Reinhard German
Kai-Steffen Hielscher
Udo R. Krieger (Eds.)

LNCS 10740

Measurement, Modelling and Evaluation of Computing Systems

19th International GI/ITG Conference, MMB 2018
Erlangen, Germany, February 26–28, 2018
Proceedings

Lecture Notes in Computer Science **10740**

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

More information about this series at http://www.springer.com/series/7408

Reinhard German · Kai-Steffen Hielscher
Udo R. Krieger (Eds.)

Measurement, Modelling and Evaluation of Computing Systems

19th International GI/ITG Conference, MMB 2018
Erlangen, Germany, February 26–28, 2018
Proceedings

Springer

Editors
Reinhard German
University of Erlangen-Nuremberg
Erlangen
Germany

Kai-Steffen Hielscher
University of Erlangen-Nuremberg
Erlangen
Germany

Udo R. Krieger
Otto-Friedrich-Universität Bamberg
Bamberg
Germany

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-74946-4 ISBN 978-3-319-74947-1 (eBook)
https://doi.org/10.1007/978-3-319-74947-1

Library of Congress Control Number: 2018930746

LNCS Sublibrary: SL2 – Programming and Software Engineering

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG
part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

As conference chairs, it is our pleasure to present this LNCS volume with its contributions on performance and dependability evaluation techniques for computer and communication systems and their related fields. The papers were presented at the 19th International GI/ITG Conference on Measurement, Modelling and Evaluation of Computing Systems (MMB 2018), held during February 26–28, 2018, at Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) in Germany.

The biennial MMB conference started in the early 1980s and has seen a significant broadening in scope but has also kept its essence: measurements, stochastic modeling, analysis, and simulation applied to computer and communication systems. From the methodological perspective, new techniques such a network calculus have been included and, besides performance, dependability and security have been added. From an application perspective, the traditional area of networking has been supplemented by many additional domains such as smart energy systems, smart mobility, social networks, and others.

Today we are faced with complex interconnected systems in which information and communication technology always plays the key role as nerve tissue. This is the case, for instance, regarding the Internet of Things, cyber-physical systems, future 5G mobile communication systems, software-defined networking, smart energy systems, connected mobility systems, and countless more examples. MMB provides a means for assessing important, mostly quantitative system aspects such as the performance, dependability, and also security of these systems that is ultimately needed for their proper design.

The technical program was composed by the Program Committee in a thorough review procedure involving at least three reviewers and after a careful selection process during a physical meeting. In all, 16 full papers were selected representing very well the broad spectrum of methodological and applied work. A Special Session on Software-Defined Networking, organized by Ognjen Dobrijevic, University of Zagreb, Croatia, and Thomas Zinner, Universität Würzburg, Germany, was included in the program and covered a very relevant topic in current networking. The program was framed by three distinguished invited speakers, giving insights into in major application fields:

1. Performance Optimization of 5G Mobile Networks by Prof. Hans van den Berg, TNO, The Netherlands, who is also affiliated with Twente University and the Centre for Mathematics and Computer Science in Amsterdam, The Netherlands, and a recipient of ITC's Arne Jensen Lifetime Award 2017 for his contributions on performance modeling and analysis
2. Future Energy Grids – Challenges and Potential for ICT by Prof. Hartmut Schmeck, Full Professor of Applied Informatics at Karlsruhe Institute of Technology, who is additionally director of the FZI Research Center for Information Technology, a shaper of the new discipline "energy informatics," and a recipient of the Heinrich-Hertz Prize 2016 from EnBW foundation

3. Autonomous Driving – The "Uncrashable" Car? What It Takes to Make Self-Driving Vehicles Safe and Reliable Traffic Participants by Dr. Frank Keck, CEO of ZF Zukunft Mobility GmbH, a company of ZF Friedrichshafen AG, and former CEO of Automotive Safety Technologies GmbH

All three talks provided insights into the latest technological trends like 5G mobile communications with new radio, "softwarization," and network slicing that include many relevant MMB topics. In particular, the energy transition with smart grids and smart markets to balance fluctuating supply and demand seems to be made for the MMB community. Finally, autonomous and connected cars provide a high potential for simulation as well as analytic performance and dependability evaluation during the design process.

The technical program additionally offered nine papers about software tools that were demonstrated during the conference. As a new element, industrial, practical experience and PhD track papers were included: one industrial paper, one practical experience report, and two PhD track papers.

As in previous MMB conferences, two satellite workshops were organized covering highly relevant research topics:

- 4th Workshop on Network Calculus (WoNeCa-4)
- Second International Workshop on Modeling, Analysis, and Management of Social Networks and Their Applications (SOCNET 2018)

At the beginning of the conference, three tutorials were presented:

- A Modern Perspective on Fault Tree Analysis, by Joost-Pieter Katoen and Matthias Volk, RWTH Aachen
- IoT — From Praxis to Theory by Florian Metzger and Tobias Hoßfeld, Universität Duisburg-Essen
- Data Analysis of Measurements with Immanent Dependencies and Heavy-Tailed Characteristics, by Natalia M. Markovich, Russian Academy of Sciences, and Udo Krieger, Universität Bamberg

As conference chairs, we express our gratitude to all members of the Program Committee and all external reviewers for their dedicated service, maintaining the quality objectives of the conference, and for the timely provision of their valuable reviews. We express our sincere appreciation to FAU Erlangen-Nürnberg as the conference host, as well as to all members of the local Organizing Committee of MMB 2018 for their great efforts devoted to the success of the conference. We thank all the authors for their submitted contributions, all the speakers for their lively presentations, and all the participants for their contributions to interesting discussions. Finally, it is our hope that readers will enjoy these MMB 2018 proceedings and use them for their future research.

February 2018

Reinhard German
Kai-Steffen Hielscher
Udo Krieger

Organization

MMB 2018 was jointly organized by the German Gesellschaft für Informatik (GI) and Informationstechnische Gesellschaft im VDE (ITG), Technical Committees on Measurement, Modelling and Evaluation of Computing Systems (MMB) in cooperation with the Department of Computer Science 7 (Computer Networks and Communication Systems) of Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany.

Executive Committee

Conference Chair

Reinhard German

Local Organization Chairs

Kai-Steffen Hielscher
Marco Pruckner

Tools Chair

Markus Siegle

Publication Chair

Udo Krieger

Web Chair

Anatoli Djanatliev

Program Committee

Robert Basmadjian	Universität Passau, Germany
Marcel Baunach	TU Graz, Austria
Peter Buchholz	TU Dortmund, Germany
Hans Daduna	Universität Hamburg, Germany
Markus Fidler	Leibnitz Universität Hannover, Germany
Anna Förster	Universität Bremen, Germany
Reinhard German	FAU Erlangen-Nürnberg, Germany
Gerhard Hasslinger	Deutsche Telekom AG, Germany
Boudewijn Haverkort	University of Twente, The Netherlands
Tobias Hoßfeld	Universität Duisburg-Essen, Germany
Holger Hermanns	Universität des Saarlands, Germany

Joost-Pieter Katoen	RWTH Aachen, Germany
Peter Kemper	The College of William and Mary, USA
Samuel Kounev	Universität Würzburg, Germany
Udo Krieger	Otto-Friedrich-Universität Bamberg, Germany
Kai Lampka	Elektrobit Automotive, Germany
Wolfram Lautenschläger	Nokia Bell Labs, Germany
Hermann de Meer	Universität Passau, Germany
Michael Menth	Universität Tübingen, Germany
Peter Reichl	Universität Wien, Austria
Anne Remke	Universität Münster, Germany
Johannes Riedl	Siemens AG, Germany
Oliver Rose	Universität der Bundeswehr München, Germany
Ramin Sadre	Université Catholique de Louvain, Belgium
Jens Schmitt	TU Kaiserslautern, Germany
Markus Siegle	Universität der Bundeswehr München, Germany
Helena Szczerbicka	Leibniz Universität Hannover, Germany
Andreas Timm-Giel	TU Hamburg, Germany
Dietmar Tutsch	Universität Wuppertal, Germany
Kurt Tutschku	BTH Karlskrona, Sweden
Oliver Waldhorst	Hochschule Karlsruhe, Germany
Max Walter	Siemens AG, Germany
Verena Wolf	Universität des Saarlands, Germany
Bernd Wolfinger	Universität Hamburg, Germany
Katinka Wolter	Freie Universität Berlin, Germany
Armin Zimmermann	TU Ilmenau, Germany

Industrial Track Chair

Thomas Herpel	ZF Zukunft Mobility, Germany

Special SDN/NFV Session Organizers

Ognjen Dobrijevic	University of Zagreb, Croatia
Thomas Zinner	Universität Würzburg, Germany

Additional Reviewers

Michael Backenköhler	Universität des Saarlands, Germany
Robert Bauer	Karlsruhe Institute of Technology, Germany
Peter Bazan	FAU Erlangen-Nürnberg, Germany
Alexander Beifuß	Universität Hamburg, Germany
Andreas Blenk	TU München, Germany
Simon Eismann	Universität Würzburg, Germany
Andreas Fischer	Technische Hochschule Deggendorf, Germany
Marija Furdek	KTH Royal Institute of Technology, Sweden
Nicholas Gray	Universität Würzburg, Germany

Tomislav Grgic	Ericsson Nikola Tesla, Croatia
Frederik Hauser	Universität Tübingen, Germany
Florian Heimgärtner	Universität Tübingen, Germany
Stefan Herrnleben	Universität Würzburg, Germany
Michael Jarschel	Nokia Bell Labs, Germany
Marijn Jongerden	University of Twente, The Netherlands
Andreas Kassler	Karlstads Universitet, Sweden
Stanislav Lange	Universität Würzburg, Germany
Anna Lukina	TU Wien, Austria
Florian Metzger	Universität Duisburg-Essen, Germany
Stefano Petrangeli	Universiteit Gent, Belgium
Björn F. Postema	University of Twente, The Netherlands
Marco Pruckner	FAU Erlangen-Nürnberg, Germany
Mark Schmidt	Universität Tübingen, Germany
Norbert Schmitt	Universität Würzburg, Germany
Andreas Stockmayer	Universität Tübingen, Germany

Abstracts of Invited Talks

Performance Optimization of 5G Mobile Networks

Hans van den Berg

TNO, University of Twente, CWI Amsterdam,
Anna van Buerenplein 1, 2595 DA The Hague, The Netherlands
J.L.vandenBerg@tno.nl

Abstract. Research on 5G in Europe is boosted by the 5G PPP consortium consisting of network vendors and operators, system integrators and academia and other research institutes, working closely together with companies from important vertical industries. 5G aims at bringing new, distinctive network and service capabilities fulfilling the needs of the future Internet of Things (IoT). As such it should sustain enormous data volumes and support critical, highly demanding communication services for e.g. self-driving cars, robotics in smart industry, and mobile virtual reality applications. Network 'softwarization' through emerging technologies as Software Defined Networking (SDN) and Network Functions Virtualization (NFV) is introduced to provide the flexibility needed to reach the required performance and scalability targets in an efficient way. However, to actually achieve the full potential of future 5G networks huge challenges regarding network management and performance optimization are faced. Big data techniques exploiting data coming from network devices in forms of e.g. device logs and usage histories provide a promising direction to address these challenges. In the talk we will briefly sketch the 5G PPP ambitions, review the aforementioned research challenges and present (ongoing) work on some specific 5G network performance optimization problems.

Future Energy Grids – Challenges and Potential for ICT

Hartmut Schmeck

Institute AIFB, Karlsruhe Institute of Technology (KIT),
76128 Karlsruhe, Germany
hartmut.schmeck@kit.edu

Abstract. The energy system is one of the most critical infrastructures of our world. The reliable supply of energy is essential for the adequate operation of almost any process in our private and professional life. Society and industry would suffer enormously, if the steady balance between demand and supply could not be guaranteed. The current transition towards energy from renewable sources is having tremendous effects on this well-established infrastructure. In particular, the restricted capabilities of controlling the supply of electricity from weather-dependent energy sources leads to the need for an essential change in one of the basic principles of the electric power system, which means that it will no longer be feasible to let the power supply follow the demand but there will be a strong need to let the demand follow the supply. This can only be achieved by discovering and exploiting the potential of flexibility of demand and supply in the best possible way. The talk will illustrate how the major challenges of the ongoing energy transition create the need for an adequately designed energy information and control network with distributed intelligence. A fundamental task in the design of this network consists of making the necessary information available to the locations where operating and control decisions have to be taken and to provide appropriate methodology for managing tomorrow's energy system in the most efficient and most reliable way. In particular, an assessment of the potential contribution of flexibility in demand and supply to guaranteeing the necessary stability and resilience needs appropriate modelling and simulation, based on effective strategies for measuring the current status and behaviour of relevant grid components.

Autonomous Driving – the "Uncrashable" Car?

What It Takes to Make Self-Driving Vehicles Safe and Reliable Traffic Participants

Frank Keck

ZF Zukunft Mobility GmbH, Ruppertswies 14, 85092 Kösching, Germany

Abstract. Autonomous driving is in the spotlight of both scientific research and industrial development. With worldwide constantly growing traffic volumes, the challenging task in putting self-driving vehicles onto the street is to cross the chasm between high system availability and low to zero malfunction rates, even in dense traffic and complex situations on the road. Developing software functions for driver assistance and vehicle safety for autonomously driving cars requires the traditional developments processes and methods to be revised. In this talk, a novel and promising development approach is presented. The combination of use case based requirement specification, algorithm development with machine learning techniques and both simulation based and real-life testing yields an agile yet sound software development framework for autonomous driving functions. Additionally, some thought-provoking impulses are given on how to achieve a high level of system reliability by exploiting the capabilities of virtualization at early development stages.

Contents

Full Papers

Full Papers

Time-Based Maintenance Models Under Uncertainty

Peter Buchholz, Iryna Dohndorf(✉), and Dimitri Scheftelowitsch

Informatik IV, TU Dortmund, Dortmund, Germany
{peter.buchholz,iryna.dohndorf,dimitri.scheftelowitsch}@cs.tu-dortmund.de

Abstract. Model based computation of optimal maintenance strategies is one of the classical applications of Markov Decision Processes. Unfortunately, a Markov Decision Process often does not capture the behavior of a component or system of components correctly because the duration of different operational phases is not exponentially distributed and the status of component is often only partially observable during operational times. The paper presents a general model for components with partially observable states and non-exponential failure, maintenance and repair times which are modeled by phase type distributions. Optimal maintenance strategies are computed using Markov decision theory. However, since the internal state of a component is not completely known, only bounds for the parameters of a Markov decision process can be computed resulting in a bounded parameters Markov decision process. For this kind of process optimal strategies can be computed assuming best, worst or average case behavior.

