. 112, 145

[138] 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)*, pages 1571–1575. IEEE, 2013. 114, 131, 144, 150

[139] M. J. Neely. Order optimal delay for opportunistic scheduling in multi-user wireless uplinks and downlinks. *IEEE/ACM Transactions on Networking (TON)*, 16(5):1188–1199, 2008. 116

[140] P. Odling, T. Magesacher, S. Host, P. O. Borjesson, M. Berg, and E. Areizaga. The fourth generation broadband concept. *Communications Magazine, IEEE*, 47(1):62–69, 2009. 107

[141] O. Osterbo. A discrete time queueing model for end-to-end delay and jitter analysis. In *2009 21st International Teletraffic Congress*, pages 1–8. IEEE, 2009. 21

[142] N. Pachler, J. J. G. Luis, M. Guerster, E. Crawley, and B. Cameron. Allocating power and bandwidth in multibeam satellite systems using particle swarm optimization. In *2020 IEEE Aerospace Conference*, pages 1–11. IEEE, 2020. 160

[143] D. P. Palomar and M. Chiang. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications*, 24(8):1439–1451, 2006. 102

[144] A. Paris, I. Del Portillo, B. Cameron, and E. Crawley. A genetic algorithm for joint power and bandwidth allocation in multibeam satellite systems. In *2019 IEEE Aerospace Conference*, pages 1–15. IEEE, 2019. 160

[145] M. Park. Non-preemptive fixed priority scheduling of hard real-time periodic tasks. In *International Conference on Computational Science*, pages 881–888. Springer, 2007. 26

[146] F. Pianese and P. J. Danielsen. Augmenting practical cross-layer mac schedulers via offline reinforcement learning. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–6. IEEE, 2017. 199

[147] G. Piro, L. A. Grieco, G. Boggia, F. Capozzi, and P. Camarda. Simulating lte cellular systems: An open-source framework. *IEEE transactions on vehicular technology*, 60(2):498–513, 2010. 148

[148] G. Piro, L. A. Grieco, G. Boggia, R. Fortuna, and P. Camarda. Two-level downlink scheduling for real-time multimedia services in lte networks. *IEEE Transactions on Multimedia*, 13(5):1052–1065, 2011. 144, 150

[149] M. J. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis*, pages 51–67. Springer, 1994. 187

[150] J. Qiu and E. W. Knightly. Measurement-based admission control with aggregate traffic envelopes. *IEEE/ACM Transactions on Networking (TON)*, 9(2):199–210, 2001. 35, 163, 164, 182

[151] 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.*, volume 1, pages 462–466. IEEE, 2003. 131, 144, 188

[152] V. J. Ribeiro, R. H. Riedi, M. S. Crouse, and R. G. Baraniuk. Simulation of nongaussian long-range-dependent traffic using wavelets. *ACM SIGMETRICS Performance Evaluation Review*, 27(1):1–12, 1999. 25

[153] F. Ridouard, J.-L. Scharbarg, and C. Fraboul. Probabilistic upper bounds for heterogeneous flows using a static priority queueing on an afdx network. In *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, pages 1220–1227. IEEE, 2008. 22

[154] J. W. Roberts and J. T. Virtamo. The superposition of periodic cell arrival streams in an atm multiplexer. *Communications, IEEE Transactions on*, 39(2):298–303, 1991. 28, 29, 30, 37

[155] O. Rose. Mpeg traces archive. `http://web.archive.org/web/20080916125231/http://www-info3.informatik.uni-wuerzburg.de/mpeg/traces/`. Accessed: 2020-02-27. 141

[156] O. Rose. University of wuerzburg, index of mpeg traces. `ftp-info3.informatik.uni-wuerzburg.de/pub/MPEG/`. Accessed 25-10-2016. 170

[157] 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*, volume 4, pages 2779–2785. IEEE, 2005. 111

[158] 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)*, pages 1097–1102. IEEE, 2009. 120

[159] B. Sadiq, R. Madan, and A. Sampath. Downlink scheduling for multiclass traffic in lte. *EURASIP Journal on Wireless Communications and Networking*, 2009:1–18, 2009. 144

[160] H. Schioeler, H.-P. Schwefel, and M. B. Hansen. Cync: a matlab/simulink toolbox for network calculus. In *ValueTools*, page 60. Citeseer, 2007. 47