Keywords: Maintenance models · Markov decision processes
Stochastic dynamic programming · Numerical methods

1 Introduction

Maintenance problems appear in various aspects of human life wherever there exists a *system* or *component* that degrades with time and has to be kept in an operational condition by means of *active maintenance*, i. e., a conscious effort to inspect and revert the effects of degradation. In general one can distinguish between *condition-* or *event*-based maintenance and *preventive-* or *time*-based maintenance [11]. In the former case, which is more often analyzed, maintenance takes places whenever a specific condition or event is observed in a system which indicates a degradation of the system. In preventive or time-based maintenance, the system is inspected at predefined time points and based on the results of inspection it is decided whether a maintenance operation is performed or not. Usually this decision is based on incomplete information about the system state because the degradation of most systems can only be partially observed or tested [16]. This implies that maintenance decisions have to be made under uncertainty about the current state and future behavior of the system.

A popular approach to model this kind of problems are Markov decision processes (MDPs) [6,15,19]. In it, the stages of degradation are modeled as individual *states* of the system, the maintenance options as possible *actions* of the

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 3–18, 2018.
https://doi.org/10.1007/978-3-319-74947-1_1

controller, the results of actions as a probabilistic transition mechanism between the states, and the degradation itself is modeled as a *reward function*. The optimal policy computed from the MDP describes the optimal maintenance strategy. Since failures occur at random times, they are usually described by continuous time models. Fixed inspection times introduce a time-step in the model which implies that a discrete time model is more appropriate for the computation of optimal policies. Thus, we consider discrete time MDPs (DTMDPs) [19] resulting from the embedding of a continuous time MDP (CTMDP) to compute optimal time-based maintenance strategies.

The basis of Markov models is the memoryless property which implies that the time to the next event, in our case next degradation of the component or system state, is independent of the past. This means that failure or degradation times are exponentially distributed which is usually not the case for real systems. It is known that Weibull or log-normal distributions are much more realistic models for the behavior of systems. Consequently, MDPs are only an approximate model because the future behavior depends on additional non-observable states of the system. To express this uncertainty in the model, one has to extend the model class to partially observable MPDs (POMDPs) or MDPs with uncertain parameters. The price for such an extension is a more complex model and the need to compute bounds rather than exact results.

In our work, we combine in some sense POMDPs and MDPs with uncertain parameters in order to provide a framework for maintenance models of systems with several components. Prior to the introduction of the approach, we give in Sect. 2 a brief overview of related work. Then, in Sect. 3, we model failure time distributions by embedded Markov processes to build a realistic model of component aging. In Sect. 4, we apply knowledge about uncertainty in Markov decision processes to derive a decision model and reduce the state space and make computing optimal maintenance policies tractable even for more complex models. Finally, we present simple examples in Sect. 5 and discuss the results in Sect. 6. The proofs for the main theorems used in the paper can be found online [1].

2 Related Work

An enormous number of papers on MDPs and their use in maintenance modeling exists. Therefore we mention only briefly the main references.

Markov decision processes and, briefly, their application to maintenance and dependability have been thoroughly analyzed in the textbook by Puterman [19]. A more elaborate discussion of Markov models for maintenance problems can be found in [11,15]; for specific applications in this context, we refer to [3,6,12,16]. Probability distributions of the phase type and their application to model failure and repair times have been considered in [5,13]. More complex maintenance models based on phase-type distributions can be found in [2,7].

The extension of Markov decision processes with interval-type transition probabilities has been discussed in [14]. Maintenance applications of a related uncertain model can be found in [18].

3 Markov Models for Maintenance

We consider dependability models that are described by Markov processes. In the following first the basic model of a component and then the composition of components is introduced.

Component Models. The behavior of a component can be described by a stochastic process with $2N + 1$ phases, which are divided in three groups: N *operational* phases, N *maintenance* phases and one *failure* phase. We use the term *phase* rather than *state* because states will later be used to model non-exponential durations of phases. Operational phases are numbered 1 through N, maintenance phases receive numbers $N + 1$ through $2N$ and the failure phase number 0.

A new component starts in operational phase N and then continuously degrades by decreasing the operational phase or by a hard failure that immediately brings the component down to the failure phase. Let T_i be the random variable describing the duration of phase i and p_i the probability of entering from operational phase i immediately failure phase 0. Consequently, $\overline{p}_i = 1 - p_i$ is the probability of entering operational phase $i - 1$ from i and $p_1 = 1$.

Eventually a component will end in phase 0, the failure phase, if no action is taken. In the failure phase, the component may be repaired or substituted by a new component, in both cases the component starts again in operational phase N. Additionally, maintenance operations may be introduced. Maintenance can be initiated at inspection times $l \cdot \Delta$ ($l \in \mathbb{N}$) when the phase of the component is determined. If a maintenance operation is initiated when the component is in operational phase i, then the phase changes to $N + i$. In phase $N + i$ ($i \in \{1, \ldots, N\}$) maintenance is performed, which requires time T_{N+i}. Afterwards the phase changes with probability $h_{N+i,j}$ to operational phase $j \in \{i, \ldots, N\}$, i.e., we assume that maintenance cannot deteriorate the component.

Figure 1 shows the state transition diagram of a component. Solid arcs describe transitions that occur if nothing is done and dashed arcs describe transitions that take place if a maintenance operation is initiated. The above model is similar to the classical models for maintenance [8,19]. To analyze the behavior and to determine optimal decisions, quantitative measures in the form of rewards

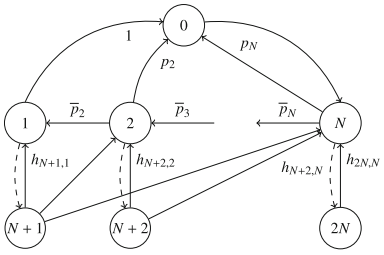


Fig. 1. Phase transition diagram of a component.

have to be associated with phases or transitions. We assume that in operational phase i, the component gains reward r_i in phase i and additionally a penalty $r_0 < 0$ has to be paid whenever the failure phase is entered.

Sojourn times in the phases are characterized by independently distributed random variables with cumulative distribution function $F_i(t)$ and probability density $f_i(t)$ for phase i. The time-dependent rate out of a phase is then given by

$$\lambda_i(t) = \frac{f_i(t)}{1 - F_i(t)}. \tag{1}$$

For the phases 1 through N this quantity describes the rate of deterioration in the operational phase. For modeling failure times, standard distributions like Weibull, Gamma or log-normal or phase type distributions are applied [5,13]. For repair or maintenance times less information is available in the literature but it seems that also in these cases phase type distributions are appropriate for modeling the durations [2,7]. In this paper, we use acyclic phase type distributions (APHs), a subclass of phase type distributions, which are described by an absorbing Markov chain with m transient states, initial vector $\boldsymbol{\nu}$ and generator matrix $\boldsymbol{D}$ that can be reordered to an upper triangular matrix. As shown for example in [5,9] general phase type distributions as well as standard failure time distributions like the Weibull distribution can be approximated sufficiently accurate by APHs. One of the advantages of APHs is the availability of a canonical representation developed in [10]. In the canonical representation matrix $\boldsymbol{D}$ has the form

$$\boldsymbol{D} = \begin{pmatrix} -\mu_1 & \mu_1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & -\mu_{m-1} & \mu_{m-1} \\ 0 & \cdots & \cdots & 0 & -\mu_m \end{pmatrix} \tag{2}$$

with $\mu_i \leq \mu_{i+1}$ for $1 \leq i < m-1$. This implies that the distribution has only one exit state, namely the last state. The representation $(\boldsymbol{\nu}, \boldsymbol{D})$ characterizes the APH and

$$f(t) = \boldsymbol{\nu} e^{t\boldsymbol{D}} \left(-\boldsymbol{D}\mathbb{1}\right), \quad F(t) = 1 - \boldsymbol{\nu} e^{t\boldsymbol{D}} \mathbb{1}, \quad \lambda(t) = \frac{\boldsymbol{\nu} e^{t\boldsymbol{D}} \left(-\boldsymbol{D}\,\mathbb{1}\right)}{\boldsymbol{\nu} e^{t\boldsymbol{D}}\,\mathbb{1}}, \tag{3}$$

where $\mathbb{1}$ is a column vector of 1 of appropriate length. For a time-dependent rate $\lambda(t)$, the bounds $\lambda^-_{t^-,t^+} = \min_{t^+ \geq t \geq t^-} \lambda(t)$ and $\lambda^+_{t^-,t^+} = \max_{t^+ \geq t \geq t^-} \lambda(t)$ with $t^- \leq t^+$ can be computed. We use the notation $\lambda^- = \lambda^-_{0,\infty}$ and $\lambda^+ = \lambda^+_{0,\infty}$. The average rate is given by $\lambda^{av} = -\left(\boldsymbol{\nu}\boldsymbol{D}^{-1}\mathbb{1}\right)^{-1}$. If the APH represents an exponential distribution, then $\lambda^{av} = \lambda^- = \lambda^+$.

We assume that the sojourn time in phase i is characterized by an APH $(\boldsymbol{\nu}_i, \boldsymbol{D}_i)$ of order m_i and define $\boldsymbol{d}_i = -\boldsymbol{D}_i\mathbb{1} = (0, \ldots, 0, \mu_{m_i})^T$, for $i \in \{0, \ldots, 2N\}$. This implies that all times are phase type distributed according to an APH. Thus, the state of a component is defined by the observable phase of

the component and the non-observable state of the APH defining the duration of the phase. The detailed stochastic process describing a component is defined in Sect. 4.

Composition of Components. If several components are considered, they run in parallel. As long as they are independent, components can be analyzed in isolation and rewards are simply added. However, usually components are dependent. We consider here the case that rewards depend on the joint phase. E.g., a system with two components is no longer operating if both components are in maintenance or in the failure phase. Thus, a penalty has to be paid, if this situation occurs. Additionally, components may interact due to limited capacity for maintenance. In this case, a maintenance operation can only be started if capacity for maintenance is available, otherwise the component has to continue working until the next decision point for maintenance.

We restrict the description and analysis to systems built from two components to avoid an extensive and complex notation. However, the general approach can be extended to more than two components following the presented ideas.

4 Decision Models

The models described above implicitly include decisions to perform a maintenance operation or let the system run. It is now shown that these decisions can be modeled adequately by a Markov Decision Process (MDP) [19], if only the average behavior is analyzed. In the general case, it can be modeled as a Bounded Parameter Markov Decision Process (BMDP) [14]. From these processes optimal maintenance strategies are computed. We consider in the following first single components and then models composed of two components.

For notational convenience we define the following abbreviations $m_{i:j} = \sum_{h=i}^{j} m_h$ if $i \leq j$ and 0 otherwise, $\boldsymbol{e}_{i,n}$ $(i \leq n)$ for the ith unit row vector of length n, $\mathbf{0}_m$ for a zero row vector of length m, $\mathbf{0}_{n,m}$ for a zero matrix of order $n \times m$ and $\boldsymbol{I}_m$ for an identity matrix of order m. If the order of vectors or matrices follows from the context, we avoid the use of prefixes. We assume that the indices of matrices and vectors start with 0 and end with $n-1$ for a vector or matrix of dimension n.

4.1 Decision Model for a Component

For maintenance the component is observed at discrete time points $l \cdot \Delta$ $(l = 0, 1, 2, \ldots)$ for some time step $\Delta > 0$ and maintenance is possibly initiated at those time points. At time $l \cdot \Delta$ we can observe the phase but not the internal state which in some sense describes the internal wear of the component. Consequently, decisions have to be based on the phase which implies that we build a decision process with $2N + 1$ states. However, since the behavior of this process depends on the unknown internal state at a decision point, we consider the worst, best and average behavior. The former two define a BMDP [14], the latter

an MDP. Before the decision process can be generated, the internal behavior of the component between decision points has to be characterized. Without maintenance the component can be described by a Markov chain with $m_{0:2N}$ states and generator matrix $\boldsymbol{Q}$.

Matrix $\boldsymbol{Q}$ can be built from the phase type distributions and the different probabilities of the degradation process. Let $\boldsymbol{F}$ be a $m_{1:N} \times m_{1:N}$ matrix with $N \times N$ submatrices $\boldsymbol{F}_{i,j}$ $(1 \leq i, j \leq N)$ of size $m_i \times m_j$.

$$\boldsymbol{F}_{i,j} = \begin{cases} \boldsymbol{D}_i & \text{if } j = i, \\ \overline{p}_i \boldsymbol{d}_i \boldsymbol{\nu}_{i-1} & \text{if } j = i-1, \\ \boldsymbol{0} & \text{otherwise.} \end{cases}$$

Matrix $\boldsymbol{F}$ includes all transitions between operational phases. Matrix $\boldsymbol{D}_{N+1:2N}$ is a $m_{N+1:2N} \times m_{N+1:2N}$ matrix with diagonal blocks $\boldsymbol{D}_i$ $(i = N+1, \ldots, 2N)$, it contains all transitions between maintenance phases. To describe transitions between maintenance, operational and failure phases define the following matrices

$$\boldsymbol{E}_0 = \left(\boldsymbol{0}_{m_0, m_{1:N-1}} \; \boldsymbol{d}_0 \boldsymbol{\nu}_N \right), \; \boldsymbol{E}_1 = \begin{pmatrix} p_1 \boldsymbol{d}_1 \boldsymbol{\nu}_0 \\ \vdots \\ p_N \boldsymbol{d}_N \boldsymbol{\nu}_0 \end{pmatrix},$$

and

$$\boldsymbol{E}_2 = \begin{pmatrix} h_{N+1,1} \boldsymbol{d}_{N+1} \boldsymbol{\nu}_1 & \cdots & \cdots & h_{N+1,N} \boldsymbol{d}_{N+1} \boldsymbol{\nu}_N \\ 0 & h_{N+2,2} \boldsymbol{d}_{N+2} \boldsymbol{\nu}_2 & \cdots & h_{N+2,N} \boldsymbol{d}_{N+2} \boldsymbol{\nu}_N \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{2N,N} \boldsymbol{d}_{2N} \boldsymbol{\nu}_N \end{pmatrix}.$$

$\boldsymbol{E}_0$ describes the start of component after a repair, $\boldsymbol{E}_1$ the immediate failure from one of the operational phases and $\boldsymbol{E}_2$ contains all transitions that transfer the component from a maintenance phase to an operational phase. Then

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{D}_0 & \boldsymbol{E}_0 & \boldsymbol{0} \\ \boldsymbol{E}_1 & \boldsymbol{F} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{E}_2 & \boldsymbol{D}_{N+1:2N} \end{pmatrix} \tag{4}$$

contains all transitions between decision points. Matrix $\boldsymbol{Q}$ is block structured into $(2N+1) \times (2N+1)$ blocks of size $m_i \times m_j$ for sub-matrix $\boldsymbol{Q}_{i,j}$.

Rewards are collected in a column vector $\boldsymbol{r}$ of length $\sum_{i=0}^{2N} m_i$. All states belonging to the phase i with $i = 0 \vee N < i \leq 2N$ have reward r_i. The state belonging to phase i $(1 \leq i \leq N)$ and state j $(1 \leq j \leq m_i)$, which corresponds to entry number $\sum_{k=0}^{i-1} m_k + j - 1$ in vector $\boldsymbol{r}$, equals $r_i + p_i \boldsymbol{d}_i(j) r_0$. For most algorithms to compute optimal policies for MDPs or BMDPs it is advantageous to have only non-negative rewards. In finite state processes with bounded rewards

this can be achieved by adding an appropriate constant which is later removed from the solution [20]. Thus define $r_{pos} = \left|\min\left(0, \min_j(\boldsymbol{r}(j))\right)\right|$ and substitute $\boldsymbol{r}$ by $\boldsymbol{r} + r_{pos}\mathbb{1} \geq \mathbf{0}$.

Now consider an inspection time and assume that the component is in phase $i \in \{0, \ldots, 2N\}$ at time $l \cdot \Delta$. We want to compute the probability that the component is in phase $j \in \{0, \ldots, 2N\}$ at time $(l+1) \cdot \Delta$. If the internal state of phase i is known, this probability can be computed exactly using matrix $\boldsymbol{Q}$. However, we only know that for the distribution of states in phase i the relation $\boldsymbol{\psi} = \boldsymbol{\nu}_i e^{t^* \boldsymbol{D}_i} / \left(\boldsymbol{\nu}_i e^{t^* \boldsymbol{D}_i} \mathbb{1}\right)$ for some $t^* \geq 0$ holds. To compute bounds for the probabilities we define the following matrices.

$$\hat{\boldsymbol{B}}_i^- = \begin{pmatrix} -\lambda_i^+ & \lambda_i^- \boldsymbol{b}_i \\ \mathbf{0} & \boldsymbol{Q} \end{pmatrix}, \quad \hat{\boldsymbol{B}}_i^+ = \begin{pmatrix} -\lambda_i^- & \lambda_i^+ \boldsymbol{b}_i \\ \mathbf{0} & \boldsymbol{Q} \end{pmatrix}, \quad \hat{\boldsymbol{B}}_i^{av} = \begin{pmatrix} -\lambda_i^{av} & \lambda_i^{av} \boldsymbol{b}_i \\ \mathbf{0} & \boldsymbol{Q} \end{pmatrix} \tag{5}$$

where $i \in \{0, \ldots, 2N\}$ and

$$\boldsymbol{b}_i = \begin{cases} \left(\mathbf{0}_{m_{1:N-1}}, \boldsymbol{\nu}_N, \mathbf{0}_{m_{N+1:2N}}\right) & \text{if } i = 0, \\ \left(\boldsymbol{\nu}_\mathbf{0}, \mathbf{0}_{m_{1:2N}}\right) & \text{if } i = 1, \\ \left(p_i \boldsymbol{\nu}_\mathbf{0}, \mathbf{0}_{m_{1:i-2}}, \overline{p}_i \boldsymbol{\nu}_{i-1}, \mathbf{0}_{m_{i:2N}}\right) & \text{if } 1 < i \leq N \\ \left(\mathbf{0}_{m_{0:i-N-1}}, h_{-N,i-N} \boldsymbol{\nu}_{i-N}, \ldots, h_{i-N,N} \boldsymbol{\nu}_N, \mathbf{0}_{m_{N+1:2N}}\right) & \text{if } N < i \leq 2N. \end{cases}$$

Matrix $\hat{\boldsymbol{B}}_i^-$ describes the component when it leaves phase i as soon as possible and the flow into the other phases is minimal. $\hat{\boldsymbol{B}}_i^+$ describes the component when it leaves phase i as late as possible and the flow into the other phases is maximal. $\hat{\boldsymbol{B}}_i^{av}$ describes the average behavior. Observe that only $\hat{\boldsymbol{B}}_i^{av}$ is a generator matrix. In the sequel we use $\pm$ for one of the elements from $\{-, +, av\}$.

Let $\boldsymbol{\phi} = \left(1, \mathbf{0}_{m_{0:2N}}\right)$ and compute $\boldsymbol{\phi}_i^\pm = \boldsymbol{\phi} e^{\Delta \hat{\boldsymbol{B}}_i^\pm}$. All vectors can be decomposed into $\boldsymbol{\phi}_i^\pm = \left(\phi_{i,-1}^\pm, \boldsymbol{\phi}_{i,0}^\pm, \ldots, \boldsymbol{\phi}_{i,2N}^\pm\right)$, $\phi_{i,-1}^\pm$ is a scalar which denotes the probability that state i has not been left during the whole interval and $\boldsymbol{\phi}_{i,j}^\pm$ is a vector of length m_j which includes (bounds for) the probabilities that the component is in one of the states of phase j after starting in i. Let

$$\hat{\boldsymbol{\phi}}_i^\pm = \left(\boldsymbol{\phi}_{i,0}^\pm \mathbb{1}, \ldots, \boldsymbol{\phi}_{i,i-1}^\pm \mathbb{1}, \boldsymbol{\phi}_{i,i}^\pm \mathbb{1} + \boldsymbol{\phi}_{i,-1}^\pm, \boldsymbol{\phi}_{i,i+1}^\pm \mathbb{1}, \ldots, \boldsymbol{\phi}_{i,2N}^\pm \mathbb{1}\right)^T. \tag{6}$$

Theorem 1. *Vector $\hat{\boldsymbol{\phi}}_i^-$ is a lower bound for the conditional probability distribution over the phases of the component at time $(l+1) \cdot \Delta$ under the condition that the process starts in phase i at time $l \cdot \Delta$.*

Vector $\hat{\boldsymbol{\phi}}_i^+$ is an upper bound for the conditional probability distribution over the phases of the component at time $(l+1) \cdot \Delta$ under the condition that the process starts in phase i at time $l \cdot \Delta$.

The proof for the theorem can be found online [1]. Vector $\hat{\boldsymbol{\phi}}_i^-$ contains only non-negative elements, whereas $\hat{\boldsymbol{\phi}}_i^+$ might include elements larger than 1 since the

matrix $\hat{\boldsymbol{B}}_i^+$ is superstochastic. Then vector elements can be substituted by the trivial upper bound 1 for probabilities. Now define $(2N+1)\times(2N+1)$ matrices

$$\hat{\boldsymbol{S}}^{\pm} = \left(\hat{\boldsymbol{\phi}}_0^{\pm} \ldots \hat{\boldsymbol{\phi}}_{2N}^{\pm}\right)^{\top} \tag{7}$$

$\hat{\boldsymbol{S}}^-$ is substochastic, $\hat{\boldsymbol{S}}^{av}$ is stochastic and $\hat{\boldsymbol{S}}^+$ is superstochastic[1] and $\hat{\boldsymbol{S}}^- \leq \hat{\boldsymbol{S}}^{av} \leq \hat{\boldsymbol{S}}^+$. Matrices $\hat{\boldsymbol{S}}^-$ and $\hat{\boldsymbol{S}}^+$ bound the transition probabilities of the component between two decision periods.

A maintenance policy can now be described by a vector $\hat{\boldsymbol{a}} \in \{0,1\}^N$. $\hat{\boldsymbol{a}}(i) = 1$ implies that maintenance is initiated if the component is in phase $i \in \{1,\ldots,N\}$. Obviously maintenance can only be initiated if the component is in an operational phase. Matrix $\hat{\boldsymbol{T}}_{\hat{\boldsymbol{a}}}$ describes the transitions induced by policy $\hat{\boldsymbol{a}}$ and is defined as

$$\hat{\boldsymbol{T}}_{\hat{\boldsymbol{a}}}(i,j) = \begin{cases} 1 \text{ if } (1 \leq i \leq N \wedge j = N+i \wedge \hat{\boldsymbol{a}}(i) = 1) \vee ((i \notin \{1,\ldots,N\} \vee \hat{\boldsymbol{a}}(i) = 0) \wedge i = j) \\ 0 \text{ otherwise} \end{cases} \tag{8}$$

Then $\hat{\boldsymbol{P}}_{\hat{\boldsymbol{a}}}^- = \hat{\boldsymbol{T}}_{\hat{\boldsymbol{a}}}\hat{\boldsymbol{S}}^-$ and $\hat{\boldsymbol{P}}_{\hat{\boldsymbol{a}}}^+ = \hat{\boldsymbol{T}}_{\hat{\boldsymbol{a}}}\hat{\boldsymbol{S}}^+$ are matrices containing transition probability bounds of a BMDP. Only for the first state, the reward has to be computed, for the remaining states reward vector $\boldsymbol{r}$ can be used. Define $r_i^{\pm} = r_i + \lambda_i^{\pm} r_0 + r_{pos}$ if $i \in \{1,\ldots,N\}$ and $r_i^{\pm} = r_i + r_{pos}$ otherwise, where r_{pos} has been defined above. Furthermore $u^+ = \max_i r_i^+$ and $u^- = \min_i r_i^-$. Reward vector $\hat{\boldsymbol{s}}^{\pm}$ is defined element-wise as

$$\hat{\boldsymbol{s}}^{\pm}(i) = \boldsymbol{e}_{0,m_{0:2N}+2} \int_0^{\Delta} e^{\tau \hat{\boldsymbol{C}}_i} \begin{pmatrix} \hat{r}_i^{\pm} \\ \boldsymbol{r} \\ u^{\pm} \end{pmatrix} \mathrm{d}\tau \text{ where } \hat{\boldsymbol{C}}_i = \begin{pmatrix} -\lambda_i^+ & \lambda_i^- \boldsymbol{b}_i & \lambda_i^+ - \lambda_i^- \\ \boldsymbol{0} & \boldsymbol{Q} & \boldsymbol{0} \\ 0 & \boldsymbol{0} & 0 \end{pmatrix}. \tag{9}$$

It is easy to show that $\hat{\boldsymbol{s}}^- \leq \hat{\boldsymbol{s}}^{av} \leq \hat{\boldsymbol{s}}^+$ holds.

Theorem 2. *$\hat{\boldsymbol{s}}^-(i)$ is a lower bound for the reward that is gained in an interval $[l\cdot\Delta(l+1)\cdot\Delta]$ under the condition that the process starts at time $l\cdot\Delta$ in phase i.*

$\hat{\boldsymbol{s}}^+(i)$ is an upper bound for the reward that is gained in an interval $[l\cdot\Delta(l+1)\cdot\Delta]$ under the condition that the process starts at time $l\cdot\Delta$ in phase i.

Again, the proof can be found online [1]. $\left(\hat{\boldsymbol{P}}_{\hat{\boldsymbol{a}}}^-, \hat{\boldsymbol{P}}_{\hat{\boldsymbol{a}}}^+\right)_{\hat{\boldsymbol{a}}\in\{0,1\}^N}$ and $\left(\hat{\boldsymbol{s}}^-, \hat{\boldsymbol{s}}^+\right)$ define a BMDP to bound the reward of the policies. Furthermore, $\hat{\boldsymbol{P}}_{\hat{\boldsymbol{a}}}^{av} = \hat{\boldsymbol{T}}_{\hat{\boldsymbol{a}}}\hat{\boldsymbol{S}}^{av}$ is the stochastic transition matrix and $\left(\hat{\boldsymbol{P}}_{\hat{\boldsymbol{a}}}^{av}\right)_{\hat{\boldsymbol{a}}\in\{0,1\}^N}$ and $\hat{\boldsymbol{s}}^{av}$ define the discrete time MDP which models the average behavior of the component.

After the optimal policy has been computed from the BMDP, the policy can be analyzed by building the CTMC described by the policy and then analyzing the policy on this process. Time complexity analysis can be found in [1].

[1] We denote a non-negative matrix $\boldsymbol{A}$ as substochastic, iff a stochastic matrix $\boldsymbol{P}$ with $\boldsymbol{A} \leq \boldsymbol{P}$ and $\boldsymbol{A} \neq \boldsymbol{P}$ exists. Similarly we denote $\boldsymbol{A}$ as superstochastic, iff a stochastic matrix $\boldsymbol{P}$ with $\boldsymbol{A} \geq \boldsymbol{P}$ and $\boldsymbol{A} \neq \boldsymbol{P}$ exists.

4.2 Decision Models for Composed Components

Modeling of composed components is more difficult than the analysis of a single component. We consider here the case of two components but the approach can be easily extended to more than two components.

In a two-component system, phases are described by tuples $\boldsymbol{i} = (i_0, i_1)$ where $i_k \in \{0, \ldots, 2N^{(k)}\}$ and $k \in \{0, 1\}$ is the index of the component. We use $.^{(k)}$ to denote quantities belonging to component k. Matrices and vectors belonging to the composed system will be shown without postfix. A detailed state of the composed system is given by $(\boldsymbol{i}, \boldsymbol{j}) = (i_0, i_1, j_0, j_1)$, $i_k \in \{0, \ldots, 2N^{(k)}\}$ and $j_k \in \{1, \ldots, m_{i_k}^{(k)}\}$.