[161] 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)*, pages 585–588. IEEE, 2019. 111

[162] S. Shakkottai and A. L. Stolyar. Scheduling algorithms for a mixture of real-time and non-real-time data in hdr. In *Teletraffic Science and Engineering*, volume 4, pages 793–804. Elsevier, 2001. 194

[163] S. Shakkottai and A. L. Stolyar. Scheduling for multiple flows sharing a time-varying channel: The exponential rule. *Translations of the American Mathematical Society-Series 2*, 207:185–202, 2002. 150

[164] A. Sharifian. *Utility-based Packet Scheduling and Resource Allocation Algorithms with Heterogeneous Trac for Wireless OFDMA Networks*. PhD thesis, Carleton University Ottawa, 2014. 111, 123

[165] P. Silverman. Techniques to mitigate uncancelled crosstalk on vectored vdsl2 lines. In *Broadband Forum Technical Report: TR-320*, 2014. 128

[166] J. Sing and B. Soh. Tcp new vegas: improving the performance of tcp vegas over high latency links. In *Network Computing and Applications, Fourth IEEE International Symposium on*, pages 73–82. IEEE, 2005. 177

[167] E. Skondras, A. Michalas, A. Sgora, and D. D. Vergados. A downlink scheduler supporting real time services in lte cellular networks. In *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–6. IEEE, 2015. 148

[168] A. Sleptchenko, J. Selen, I. Adan, and G.-J. van Houtum. Joint queue length distribution of multi-class, single-server queues with preemptive priorities. *Queueing Systems*, 81(4):379–395, 2015. 20

[169] G. Song. *Cross-layer resource allocation and scheduling in wireless multicarrier networks*. PhD thesis, Citeseer, 2005. 124, 194

[170] 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*, 57(7):2109–2121, 2009. 111, 114, 188

[171] D. Staehle, K. Leibnitz, and P. Tran-Gia. *Source traffic modeling of wireless applications*. Inst. für Informatik, 2000. 46

[172] Steven G. Johnson. The nlopt nonlinear-optimization package. 187

[173] D. Stiliadis and A. Varma. Latency-rate servers: a general model for analysis of traffic scheduling algorithms. *IEEE/ACM Transactions on Networking (ToN)*, 6(5):611–624, 1998. 184

[174] X. Sun, Q. Zhang, X. Xin, and C. Yu. Cross-layer dynamic bandwidth allocation based on fairness and system utility. In *2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pages 1322–1325. IEEE, 2012. 161

[175] M. Swarna, S. Ravi, and M. Anand. Leaky bucket algorithm for congestion control. *International Journal of Applied Engineering Research*, 11(5):3155–3159, 2016. 47

[176] Y. Takahashi and O. Hashida. Delay analysis of discrete-time priority queue with structured inputs. *Queueing Systems*, 8(1):149–163, 1991. 19

[177] P. P. Tang and T.-Y. Tai. Network traffic characterization using token bucket model. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 51–62. IEEE, 1999. 48, 183

[178] J. Tanner. A derivation of the borel distribution. *Biometrika*, 48(1/2):222–224, 1961. 56

[179] M. S. Taqqu, W. Willinger, and R. Sherman. Proof of a fundamental result in self-similar traffic modeling. *ACM SIGCOMM Computer Communication Review*, 27(2):5–23, 1997. 139, 170

[180] 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*, 37(12):1936–1948, 1992. 102, 111, 120, 188, 194

[181] TNO. G.fast: Release of measured transfer characteristics of the 104m KPN access cable. Technical report, ITU - Telecommunication Standardization Sector, Mar. 2013. 129

[182] J. S. Turner. New directions in communications (or which way to the information age?). *IEEE Communications Magazine*, 40(5):50–57, 2002. 47

[183] J. Van den Eynde and C. Blondia. Cross-layer optimization with real-time adaptive dynamic spectrum management for fourth generation broadband access networks. In *IFIP International Conference on Autonomous Infrastructure, Management and Security*, pages 184–188. Springer, 2014. 159

[184] J. Van den Eynde, J. Verdyck, C. Blondia, and M. Moonen. Minimal delay violation-based cross-layer scheduler and resource allocation for DSL networks. *IEEE Access*, 9:75905–75922, 2021. 129