At a decision point, the detailed state of both components is unknown. Consequently, we have to analyze all possible starting phases (i_0, i_1) using matrices $\bar{\boldsymbol{B}}^{\pm}_{i_0,i_1}$ which are an extension of the matrices $\hat{\boldsymbol{B}}^{\pm}_{i}$ defined in (5). However, in contrast to the case with one component we now have to distinguish four cases; both components remain in their phase for the whole interval of length Δ, the first component changes its phase and the second remains in its phase, the first component remains in its phases and the second changes its phases, and, finally, both components change their phases. This results in the definition of the matrices $\bar{\boldsymbol{B}}^{\pm}_{i_0,i_1}$ by

$$\bar{\boldsymbol{B}}^{-}_{i_0,i_1} = \begin{pmatrix} -\lambda^{+(0)}_{i_0} - \lambda^{+(1)}_{i_1} & \lambda^{-(1)}_{i_1}\boldsymbol{b}^{(1)}_{i_1} & \lambda^{-(0)}_{i_0}\boldsymbol{b}^{(0)}_{i_0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}^{(0)} \oplus -\lambda^{+(1)}_{i_1} & \boldsymbol{0} & \boldsymbol{I} \otimes \lambda^{-(1)}_{i_1}\boldsymbol{b}^{(1)}_{i_1} \\ \boldsymbol{0} & \boldsymbol{0} & -\lambda^{+(0)}_{i_0} \oplus \boldsymbol{Q}^{(1)} & \lambda^{-(0)}_{i_0}\boldsymbol{b}^{(0)}_{i_0} \otimes \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{Q}^{(0)} \oplus \boldsymbol{Q}^{(1)} \end{pmatrix}.$$

The matrices $\bar{\boldsymbol{B}}^{+}_{i_0,i_1}$ and $\bar{\boldsymbol{B}}^{av}_{i_0,i_1}$ are defined analogously by swapping $+$ and $-$ resp. using av in the superscripts.

Then define $\bar{\boldsymbol{\phi}} = \left(1, \boldsymbol{0}_{(m^{(0)}_{0:2N^{(0)}}+1)(m^{(1)}_{0:2N^{(1)}}+1)}\right)$ and compute

$$\bar{\boldsymbol{\phi}}^{\pm}_{(i_0,i_1)} = \bar{\boldsymbol{\phi}} e^{\Delta \bar{\boldsymbol{B}}^{\pm}_{i_0,i_1}}. \tag{10}$$

Vector $\bar{\boldsymbol{\phi}}^{\pm}_{(i_0,i_1)}$ can be decomposed into sub-vectors. The first element equals the scalar $\bar{\phi}^{\pm}_{(i_0,i_1)(-1,-1)}$ including a bound or the average value for the probability that the initial phases have not been left. Then follow $2N^{(1)}+1$ vectors $\bar{\boldsymbol{\phi}}^{\pm}_{(i_0,i_1)(-1,j_1)}$ of length $m^{(1)}_{j_1}$, followed by $2N^{(0)}+1$ vectors $\bar{\boldsymbol{\phi}}^{\pm}_{(i_0,i_1)(j_0,-1)}$ of length $m^{(0)}_{j_0}$. Finally $N^{(0)}N^{(1)}$ vectors $\bar{\boldsymbol{\phi}}^{\pm}_{(i_0,i_1)(j_0,j_1)}$ of length $m^{(0)}_{j_0}m^{(1)}_{j_1}$ follow. From these sub-vectors the $(2N^{(0)}+1)(2N^{(1)}+1) \times (2N^{(0)}+1)(2N^{(1)}+1)$ matrix $\bar{\boldsymbol{S}}$ can be computed as shown in the following equation.

$$\bar{\boldsymbol{S}}((i_0,i_1)(j_0,j_1))^{\pm} = \begin{cases} \bar{\boldsymbol{\phi}}^{\pm}_{(i_0,i_1)(j_0,j_1)}\mathbb{I} & \text{if } i_0 \neq j_0 \wedge i_1 \neq j_1, \\ \bar{\boldsymbol{\phi}}^{\pm}_{(i_0,i_1)(j_0,j_1)}\mathbb{I} + \bar{\boldsymbol{\phi}}^{\pm}_{(i_0,i_1)(-1,j_1)}\mathbb{I} & \text{if } i_0 = j_0 \wedge i_1 \neq j_1, \\ \bar{\boldsymbol{\phi}}^{\pm}_{(i_0,i_1)(j_0,j_1)}\mathbb{I} + \bar{\boldsymbol{\phi}}^{\pm}_{(i_0,i_1)(j_0,-1)}\mathbb{I} & \text{if } i_0 \neq j_0 \wedge i_1 = j_1, \\ \bar{\boldsymbol{\phi}}^{\pm}_{(i_0,i_1)(j_0,j_1)}\mathbb{I} + \bar{\boldsymbol{\phi}}^{\pm}_{(i_0,i_1)(-1,j_1)}\mathbb{I} & \\ +\bar{\boldsymbol{\phi}}^{\pm}_{(i_0,i_1)(j_0,-1)}\mathbb{I} + \bar{\phi}^{\pm}_{(i_0,i_1)(-1,-1)} & \text{if } i_0 = j_0 \wedge i_1 = j_1. \end{cases} \tag{11}$$

Again the relation $\bar{\boldsymbol{S}}^{-} \leq \bar{\boldsymbol{S}}^{av} \leq \bar{\boldsymbol{S}}^{+}$ holds and $\bar{\boldsymbol{S}}^{av}$ is a stochastic matrix. In $\bar{\boldsymbol{S}}^{+}$ elements larger than 1 can be substituted by the trivial bound 1.

Maintenance decisions now can be made whenever the component is in an operational phase at a decision point. We define two vectors $\bar{\boldsymbol{a}}^{(k)}$ $(k = 0, 1)$ of length $(2N^{(0)} + 1)(2N^{(1)} + 1)$. $\bar{\boldsymbol{a}}^{(k)}(i_k, i_{\bar{k}}) = 0$ if $i_k \notin \{1, \ldots, N^{(k)}\}$ and for $i_k \in \{1, \ldots, N^{(k)}\}$ the element can be 1, a maintenance operation for component k starts, or 0, no maintenance starts. In a state, up to 4 decisions can be made by combining the two possibilities for each component. Maintenance decisions are only possible in $N^{(k)}$ operational phases but may depend on the phase of the other component. Again, if the maintenance capacity is restricted, maintenance operations can only be started if capacity is available, i.e. the other component is not in a maintenance phase. For vectors $\bar{\boldsymbol{a}}^{(0)}, \bar{\boldsymbol{a}}^{(1)}$ the binary matrix $\bar{\boldsymbol{T}}_{\bar{\boldsymbol{a}}^{(0)},\bar{\boldsymbol{a}}^{(1)}}$ is defined by

$$\begin{aligned} \bar{\boldsymbol{T}}_{\bar{\boldsymbol{a}}^{(0)},\bar{\boldsymbol{a}}^{(1)}}(i_0,i_1)(j_0,j_1) = 1 \Leftrightarrow (i_0 = j_0 \wedge i_1 = j_1 \wedge \bar{\boldsymbol{a}}^{(0)}(i_0,i_1) = \bar{\boldsymbol{a}}^{(1)}(i_0,i_1) = 0) \\ \vee (j_0 = N^{(0)} + i_0 \wedge i_1 = j_1 \wedge \bar{\boldsymbol{a}}^{(0)}(i_0,i_1) = 1 \wedge \bar{\boldsymbol{a}}^{(1)}(i_0,i_1) = 0) \\ \vee (i_0 = j_0 \wedge j_1 = N^{(1)} + i_1 \wedge \bar{\boldsymbol{a}}^{(0)}(i_0,i_1) = 0 \wedge \bar{\boldsymbol{a}}^{(1)}(i_0,i_1) = 1) \\ \vee (j_0 = N^{(0)} + i_0 \wedge j_1 = N^{(1)} + i_1 \wedge \bar{\boldsymbol{a}}^{(0)}(i_0,i_1) = \bar{\boldsymbol{a}}^{(1)}(i_0,i_1) = 1) \end{aligned} \tag{12}$$

otherwise $\bar{\boldsymbol{T}}_{\bar{\boldsymbol{a}}^{(0)},\bar{\boldsymbol{a}}^{(1)}}(i_0,i_1)(j_0,j_1) = 0$. Then $\bar{\boldsymbol{P}}^{-}_{\bar{\boldsymbol{a}}^{(0)},\bar{\boldsymbol{a}}^{(1)}} = \bar{\boldsymbol{T}}_{\bar{\boldsymbol{a}}^{(0)},\bar{\boldsymbol{a}}^{(1)}}\bar{\boldsymbol{S}}^{-}$ and $\bar{\boldsymbol{P}}^{+}_{\bar{\boldsymbol{a}}^{(0)},\bar{\boldsymbol{a}}^{(1)}} = \bar{\boldsymbol{T}}_{\bar{\boldsymbol{a}}^{(0)},\bar{\boldsymbol{a}}^{(1)}}\bar{\boldsymbol{S}}^{+}$ define transitions probability bounds for a BMDP. $\bar{\boldsymbol{P}}^{av}_{\bar{\boldsymbol{a}}^{(0)},\bar{\boldsymbol{a}}^{(1)}} = \bar{\boldsymbol{T}}_{\bar{\boldsymbol{a}}^{(0)},\bar{\boldsymbol{a}}^{(1)}}\bar{\boldsymbol{S}}^{av}$ is a stochastic matrix for an MDP describing the average behavior.

It remains to compute the reward vectors for the BMDP and the MDP. Reward bounds are computed similarly to the case with one component in (9).

$$\bar{\boldsymbol{s}}^{\pm}(i_0,i_1) = \boldsymbol{e}_0 \int_0^{\Delta} e^{\tau \bar{\boldsymbol{C}}_{i_0,i_1}} \begin{pmatrix} r^{\pm}_{i_0,i_1} \\ \bar{\boldsymbol{r}}^{\pm}_{i_0,*} \\ \bar{\boldsymbol{r}}^{\pm}_{*,i_1} \\ \tilde{\boldsymbol{r}} \\ \tilde{u}^{\pm} \end{pmatrix} d\tau, \text{ where}$$

$$\bar{\boldsymbol{C}}_{i_0,i_1} = \begin{pmatrix} -\lambda^{+(0)}_{i_0} - \lambda^{+(1)}_{i_1} & \lambda^{-(1)}_{i_1}\boldsymbol{b}^{(1)}_{i_1} & \lambda^{-(0)}_{i_0}\boldsymbol{b}^{(0)}_{i_0} & \mathbf{0} & \lambda^{+(0)}_{i_0} + \lambda^{+(1)}_{i_1} - \lambda^{-(0)}_{i_0} - \lambda^{-(1)}_{i_1} \\ \mathbf{0} & \boldsymbol{Q}^{(0)} \oplus -\lambda^{+(1)}_{i_1} & \mathbf{0} & \boldsymbol{I} \otimes \lambda^{-(1)}_{i_1}\boldsymbol{b}^{(1)}_{i_1} & (\lambda^{+(1)}_{i_1} - \lambda^{-(1)}_{i_1})\mathbb{I} \\ \mathbf{0} & \mathbf{0} & -\lambda^{+(0)}_{i_0} \oplus \boldsymbol{Q}^{(1)}\lambda^{-(0)}_{i_0}\boldsymbol{b}^{(0)}_{i_0} \otimes \boldsymbol{I} & & (\lambda^{+(1)}_{i_0} - \lambda^{-(1)}_{i_0})\mathbb{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{Q}^{(0)} \oplus \boldsymbol{Q}^{(1)} & \mathbf{0} \\ 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 \end{pmatrix}. \tag{13}$$

$\tilde{\boldsymbol{r}}$ is the reward vector of the system, $\tilde{u}^+ = \max_{i_0,i_1} r^+_{i_0,i_1}$, $\tilde{u}^- = \min_{i_0,i_1} r^-_{i_0,i_1}$ and

$$\begin{aligned} \bar{\boldsymbol{r}}^{\pm}_{i_0,*} &= \left(r^{\pm}_{i_0,0} \mathbb{1}^T_{m_0^{(1)}}, \ldots, r^{\pm}_{i_0,2N^{(1)}+1} \mathbb{1}^T_{m_2^{(1)} N^{(1)}+1} \right)^\top \\ \bar{\boldsymbol{r}}^{\pm}_{*,i_1} &= \left(r^{\pm}_{0,i_1} \mathbb{1}^T_{m_0^{(0)}}, \ldots, r^{\pm}_{2N^{(0)}+1,i_1} \mathbb{1}^T_{m_2^{(0)} N^{(0)}+1} \right)^\top \end{aligned} \tag{14}$$

The values $r^{\pm}_{i_0,i_1}$ have been defined above.

The generation of the processes requires the solution of $(2N^{(0)}+1)(2N^{(1)}+1)$ systems of differential Eqs. (10) and (13) of order $(m^{(0)}_{0:2N^{(0)}} + 1)(m^{(1)}_{0:2N^{(1)}} + 1)$ each. The different computations can be parallelized. The resulting MDPs or BMDPs are usually of a moderate size because the number of phases is usually small.

For computing the optimal policy and average or discounted reward, standard methods for MDP analysis can be applied [19]. For BMDPs optimal policies for the worst case are usually computed. An overview of available numerical methods is given in [4].

4.3 Improved Bounds

The quality of the bounds depends on the difference between λ^+_i and λ^-_i. Up to now we have computed $\lambda^- = \lambda^-_{0,\infty}$ and $\lambda^+ = \lambda^+_{0,\infty}$ which defines bounds without any knowledge how long the component or system is already in the current phase. However, due to regular inspection of a component, some information is available. Thus, if the component has been in a different phase at the last inspection, $\lambda^- = \lambda^-_{0,\Delta}$ and $\lambda^+ = \lambda^+_{0,\Delta}$. More general, if we know that the component has been in the same phase for the last i inspections, then $\lambda^- = \lambda^-_{(i-1)\cdot\Delta,i\cdot\Delta}$ and $\lambda^+ = \lambda^+_{(i-1)\cdot\Delta,i\cdot\Delta}$. The average rate can then be computed $\lambda^{av} = \Delta^{-1} \int_{(i-1)\cdot\Delta}^{i\cdot\Delta} \lambda(t)$.

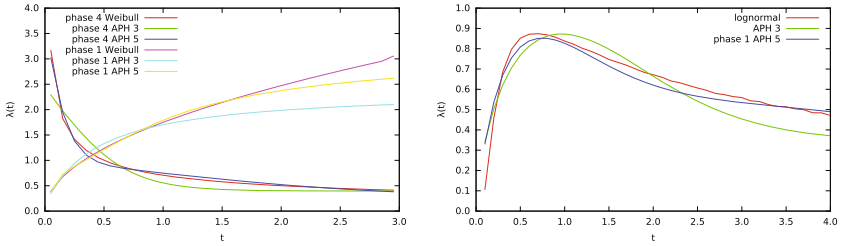
If the number of intervals the system resides in a phase is considered, then MDPs/BMDPs have to be generated and analyzed for each combination of state and residence time which increases the effort. However, for Δ much larger than the expected residence time in a phase not much changes because it is likely that the phase has been left at the next inspection.

5 Examples

The following examples have all been analyzed with Matlab/Octave where the proposed approach has been implemented. Matrices are constructed from the specification of the APHs and the remaining parameters and the resulting MDP or BMDP is analyzed using value iteration. Model generation algorithms and runtime analysis are presented in the online companion [1].

We first consider the computation of maintenance policies for single components and consider afterwards the composition of two components.

Single Components: We start with a single component which has 4 operational phases and a reward of 1 in all operational phases, 0 in maintenance phases, and −10 in the failure phase. The duration of the first and the last operational phase are Weibull distributed. Phase 1 has increasing failure rate and phase 4 decreasing failure rate. The operational phases 2 and 3 are exponentially distributed. This implies that the failure rate over all operational phases is first decreasing, then constant and finally increasing which together gives the typical bathtub like curve. Figure 2a shows the time dependent failure rates of the Weibull distributions and the failure rates of the APHs that have been used to approximate the Weibull distributions. For the computation of the parameters of the APHs an EM algorithm and publicly available software has been applied [21]. It can be seen that 3 states of the underlying Markov chain in the APH model result in a reasonable and 5 states in a good approximation of the Weibull distributions.



(a) Weibull distributions (b) Log-normal distribution

Fig. 2. Time dependent failure rates of various distributions and the respective APH approximations

For the duration of maintenance operations, log-normal distributions have shown to be an adequate model [17]. Usually the variance of the maintenance time is not too large. We generated a log-normal distribution from standard normal distribution and scaled afterwards the distribution to meet the required mean value. APHs of a low order are also suitable to approximate log-normal distributions. Figure 2b show the time dependent rate of the log-normal distribution and APHs with 3 and 5 Markov chain states which have been computed for approximating the log-normal distribution. Again 3 states provide a reasonable and 5 phases a good approximation.

The failure and replacement time is hyper-exponentially distributed with a squared coefficient of variation of 2 to indicate the uncertainty of substituting a component by a new one. We assume that the mean duration of each operational phase is 10, the mean duration of the failure phase 1 and the mean durations of the maintenance phases are 0.4, 0.6, 0.8 and 1, for maintenance starting in the operational phases 4 through 1 (i.e., maintenance becomes more expensive if started later).

The Markov models resulting from a component have 22 or 34 states depending whether we use APHs with 3 or 9 states. The MDP and BMDP have 3 states.

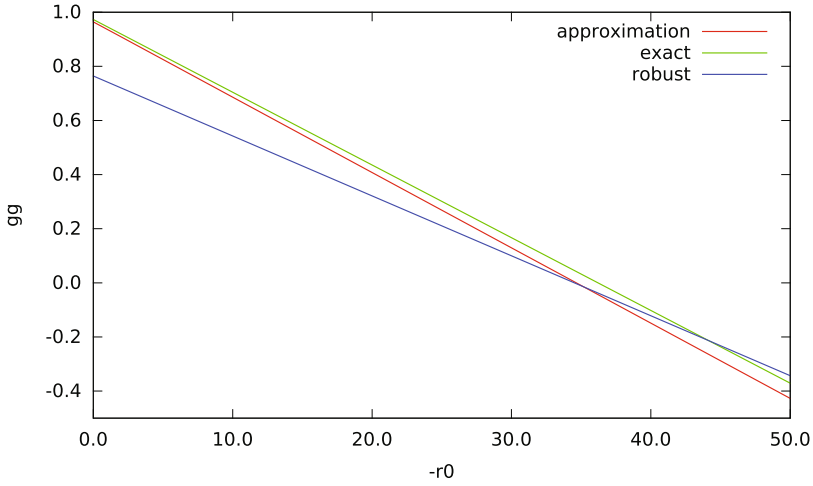


Fig. 3. Optimal gain for different values of the repair costs. (Color figure online)

We consider the configuration with $p_2 = p_3 = p_4 = 0.1$, perfect maintenance (i.e., after each maintenance operation the component behaves like a new component) and an inspection interval of length $\Delta = 1$. If we optimize the average model, no maintenance is performed and the gain is linearly increasing with the costs for a repair. If we analyze the BMDP according to the worst case behavior, a maintenance operation is performed whenever the component is in the last operational phase at an inspection interval. In Fig. 3 we compare the gain under the optimal policy without repair and the robust policy with repair in the last operational phase. The red line shows the gain from the approximation which is used to compute the policy. The green line shows the exact result for the policy. It can be seen that the approximation is for small repair costs good but becomes worse if an repair becomes more expensive. The blue line shows the exact gain of the robust policy which is for small repair cost clearly worse but becomes advantageous if repair cost are high.

Composed Components: For a composed system, we have added a second component that is identical to the first. We analyze the behavior for varying inspection interval $\Delta \in \{0.05, 0.15, \ldots, 1.95, 2\}$ with a penalty of -10. It is assumed that penalties are only paid when both components are down.

Furthermore, we consider a system with fixed $\Delta = 0.5$ and varying penalties in the range from -10 to -50. For varying penalties, we observe in Fig. 4 that the lower bound for optimal policies in the resulting BMDP model decreases with increasing penalty value; the upper bound and the average model are not affected, as the optimal policy in both cases is very close to the upper bound, which is 2 (one reward unit for each component).

For varying time steps, in Fig. 5 one can see that more often inspection contributes to decreasing level of system uncertainty. For a small time interval Δ,

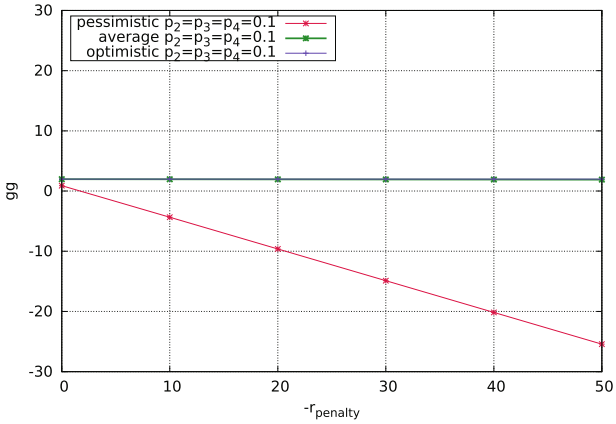
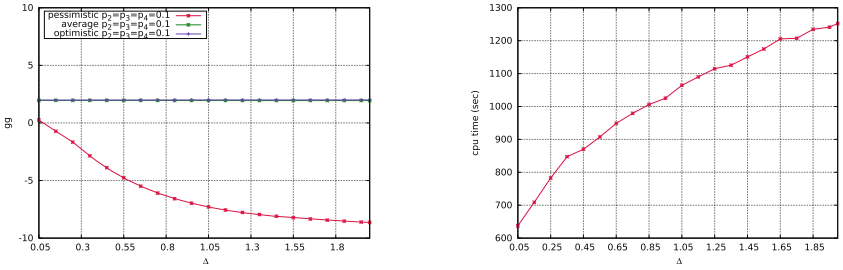


Fig. 4. Gain for the time-based composed system with varying penalties, exclusive maintenance case.



(a) Gain (b) Effort

Fig. 5. Time dependent gains and CPU times for the composed system with two components, exclusive maintenance case. Computational times have been obtained from a mean of 30 independent runs.

we obtain tighter bounds of the underlying BMDP which indicates that for small inspection intervals the captured uncertainty on the internal state of a component is also small and results from uncertainty in the reward of the phase. In Fig. 5a, one can observe that a smaller time step, i. e., a smaller checking interval allows for better control of the system state and thus, for better performance of the composed system. This observation is supported by a slightly better behavior of the average MDP model. However, the price for smaller values of Δ are larger inspection costs which have to be added to the results of the optimization model.

We furthermore observe higher computation times with larger values of Δ. This includes the construction of the BMDP and MDP model as well as their optimization. The main reason for this behavior is the computational complexity of the transient analysis of reward bounds in (13).

6 Conclusion

The paper introduces a new approach to model systems of components with generally distributed failure, maintenance and repair times using Markov chains. By integrating decision points where a maintenance operation may start, a Markov decision process or a bounded parameter Markov decision process, which captures the uncertainty about the internal state of a component, can be generated and optimal maintenance policies can be computed using well established algorithms from Markov decision theory.

The approach has been described here for the composition of two components. It is not hard to extend it to more than two components. For a larger number of components the curse of dimensions resulting in the state space explosion will come up. However, the decision process which have been constructed are fairly small because they only consider the observable phase of a component and not the much larger detailed state space. For the construction of these processes larger systems of linear equations or differential equations have to be solved which can be done nowadays efficiently for relatively large state spaces.

References

1. Markov Maintenance Models Under Uncertainty Online Companion. http://ls4-www.cs.tu-dortmund.de/cms/de/home/dohndorf/publications/
2. Barron, Y., Frostig, E., Levikson, B.: Analysis of R out of N systems with several repairmen, exponential life times and phase type repair times: an algorithmic approach. Eur. J. Oper. Res. **169**(1), 202–225 (2006)
3. Bayle, T.: Preventive maintenance strategy for data centers. White Paper 124, American Power Conversation (2009)
4. Buchholz, P., Dohndorf, I., Scheftelowitsch, D.: Analysis of Markov decision processes under parameter uncertainty. In: Reinecke, P., Di Marco, A. (eds.) EPEW 2017. LNCS, vol. 10497, pp. 3–18. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66583-2_1
5. Buchholz, P., Kriege, J.: Markov modeling of availability and unavailability data. In: 2014 Tenth European Dependable Computing Conference, Newcastle, United Kingdom, 13–16 May 2014, pp. 94–105. IEEE Computer Society (2014)
6. Chan, G.K., Asgarpoor, S.: Optimum maintenance policy with Markov processes. Electr. Power Syst. Res. **76**(67), 452–456 (2006)
7. Chen, D., Cao, Y., Trivedi, K.S., Hong, Y.: Preventive maintenance of multi-state system with phase-type failure time distribution and non-zero inspection time. Int. J. Reliab. Qual. Saf. Eng. **10**(3), 323–344 (2003)
8. Chen, D., Trivedi, K.S.: Optimization for condition-based maintenance with semi-Markov decision process. Rel. Eng. Syst. Saf. **90**(1), 25–29 (2005)
9. Crouzen, P., Pulungan, R.: Acyclic phase-type distributions in fault trees. In: Proceedings of the International Workshop on Performability Modeling of Computer and Communication Systems (2009)
10. Cumani, A.: On the canonical representation of homogeneous Markov processes modeling failure-time distributions. Micorelectronics Reliab. **22**(3), 583–602 (1982)

11. Dekker, R., Nicolai, R.P., Kallenberg, L.: Maintenance and Markov decision models. In: Encyclopedia of Statistics in Quality and Reliability. Wiley, Chichester (2007)
12. Endrenyi, J., et al.: The present status of maintenance strategies and the impact of maintenance on reliability. IEEE Trans. Power Syst. **16**(4), 638–646 (2001)
13. Faddy, M.J.: Phase-type distributions for failure times. Math. Comput. Model. **22**(10–12), 63–70 (1995)
14. Givan, R., Leach, S.M., Dean, T.L.: Bounded-parameter Markov decision processes. Artif. Intell. **122**(1–2), 71–109 (2000)
15. Hatoyama, Y.: Markov maintenance models with repair. Technical Report 175, Stanford University Department of Statistics (1976)
16. Ivy, J.S., Nembhard, H.B.: A modeling approach to maintenance decisions using statistical quality control and optimization. Qual. Reliab. Eng. Int. **21**(4), 355–366 (2005)
17. Kline, M.B.: Suitability of the lognormal distribution for corrective maintenance repair times. Reliab. Eng. **9**(2), 65–80 (1984)
18. Kuhn, K.D., Madanat, S.M.: Robust maintenance policies for Markovian systems under model uncertainty. Comput. Aid. Civil Infrastruct. Eng. **21**(3), 171–178 (2006)
19. Puterman, M.L.: Markov Decision Processes. Wiley, Boca Raton (2005)
20. Sennott, L.I.: Average reward optimization theory for denumerable state spaces. In: Feinberg, E.A., Schwartz, A. (eds.) Handbook of Markov Decision Processes, pp. 153–172. Kluwer, Boston (2002)
21. Thümmler, A., Buchholz, P., Telek, M.: A novel approach for phase-type fitting with the EM algorithm. IEEE Trans. Dependable Sec. Comput. **3**(3), 245–258 (2006)

Markov Automata on Discount!

Yuliya Butkova[1(✉)], Ralf Wimmer[2], and Holger Hermanns[1]

[1] Saarland University, Saarbrücken, Germany
{butkova,hermanns}@cs.uni-saarland.de
[2] Albert-Ludwigs-Universität Freiburg, Freiburg im Breisgau, Germany
wimmer@informatik.uni-freiburg.de

Abstract. Markov automata (MA) are a rich modelling formalism for complex systems combining compositionality with probabilistic choices and continuous stochastic timing. Model checking algorithms for different classes of properties involving probabilities and rewards have been devised for MA, opening up a spectrum of applications in dependability engineering and artificial intelligence, reaching out into economy and finance. In the latter more general contexts, several quantities of considerable importance are based on the idea of discounting reward expectations, so that the near future is more important than the far future. This paper introduces the expected discounted reward value for MA and develops effective iterative algorithms to quantify it, based on value- as well as policy-iteration. To arrive there, we reduce the problem to the computation of expected discounted rewards and expected total rewards in Markov decision processes. This allows us to adapt well-known algorithms to the MA setting. Experimental results clearly show that our algorithms are efficient and scale to MA with hundred thousands of states.