[185] 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*, pages 1–6. IEEE, 2017. 194

[186] L. A. van Vianen, A. F. Gabor, and J.-K. van Ommeren. Waiting times in classical priority queues via elementary lattice path counting. *Queueing systems*, 84(3):295–307, 2016. 20

[187] 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)*, pages 787–791. IEEE, 2018. 128, 129

[188] J. Verdyck and M. Moonen. Dynamic spectrum management in digital subscriber line networks with unequal error protection requirements. *IEEE Access*, 5:18107–18120, 2017. 118

[189] J. T. Virtamo. Idle and busy period distributions of an infinite capacity n* d/d/1 queue. In *Teletraffic Science and Engineering*, volume 1, pages 453–459. Elsevier, 1994. 35, 37

[190] K. V. Vishwanath and A. Vahdat. Realistic and responsive network traffic generation. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 111–122, 2006. 115

[191] J. Walraevens, H. Bruneel, D. Fiems, and S. Wittevrongel. Delay analysis of multiclass queues with correlated train arrivals and a hybrid priority/fifo scheduling discipline. *Applied Mathematical Modelling*, 45:823–839, 2017. 20

[192] J. Walraevens, D. Fiems, S. Wittevrongel, and H. Bruneel. Calculation of output characteristics of a priority queue through a busy period analysis. *European Journal of Operational Research*, 198(3):891–898, 2009. 20

[193] J. Walraevens, B. Steyaert, M. Moeneclaey, and H. Bruneel. Delay analysis of a hol priority queue. *Telecommunication Systems*, 30(1):81–98, 2005. 19

[194] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang. Machine learning for networking: Workflow, advances and opportunities. *Ieee Network*, 32(2):92–99, 2017. 198

[195] 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*, 67(1):503–517, 2018. 110, 112

[196] X. Wang and N. Gao. Stochastic resource allocation over fading multiple access and broadcast channels. *IEEE Transactions on Information Theory*, 56(5):2382–2391, 2010. 194

[197] 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*, pages 1–6. IEEE, 2020. 110, 111

[198] 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*, 49(1):161–178, 2020. 110, 112, 114, 131

[199] 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)*, pages 757–761. IEEE, 2018. 114, 131, 150

[200] S. L. Yadav and M. Phogat. A survey on cross layer design implementation in wireless sensor networks. *International Journal of Science, Engineering and Computer Technology*, 7(1):17–19, 2017. 102

[201] H. Yao, J. McLamb, M. Mustafa, A. Narula-Tam, and N. Yazdani. Dynamic resource allocation dama alternatives study for satellite communications systems. In *Military Communications Conference, 2009. MILCOM 2009. IEEE*, pages 1–7. IEEE, 2009. 160

[202] 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)*, 10(2), 2020. 114, 131

[203] S. M. Zahedi, S. Fan, and B. C. Lee. Managing heterogeneous datacenters with tokens. *ACM Transactions on Architecture and Code Optimization*, 15(2):18, 2018. 194

[204] E. Zhang and L. Xu. Capacity and token rate estimation for networks with token bucket shapers. *Computer Networks*, 88:1–11, 2015. 184

[205] G.-x. Zhao, Y. Chen, and X.-d. Xue. M/m/1 queue under nonpreemptive priority. *College Mathematics*, 22(1):44–48, 2006. 20

[206] W. Zhou, Q. Zhang, Q. Tian, X. Xin, B. Liu, L. Zhang, F. Tian, Y. Tao, Y. Shen, D. Chen, et al. Cross-layer dynamic bandwidth allocation algorithm based on convex optimization theory in satellite communication system. In *2017 16th International Conference on Optical Communications and Networks (ICOCN)*, pages 1–3. IEEE, 2017. 161

[207] R. Zhu and J. Yang. Buffer-aware adaptive resource allocation scheme in lte transmission systems. *EURASIP Journal on Wireless Communications and Networking*, 2015(1):1–16, 2015. 144

[208] P. Zuo, T. Peng, W. Linghu, and W. Wang. Resource allocation for cognitive satellite communications downlink. *IEEE Access*, 6:75192–75205, 2018. 160