1 Introduction

The design and analysis of complex systems operating in uncertain environments requires a powerful modelling language. It is desirable to support compositionality for constructing large models from individual components; nondeterminism for abstraction and representing unknown behaviour of the environment; continuous stochastic timing and probabilistic choices. *Markov automata* (MA) [9] combine all these aspects in one formalism. They have been extended to express costs and rewards, yielding Markov reward automata [12]. Efficient algorithms for the automatic analysis of Markov (reward) automata are available for a broad range of properties like time- and cost-bounded reachability probabilities [11,14], expected rewards [12], long-run averages [6,11] and properties expressed in the temporal logic CSL [13]. Tool support is available: Both IMCA [10] and Storm [7] support Markov automata model checking. This makes Markov automata well suited not only as a modelling formalism by itself, but also as the semantical

This work was partly supported by the ERC Advanced Grant POWVER (695614) and by the Sino-German project CAP (GZ 1023).

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 19–34, 2018.
https://doi.org/10.1007/978-3-319-74947-1_2

foundation of higher-level formalisms like dynamic fault trees [4] and generalized stochastic Petri nets [8].

All the measures considered so far for Markov reward automata do not make a difference between a unit of reward being accumulated early or late along the evolution of the system. In economics, in artificial intelligence, and in the theory of optimal control [3], it is a well-understood practice to discount the future, that is, to give more weight to the near future than to the far away future. This view is natural in model checking quantitative temporal logic properties of systems [1,2], including continuous-time Markov decision processes (CTMDPs) [17]. It appears equally natural for Markov automata, but as yet, there is neither a theory nor an algorithmic approach for discounting in Markov reward automata.

The present paper investigates discounting on MRA. We first settle the foundational basis of what discounting actually means for Markov automata. Due to the continuous nature of time in MRA we define discounting analogously to the way it is defined for CTMDPs.

Our findings are rooted in the observation that we can view any MRA as a representation encoding a possibly exponentially larger CTMDP, preserving discounted reward values. This enables one to quantify the discounted reward in MRA by computing the respective value on its value-preserving CTMDP, however at the price of possibly exponential time and space requirements.

Overall our approach has similarities in spirit to the one introduced to quantify long-run average rewards on MRA [6], but the constructions needed have to be entirely different due to the dependency of the discounted reward on time. Instead of the naïve approach, we show that the exponential blow-up can be avoided by recognising that the value requiring exponentially many computational steps as the expected total reward in a specific linear-sized discrete-time MDP. Using classic dynamic programming for the latter then turns the exponential naïve approach into an effective polynomial characterisation. In this way we derive the Bellman equation characterising the expected total reward in the presence of discounting in MRA. The Bellman equation in turn is the basis for value- and policy-iteration algorithms quantifying the discounted reward on MRA. The efficiency of the approach is demonstrated with examples of MRAs with hundreds of thousands of states.

2 Foundations

Given a finite set S, a *probability distribution* over S is a function $\mu : S \to [0,1]$ with $\sum_{s \in S} \mu(s) = 1$. We denote the set of all probability distributions over S by $\mathrm{Dist}(S)$. ξ_s is the *Dirac distribution* on s, i. e. $\xi_s(s) = 1$ and $\xi_s(s') = 0$ for $s' \neq s$.

Definition 1. *A* Markov reward automaton (MRA) $\mathcal{M}$ *is a tuple* $\mathcal{M} = (S, s_{\mathrm{init}}, Act, \hookrightarrow, \rightsquigarrow, \mathrm{r}, \rho)$ *s. t. S is a finite set of states; $s_{\mathrm{init}} \in S$ is the initial state; Act is a finite set of actions; $\hookrightarrow \subseteq S \times Act \times \mathrm{Dist}(S)$ is a finite* probabilistic transition relation*; $\rightsquigarrow \subseteq S \times \mathbb{R}_{>0} \times S$ is a finite* Markovian transition relation*; $\mathrm{r} : \hookrightarrow \to \mathbb{R}_{\geqslant 0}$ is a* transition reward function*; and $\rho : S \to \mathbb{R}_{\geqslant 0}$ is a* state reward function.

We abbreviate $(s, \alpha, \mu) \in \hookrightarrow$ by $s \overset{\alpha}{\hookrightarrow} \mu$ and write $s \overset{\lambda}{\rightsquigarrow} s'$ instead of $(s, \lambda, s') \in \rightsquigarrow$. $Act(s) = \{\alpha \in Act \mid \exists \mu \in \mathrm{Dist}(S) : s \overset{\alpha}{\hookrightarrow} \mu\}$ denotes the set of actions that are enabled in state $s \in S$. A state s is *probabilistic (Markovian)*, if it has at least one probabilistic (Markovian) transition $s \overset{\alpha}{\hookrightarrow} \mu$ ($s \overset{\lambda}{\rightsquigarrow} s'$, resp.). States can be both probabilistic and Markovian. We denote the set of probabilistic states by $PS_{\mathcal{M}}$ and the Markovian states by $MS_{\mathcal{M}}$. We assume w. l. o. g. that actions of probabilistic transitions of a state are pairwise different[1]. Therefore we will write $\mathrm{r}(s, \alpha)$ instead of $\mathrm{r}(s, \alpha, \mu)$. The successors of a state $s \in S$ are given by $\mathrm{succ}(s) = \{s' \in S \mid \exists \alpha \in Act\ \exists \mu \in \mathrm{Dist}(S) : s \overset{\alpha}{\hookrightarrow} \mu \wedge \mu(s') > 0 \vee \exists \lambda \in \mathbb{R}_{>0} : s \overset{\lambda}{\rightsquigarrow} s'\}$ and its predecessors by $\mathrm{pred}(s) = \{s' \in S \mid s \in \mathrm{succ}(s')\}$.

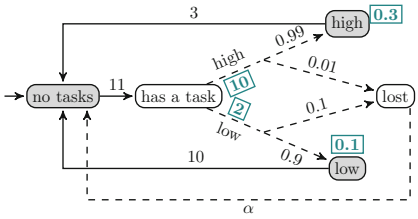


Fig. 1. An example MRA (Color figure online)

For a Markovian state $s \in MS_{\mathcal{M}}$, the value $R(s, s') := \sum_{(s,\lambda,s') \in \rightsquigarrow} \lambda$ is called the *transition rate* from s to s'. The *exit rate* of a Markovian state s is $E(s) := \sum_{s' \in S} R(s, s')$. We require $E(s) < \infty$ for all $s \in MS_{\mathcal{M}}$.

For $s \in PS_{\mathcal{M}}$ with $s \overset{\alpha}{\hookrightarrow} \mu$ for some α, we set $\mathbb{P}[s, \alpha, s'] := \mu(s')$. For $s \in MS_{\mathcal{M}}$ with $E(s) > 0$, the branching probability distribution when leaving the state through a Markovian transition is denoted by $\mathbb{P}[s, \cdot] \in \mathrm{Dist}(S)$ and defined by $\mathbb{P}[s, s'] := R(s, s')/E(s)$.

The evolution of an MRA starts in its initial state. Whenever the system encounters a Markovian state $s \in MS_{\mathcal{M}}$, its sojourn time in s is governed by an exponential distribution, i. e. the probability of leaving s within $t \geqslant 0$ time units is given by $1 - \mathrm{e}^{-E(s) \cdot t}$, after which the next state is chosen according to $\mathbb{P}[s, \cdot]$.

The behaviour of the system in probabilistic states is different. In this paper we consider *closed* MRA, which are not subject to further composition operations that could delay the execution of probabilistic transitions. Therefore we can make the usual *urgency assumption: Probabilistic transitions happen instantaneously.* The residence time in probabilistic states is therefore always 0. Whenever the system is in state s with $Act(s) \neq \emptyset$ and an action $\alpha \in Act(s)$ is chosen, the successor s' is selected according to the distribution $\mathbb{P}[s, \alpha, \cdot]$ and the system moves instantaneously from s to s'. As the execution of a probabilistic transition is instantaneous and because the probability that a Markovian transition is triggered immediately is 0, we can assume that the probabilistic transitions take precedence over Markovian transitions. We therefore assume $PS_{\mathcal{M}} \cap MS_{\mathcal{M}} = \emptyset$. The way Markov automata choose actions will be covered shortly.

During its evolution a Markov reward automaton collects rewards. The transition reward $\mathrm{r}(s, \alpha)$ is granted immediately for taking the (probabilistic)

[1] This can be achieved by renaming the actions and does not affect compositionality properties of MRA due to the fact that only *closed* MRA are considered in this work.

transition $s \overset{\alpha}{\hookrightarrow} \mu$, while the state rewards are accumulated over time, i. e. for staying in a state s for $t > 0$ time units, a reward of $\rho(s) \cdot t$ is granted.

Example 1. Figure 1 shows an example MRA. Grey and white colouring of states indicates the sets $MS_{\mathcal{M}}$ and $PS_{\mathcal{M}}$, resp.; they are disjoint here. Labels *high*, *low*, and α denote actions. Dashed transitions are probabilistic; solid transitions are Markovian. We omitted Dirac distributions from probabilistic transitions. Rewards associated with states and transitions are depicted as numbers in green rectangular frames.

The MRA models a task processing system aiming at maximising the revenue on the long run. Tasks arrive at rate 11; this is modelled by a Markovian transition with rate 11. Whenever there is a task to process, the system decides whether to handle it with high or low reliability. In the former case, the system receives an immediate reward of 10, modelled by $\mathrm{r}(\textit{has a task}, \textit{high}) = 10$. A low reliability task produces a reward of 2 only. The tasks are sent for processing to a remote server over lossy channels. The high reliability channel looses tasks with probability 0.01, while low reliability tasks are lost ten times more often. Whenever a task is lost, no further reward for it is paid. Processing high reliability tasks takes more time, which is modelled with exit rate 3 of state *high*, however, it generates reward proportional to the processing time (modelled by state reward of 0.3). Low reliability tasks are faster to process but produce less reward.

An MRA is *non-Zeno* iff no *maximal end component* (see [11]) of only probabilistic states is reachable with probability > 0. This excludes models in which there is a chance to get trapped in an infinite number of transitions occurring in finite time. In this work, similarly to [11], we restrict ourselves to non-Zeno models; Zenoness is typically considered a modelling error.

For ease of representation, we additionally assume that all states have at least one outgoing transition. This can be easily achieved by adding a Markovian self-loop to the state, with reward 0 and arbitrary non-zero rate.

Paths and Schedulers. A *(timed) path* in $\mathcal{M}$ is a finite or infinite sequence $\pi = s_0 \xrightarrow{\alpha_0,t_0} s_1 \xrightarrow{\alpha_1,t_1} \cdots \xrightarrow{\alpha_k,t_k} s_{k+1} \xrightarrow{\alpha_{k+1},t_{k+1}} \cdots$ with $s_0 = s_{\text{init}}$, and for all $i \geq 0 :\ s_i \in S$, $t_i \in \mathbb{R}_{\geqslant 0}$, and $\alpha_i \in Act \dot{\cup} \{\bot\}$. Here $s_i \xrightarrow{\alpha_i,0} s_{i+1}$ s. t. $\alpha_i \in Act(s_i)$ is a probabilistic transition via action α_i, and $s_i \xrightarrow{\bot,t_i} s_{i+1}$, s. t. $t_i > 0$ and there exists a transition $s_i \overset{\lambda}{\rightsquigarrow} s_{i+1}$, denotes a Markovian transition with sojourn time t_i in state s_i. We define $\pi[i] := s_i$, $\alpha[\pi, i] := \alpha_i$ and for an infinite path π and $k \geq 0$, the elapsed time $\tau[\pi, k]$ until entering $\pi[k]$ is defined by $\tau[\pi, 0] := 0$ and $\tau[\pi, k] := t_0 + \cdots + t_{k-1}$. Whenever it is clear from the context, we omit π and just use $\alpha[i]$ and $\tau[k]$ instead. The set of all finite (infinite) paths of $\mathcal{M}$ is denoted by $Paths^*_{\mathcal{M}}$ ($Paths_{\mathcal{M}}$). The length $|\pi|$ of a finite path π is the number of its transitions; its last state is denoted by $\pi\downarrow$.

In order to resolve the nondeterminism in probabilistic states of an MRA, we need the notion of a scheduler. A *(measurable) scheduler* (or *policy*) σ :

$Paths^*_{\mathcal{M}} \to \mathrm{Dist}(\hookrightarrow)$ is a measurable function, s. t. $\sigma(\pi)$ assigns positive probability only to transitions $(\pi\downarrow, \alpha, \mu) \in \hookrightarrow$, for some α, μ. The set of all measurable schedulers is denoted by $GM_{\mathcal{M}}$. A *(deterministic) stationary scheduler* is a function $\sigma : PS_{\mathcal{M}} \to \hookrightarrow$, s. t. $\sigma(s)$ chooses only from transitions $(s, \alpha, \mu) \in \hookrightarrow$, for some α, μ. For the definition of the probability measure on MRA we refer to [15, Sect. 3.2].

Markov Decision Processes

Definition 2. *A* continuous-time Markov decision process *(CTMDP) is a tuple* $\mathcal{C} = (S, s_{\mathrm{init}}, Act, \mathrm{R})$*, where* S *is a finite set of* states*,* $s_{\mathrm{init}} \in S$ *is an initial state,* Act *is a finite set of* actions*, and* $\mathrm{R} : S \times Act \times S \to \mathbb{R}_{\geqslant 0}$ *is a rate function.*

The set $Act(s) = \{\alpha \in Act \mid \exists s' \in S : \mathrm{R}(s, \alpha, s') > 0\}$ is the set of *enabled actions* in state s. A path in a CTMDP is a finite or infinite sequence $\pi = s_0 \xrightarrow{\alpha_0, t_0} s_1 \xrightarrow{\alpha_1, t_1} \cdots \xrightarrow{\alpha_{k-1}, t_{k-1}} s_k \cdots$, where $s_0 = s_{\mathrm{init}}$, $\alpha_i \in Act(s_i)$ and t_i denotes the residence time of the system in state s_i. $E(s, \alpha) := \sum_{s' \in S} \mathrm{R}(s, \alpha, s')$ and $\mathbb{P}_{\mathcal{C}}[s, \alpha, s'] := \frac{\mathrm{R}(s,\alpha,s')}{E(s,\alpha)}$. The notions of $Paths^*_{\mathcal{C}}$, $Paths_{\mathcal{C}}$, $\pi\downarrow$ and schedulers are defined analogously to corresponding definitions for an MRA. A *reward structure* on a CTMDP $\mathcal{C}$ is a tuple $(\rho_{\mathcal{C}}, \mathrm{r}_{\mathcal{C}})$, where $\rho_{\mathcal{C}} : S \to \mathbb{R}_{\geqslant 0}$ and $\mathrm{r}_{\mathcal{C}} : S \times Act \to \mathbb{R}_{\geqslant 0}$.

The counterpart of CTMDPs and MRA in discrete time are (discrete-time) Markov decision processes:

Definition 3. *A* Markov decision process *(MDP) is a tuple* $\mathcal{D} = (S_{\mathcal{D}}, s_{\mathrm{init}}, Act_{\mathcal{D}}, \mathbb{P}_{\mathcal{D}})$ *where* $S_{\mathcal{D}}$ *is a finite set of* states*,* s_{init} *is the initial state,* $Act_{\mathcal{D}}$ *is a finite set of* actions *and* $\mathbb{P}_{\mathcal{D}} : S_{\mathcal{D}} \times Act_{\mathcal{D}} \to \mathrm{Dist}(S_{\mathcal{D}})$ *is a probabilistic transition function.*

The definitions of paths, schedulers, etc. are discrete analogues of those definitions for CTMDPs. A reward structure on an MDP is a function $\mathrm{r}_{\mathcal{D}} : S_{\mathcal{D}} \times Act_{\mathcal{D}} \to \mathbb{R}_{\geqslant 0}$.

In the following a special subclass of MDPs – acyclic MDPs – will be of particular importance. A state of an MDP is called *terminal* if all its outgoing transitions are self-loops with probability 1 and reward 0. We call an MDP *acyclic* if the self-loops of terminal states are the only loops appearing in the MDP.

3 Discounted Reward for Markov Automata

In this section, we define the discounted reward value for Markov reward automata and consider its relation to discounted rewards on CTMDPs.

3.1 Continuous Discounting

Markov automata are a continuous-time model and we therefore define discounting in a classical way via the continuous exponential decay over time. Yet special care has to be taken when dealing with probabilistic states due to the fact that time in those states does not pass. Essentially the definition is lifted to the MRA setting from CTMDPs [19].

Let $\mathcal{M} = (S, s_{\text{init}}, Act, \hookrightarrow, \rightsquigarrow, \mathrm{r}, \rho)$ be a Markov reward automaton, $\beta > 0$, and $\pi = s_0 \xrightarrow{\alpha_0,t_0} s_1 \xrightarrow{\alpha_1,t_1} \cdots \xrightarrow{\alpha_k,t_k} s_{k+1} \xrightarrow{\alpha_{k+1},t_{k+1}} \cdots$ an infinite path in $\mathcal{M}$. Then $\mathrm{rew}^N_{\mathcal{M},\beta}(\pi)$ is the *discounted reward with rate* β of the path π within $N \in \mathbb{N}$ steps, where $\mathrm{rew}^0_{\mathcal{M},\beta}(\pi) := 0$ and

$$\mathrm{rew}^N_{\mathcal{M},\beta}(\pi) := \sum_{k=0}^{N-1} \left[e^{-\beta\cdot\tau[k]} \cdot \mathrm{r}(s_k, \alpha_k) + \int_{\tau[k]}^{\tau[k]+t_k} e^{-\beta\cdot t} \cdot \rho(s_k)\,\mathrm{d}t \right].$$

Example 2. Consider the MRA from Fig. 1 and its path $\pi = (nt) \xrightarrow{\perp,t_1} (ht) \xrightarrow{high,0} (lost) \xrightarrow{\alpha,0} (nt) \xrightarrow{\perp,t_2} (ht) \longrightarrow \cdots$, where *(nt)* stands for *(no tasks)* and *(ht)* for *(has a task)*. Then the discounted reward collected over this path for $N = 4$ is:

$$\mathrm{rew}^4_{\mathcal{M},\beta}(\pi) = \int_0^{t_1} \rho(nt)\cdot e^{-\beta\cdot\tau}\,\mathrm{d}\tau + e^{-\beta\cdot t_1}\big(\mathrm{r}(ht, high) + \mathrm{r}(lost, \alpha)\big) + \int_{t_1}^{t_1+t_2} \rho(nt)\cdot e^{-\beta\cdot\tau}\,\mathrm{d}\tau\,.$$

The *optimal expected cumulative discounted reward* (or just *discounted reward*) in $\mathcal{M}$ with *discount rate* β is:

$$\mathrm{dR}^{\mathrm{opt}}_{\mathcal{M},\beta} := \operatorname*{opt}_{\sigma\in GM} \left\{ \lim_{N\to\infty} \mathbf{E}_\sigma[\mathrm{rew}^N_{\mathcal{M},\beta}] \right\} = \operatorname*{opt}_{\sigma\in GM} \left\{ \lim_{N\to\infty} \int_{Paths_{\mathcal{M}}} \mathrm{rew}^N_{\mathcal{M},\beta}(\pi)\cdot \Pr_{\mathcal{M},\sigma}[\mathrm{d}\pi] \right\},$$

where $\mathrm{opt} \in \{\sup, \inf\}$. A scheduler σ is called *optimal for* $\mathrm{dR}^{\mathrm{opt}}_{\mathcal{M},\beta}$, if it attains optimum in this equation. In the following, $\mathrm{dR}^{\mathrm{opt}}_{\mathcal{M},\beta}(s)$ denotes the discounted reward collected in $\mathcal{M}$ assuming that the initial state is s. Whenever it is clear from the context, we omit the subscripts and use notation $\mathrm{dR}^{\mathrm{opt}}$.

Lemma 1. *The value* $\mathrm{dR}^{\mathrm{opt}}_{\mathcal{M},\beta}$ *exists.*

3.2 Relation to Discounted Rewards on CTMDP

We now show that for each MRA there exists a (possibly exponentially larger) CTMDP that preserves the discounted reward property. We first need to introduce *uniform* and *normalised* MRA:

Definition 4. *An MRA* $\mathcal{M}$ *is* uniform *if* $\exists \eta \in \mathbb{R}_{>0}$, *s. t.* $\forall s \in MS_{\mathcal{M}}$: $E(s) = \eta$.

Definition 5. *An MRA $\mathcal{M}$ is called* normalised *if*

1. *the initial state of $\mathcal{M}$ is probabilistic;*
2. *every Markovian state s has only probabilistic predecessors:* $\mathrm{pred}(s) \subseteq PS_{\mathcal{M}}$;
3. *probabilistic states of $\mathcal{M}$ have either only probabilistic or only Markovian predecessors:* $\forall s \in PS_{\mathcal{M}} : \mathrm{pred}(s) \subseteq PS_{\mathcal{M}} \vee \mathrm{pred}(s) \subseteq MS_{\mathcal{M}}$.

Lemma 2. *For any MRA $\mathcal{M}, \eta \geqslant \max_{s \in S} E(s)$ there exists a uniform normalised MRA $\overline{\mathcal{M}_\eta}$, s. t.* $\mathrm{dR}^{\mathrm{opt}}_{\overline{\mathcal{M}_\eta},\beta} = \mathrm{dR}^{\mathrm{opt}}_{\mathcal{M},\beta}$ *and its size is linear in the size of $\mathcal{M}$.*

Informally, $\overline{\mathcal{M}_\eta}$ is obtained by first uniformising the Markovian states. This is performed via the well-known approach from [18] by adding self-loop transitions to them. Then this uniform MRA is normalised by introducing probabilistic states of zero reward (i) in between each pair of states that violate Properties 2 or 3 of the definition above, or (ii) as a new initial state, as detailed in [5].

In the following, we assume that the MRA at hand is uniform and normalised and show how to construct a value-preserving CTMDP for it. Before proceeding we need to introduce some notation:

- $\Pi_{\setminus B}(s, s')$ is the set of all *untimed* paths $\pi = s \xrightarrow{\alpha} s_1 \xrightarrow{\alpha_1} \cdots s_k \xrightarrow{\alpha_k} s'$ (paths of $\mathcal{M}$ with abstracted timing information), such that $\forall i = 1..k,\ s_i \notin B$;
- $PS_{\setminus B}(s)$ is the set of states containing s and all states $s' \in PS \setminus B$ that are related to s via the transitive closure of relation $\hookrightarrow$;
- $\mathbb{P}[\pi] := \prod_{i=1}^{|\pi|-1} \mathbb{P}\big[\pi[i], \alpha[i], \pi[i+1]\big]$, $\quad \mathrm{r}(\pi) := \sum_{i=0}^{|\pi|-1} \mathrm{r}\big(\pi[i], \alpha[i]\big)$, $\quad \rho(\pi) := \sum_{i=0}^{|\pi|-1} \rho\big(\pi[i]\big)$.

Value-Preserving CTMDP. Let $\mathcal{M} = (S, s_{\mathrm{init}}, Act, \hookrightarrow, \rightsquigarrow, \mathrm{r}, \rho)$ be a uniform normalised MRA with exit rate η. We define the CTMDP $\mathcal{C}(\mathcal{M}) := (S_{\mathcal{C}}, s_{\mathrm{init}}, Act_{\mathcal{C}}, \mathrm{R}_{\mathcal{C}})$ and reward structure $(\rho_{\mathcal{C}}, \mathrm{r}_{\mathcal{C}})$ as follows:

$\mathbf{S_C}$ The state space of $\mathcal{C}(\mathcal{M})$ is the set $S_{\mathcal{C}} \subseteq PS$ that contains the initial state s_{init} and all probabilistic states of $\mathcal{M}$ that are successors of a Markovian state in $\mathcal{M}$: $S_{\mathcal{C}} = \{s \in PS \mid s = s_{\mathrm{init}} \text{ or } \exists s' \in MS : s' \overset{\lambda}{\rightsquigarrow} s\}$. We define the set of *marked* states as $S_{\mathrm{mrk}} := S_{\mathcal{C}}$.

$\mathbf{Act_C}$ An action of a state s in this CTMDP is a mapping $A : PS_{\setminus S_{\mathrm{mrk}}}(s) \to Act$, such that $A(s') \in Act(s')$. Then the set of all enabled actions $Act_{\mathcal{C}}(s)$ is the set of all possible functions A, and $Act_{\mathcal{C}} = \bigcup_{s \in S_{\mathrm{mrk}}} Act_{\mathcal{C}}(s)$.

$\mathbf{R_C}$ Let $s, s' \in S_{\mathrm{mrk}}$ and $\Pi_{\setminus S_{\mathrm{mrk}}}(s, A, s') \subseteq \Pi_{\setminus S_{\mathrm{mrk}}}(s, s')$ be the set of all paths from $\Pi_{\setminus S_{\mathrm{mrk}}}(s, s')$ that select actions according to A, i. e. for each path π : $\pi[i] \in PS \Rightarrow \alpha[i] = A(\pi[i])$. Then $\mathrm{R}_{\mathcal{C}}(s, A, s') := \eta \cdot \sum_{\pi \in \Pi_{\setminus S_{\mathrm{mrk}}}(s,A,s')} \mathbb{P}[\pi]$.

$\boldsymbol{\rho_C}$ The state reward of a state s in $\mathcal{C}(\mathcal{M})$ is the expected state reward gathered in $\mathcal{M}$ on paths between s and any other $s' \in S_{\mathrm{mrk}}$ that is a successor of s in $\mathcal{C}(\mathcal{M})$: $\rho_{\mathcal{C}}(s) := \sum_{s' \in S_{\mathrm{mrk}}} \sum_{\pi \in \Pi_{\setminus S_{\mathrm{mrk}}}(s,s')} \rho(\pi) \cdot \mathbb{P}[\pi]$.

r$_\mathcal{C}$ The transition reward of a state s and action A in $\mathcal{C}(\mathcal{M})$ is the expected transition reward gathered in $\mathcal{M}$ on paths between s and any other $s' \in S_{\text{mrk}}$ that is a successor of s in $\mathcal{C}(\mathcal{M})$: $\mathrm{r}_\mathcal{C}(s, A) := \sum_{s' \in S_{\text{mrk}}} \sum_{\pi \in \Pi_{\setminus S_{\text{mrk}}}(s,A,s')} \mathrm{r}(\pi) \cdot \mathbb{P}[\pi]$.

The main idea of this construction is to lump together states that are all entered at the same time point. For example, in a sequence of probabilistic states followed by a Markovian state all of the states of the sequence will be entered at the same time due to the fact that probabilistic states are left instantaneously upon entry. The construction is similar in spirit to the construction of a value-preserving CTMDP for the *long-run average reward* problem from [6]. It differs, however, in the treatment of Markovian and probabilistic states due to the fact that timing information effects collected rewards and thus has to be preserved.

An example of this transformation is depicted in Fig. 2. One can easily see that even in small examples the amount of transitions of $\mathcal{C}(\mathcal{M})$ can grow extremely fast. Let $s \in S_{\text{mrk}}$ and $|PS_{\setminus S_{\text{mrk}}}(s)| = n$. If every probabilistic state of $\mathcal{M}$ has two enabled actions, then the set of enabled actions of s in $\mathcal{C}(\mathcal{M})$ is 2^n. This growth is therefore exponential in the worst case.

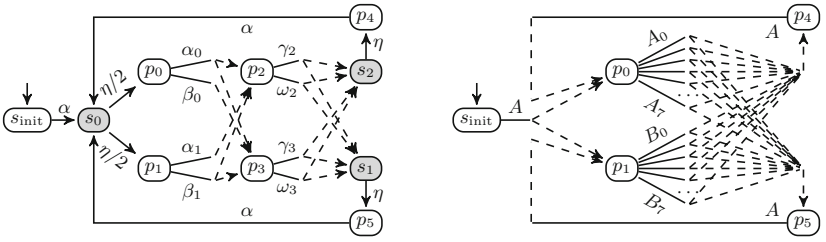


Fig. 2. An example of $\mathcal{C}(\mathcal{M})$ (on the right) for an MRA $\mathcal{M}$ (on the left). We omitted the probabilities of the probabilistic transitions of $\mathcal{M}$. Here $A_0 = [p_0 \to \alpha_0, p_2 \to \gamma_2, p_3 \to \gamma_3]$ and other actions of $\mathcal{C}(\mathcal{M})$ are constructed analogously. If all the probabilistic distributions are uniform, then $\mathrm{R}_\mathcal{C}(p_0, A_0, p_4) = \eta \cdot [0.5 \cdot 0.5 \cdot 1 + 0.5 \cdot 0.5 \cdot 1] = 0.5 \cdot \eta$.

Theorem 1. *For any uniform normalised MRA $\mathcal{M}$ we have* $\mathrm{dR}^{\text{opt}}_{\mathcal{M},\beta} = \mathrm{dR}^{\text{opt}}_{\mathcal{C}(\mathcal{M}),\beta}$[2]*, and there is an optimal scheduler for $\mathcal{M}$ that is stationary.*

4 Bellman Equation

In this section, we introduce the Bellman equation for the discounted reward problem on MRA.

First of all, due to the results obtained in Sect. 3.2, one could obtain the Bellman equation for an MRA by constructing the value-preserving CTMDP

[2] Here $\mathrm{dR}^{\text{opt}}_{\mathcal{C},\beta}$ denotes discounted reward on a CTMDP $\mathcal{C}$ [19].

and using the Bellman equation for this CTMDP[3] [19]. However, since the construction of $\mathcal{C}(\mathcal{M})$ is exponential in the size of $\mathcal{M}$, using the thus obtained Bellman equation for quantifying $\mathrm{dR}^{\mathrm{opt}}$ is not efficient for general MRA.

First we informally discuss the reasons why using this method would be inefficient for $\mathcal{M}$. First of all, in order to use this approach one would need to construct $\mathcal{C}(\mathcal{M})$, which may require exponentially many computations. Additionally, having constructed $\mathcal{C}(\mathcal{M})$, the solution of its Bellman equation requires computing an extremum of an operator $F_{\mathcal{C}(\mathcal{M})}$ over all enabled actions: $V^* := \mathrm{opt}_{A \in Act_{\mathcal{C}}} F_{\mathcal{C}(\mathcal{M})}(A)$. The definition of $F_{\mathcal{C}(\mathcal{M})}$ is irrelevant for the current discussion. Since the number of enabled actions in $\mathcal{C}(\mathcal{M})$ is in the worst case exponential in the size of $\mathcal{M}$, this operation is essentially a brute-force check over exponentially many options. However, we can show that this optimisation problem on $\mathcal{C}(\mathcal{M})$ when mirrored back to $\mathcal{M}$ itself reduces to the computation of the *expected total reward* $\mathrm{tR}^{\mathrm{opt}}_{\mathcal{D}(\mathcal{M})}$ on a discrete-time Markov decision process $\mathcal{D}(\mathcal{M})$, whose size is linear in the size of $\mathcal{M}$. Computing $\mathrm{tR}^{\mathrm{opt}}_{\mathcal{D}(\mathcal{M})}$ is a well-studied problem on MDPs that admits an efficient solution via dynamic programming. Thus instead of naïvely brute-forcing $\sup_{A \in Act_{\mathcal{C}}} F_{\mathcal{C}(\mathcal{M})}(A)$, the value V^* can be efficiently computed by well-known dynamic programming techniques for $\mathrm{tR}^{\mathrm{opt}}_{\mathcal{D}(\mathcal{M})}$ [3,19]. To formalise this result we need to introduce the expected total reward $\mathrm{tR}^{\mathrm{opt}}$ on MDPs, as defined in [19].

Expected Total Reward. Let $\mathcal{D}$ be a (discrete-time) MDP and X_i^s, Y_i^s be random variables denoting the state occupied by $\mathcal{D}$ and the action chosen at step i starting from state s. Then the value

$$\mathrm{tR}^{\mathrm{opt}}_{\mathcal{D},\mathrm{r}_{\mathcal{D}}}(s) := \operatorname*{opt}_{\sigma \in GM_{\mathcal{D}}} \mathbf{E}_{s,\sigma}\left[\lim_{N\to\infty} \sum_{i=0}^{N-1} \mathrm{r}_{\mathcal{D}}(X_i^s, Y_i^s)\right],$$

where $\mathrm{opt} \in \{\sup, \inf\}$, denotes the *optimal expected total reward* on $\mathcal{D}$ with reward structure $\mathrm{r}_{\mathcal{D}}$, starting from state s.

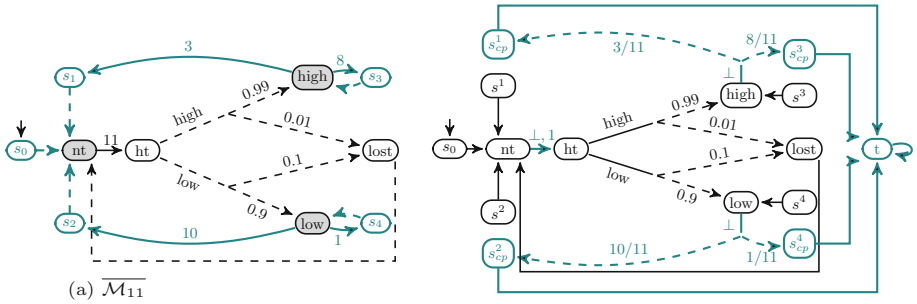
Terminal MDP. We now construct the discrete MDP and the reward structure on it that enables us to substitute the naïve brute-force approach with the efficient computation of the expected total reward.

Let $\mathcal{M}$ be a uniform normalised MRA. Informally, we keep the structure of $\mathcal{M}$, but for each marked state $s \in S_{\mathrm{mrk}}$, we introduce a copy state s_{cp} and redirect all the transitions leading to s to the new copy state s_{cp}. These copy states have only transitions with probability 1 to a new terminal state t. Formally, the *terminal MDP* of $\mathcal{M}$ is $\mathcal{D}(\mathcal{M}) := (S_{\mathcal{D}}, s_{\mathrm{init}}, Act \,\dot\cup\, \{\bot\}, \mathbb{P}_{\mathcal{D}})$, where $S_{cp} = \{s_{cp} \mid s \in S_{\mathrm{mrk}}\}$, $S_{\mathcal{D}} = S \,\dot\cup\, S_{cp} \,\dot\cup\, \{t\}$ and

[3] For details we refer to [5].

$$\mathbb{P}_{\mathcal{D}}[s,\alpha,s'] = \begin{cases} \mathbb{P}[s,\alpha,s'] & \text{for } s \in PS,\ s' \notin S_{cp}, \\ \mathbb{P}[s,p] & \text{for } s \in MS,\ s' = p_{cp} \in S_{cp}, \alpha = \bot, \\ 1 & \text{for } (s \in S_{cp} \text{ or } s = t), s' = t, \alpha = \bot, \\ 0 & \text{otherwise.} \end{cases}$$

Figure 3(b) depicts the terminal MDP for the MRA from Fig. 1.



(a) $\overline{\mathcal{M}_{11}}$ (b) $\mathcal{D}(\overline{\mathcal{M}_{11}})$

Fig. 3. Figure (a) depicts the uniform normalised MRA from Fig. 1 and (b) shows the corresponding terminal MDP. All the added/updated transitions and states are highlighted in green color. The figure omits Dirac distributions and rewards. (Color figure online)

We can now present an efficient characterisation of the discounted reward on MRA. Let $\mathcal{D}(\mathcal{M}) = (S_{\mathcal{D}}, s_{\text{init}}, Act_{\mathcal{D}}, \mathbb{P}_{\mathcal{D}})$ be the terminal MDP of $\mathcal{M}$ and $h : S_{\text{mrk}} \to \mathbb{R}$. We define a reward structure $\text{rew}_{\mathcal{D}(\mathcal{M}),h}$ for $\mathcal{D}(\mathcal{M})$ as follows:

$$\text{rew}_{\mathcal{D}(\mathcal{M}),h}(s,\alpha) := \begin{cases} \text{r}(s,\alpha) & \text{for } s \in PS,\ \alpha \in Act_{\mathcal{D}}(s), \\ \frac{\rho(s)}{\beta+\eta} & \text{for } s \in MS,\ \alpha = \bot, \\ \frac{\eta}{\beta+\eta} h(s) & \text{for } s \in S_{cp},\ \alpha = \bot, \\ 0 & \text{for } s = t,\ \alpha = \bot\,. \end{cases}$$

Theorem 2. *Let $\mathcal{M}$ be a uniform normalised MRA with exit rate η and $\mathcal{D}(\mathcal{M})$ the corresponding terminal MDP. Then the vector $\mathbf{dR}^{\mathbf{opt}}_{\mathcal{M},\boldsymbol{\beta}} := (\text{dR}^{\text{opt}}_{\mathcal{M},\beta}(s)), \forall s \in S_{\text{mrk}}$, is the unique solution to the Bellman equation:*

$$\forall s \in S_{\text{mrk}} : \quad v(s) = \text{tR}^{\text{opt}}_{\mathcal{D}(\mathcal{M}),\text{rew}_{\mathcal{D}(\mathcal{M}),v}}(s)\,. \tag{1}$$

The reason for this characterisation being efficient is the right-hand side of Eq. (1). Quantification of the expected total reward on MDPs is a well-established problem that admits such algorithms as policy-iteration and linear programming [19]. Moreover, for a subclass of models it can be solved in time linear in the size of the MRA. Those are models that have no cycles consisting of only probabilistic states. Such cycles (even non-Zeno ones) almost never happen in real-world applications and are usually considered a modelling mistake. In fact, we are not aware of any practical example where that case occurs.

5 Numerical Solution

Having a Bellman equation at hand, one can normally derive three types of algorithms: based on value-iteration, policy-iteration, and linear programming. Due to scalability issues of the latter we do not consider it and present here only value- and policy-iteration algorithms.

In the following let $sp(v) := \left|\max_{s\in S_{\text{mrk}}} v(s) - \min_{s\in S_{\text{mrk}}} v(s)\right|$. Additionally, we denote with $\texttt{ExpectedTotalReward}(\mathcal{D}, \text{rew}, \sigma)$ the function that computes the expected total reward on an MDP $\mathcal{D}$ for reward structure rew. The last parameter σ can be either a stationary deterministic scheduler or one of $\{\sup, \inf\}$. In the latter case, the optimal expected total reward is computed and in the former the expected total reward for scheduler σ.

Algorithm 1. `ValueIteration`

input : MRA $\mathcal{M} = (S, s_{\text{init}}, Act, \hookrightarrow, \rightsquigarrow, \text{r}, \rho)$, opt $\in \{\sup, \inf\}$, $\beta > 0$, approximation error $\varepsilon > 0$
output : v such that $\|v - \text{dR}^{\text{opt}}_{\mathcal{M},\beta}\| < \varepsilon$, and the ε-optimal scheduler σ

$\overline{\mathcal{M}_\eta} := \text{normalise}\big(\text{uniformise}(\mathcal{M}, \text{rate } \eta := \max_{s\in S} E(s))\big)$;
$\mathcal{D}(\overline{\mathcal{M}_\eta}) :=$ terminal MDP for $\overline{\mathcal{M}_\eta}$;
$v_0 := \overline{0}$; /* *vector of zeros* */
for $(n := 0;\ sp(v_{n+1} - v_n) < \frac{\varepsilon\cdot\beta}{\eta};\ n{+}{+})$ **do**
  $(v_{n+1}, \sigma) := \texttt{ExpectedTotalReward}(\mathcal{D}(\overline{\mathcal{M}_\eta}), \text{rew}_{\mathcal{D}(\overline{\mathcal{M}_\eta}),v_n}, \text{opt})$;
return $v_{n+1}(s_{\text{init}}), \sigma$;

Algorithms 1 and 2 are value- and modified policy-iteration algorithms that compute the value $\text{dR}^{\text{opt}}_{\mathcal{M},\beta}$ for an arbitrary MRA $\mathcal{M}$ and discounting rate $\beta > 0$. The standard policy-iteration algorithm in which the policy evaluation step is performed exactly is also possible in the setting of MRA. However, this requires the exact solution of a linear equation system with one variable per state of $\mathcal{M}$. Since this is an expensive operation for hundreds of thousands of states, we choose the modified policy-iteration instead. The latter bypasses this issue by performing the policy evaluation step numerically. The algorithm depends on a sequence of natural numbers called *order sequence* $(m_n)_{n\in\mathbb{N}_{\geqslant 0}}$; it converges however for an arbitrary sequence.

Theorem 3. *Algorithms 1 and 2 are sound and complete.*

Algorithms 1 and 2 are essentially the respective algorithms on CTMDPs [19], in which the extremum value over enabled actions is searched through the solution of the expected total reward problem $\text{tR}^{\text{opt}}_{\mathcal{D}(\mathcal{M})}$ on the terminal MDP $\mathcal{D}(\mathcal{M})$. Therefore in both algorithms the complexity of an iteration equals the complexity of computing the value $\text{tR}^{\text{opt}}_{\mathcal{D}(\mathcal{M})}$ (which is polynomial), and the convergence rate is the same as the convergence rate of the corresponding CTMDP algorithms.

Algorithm 2. `ModifiedPolicyIteration`

input : MRA $\mathcal{M} = (S, s_{\text{init}}, Act, \hookrightarrow, \rightsquigarrow, \mathrm{r}, \rho)$, opt $\in \{\sup, \inf\}$, $\beta > 0$, $\varepsilon > 0$, order sequence $(m_n)_{n \in \mathbb{N}_{\geqslant 0}}$

output : v such that $\|v - \mathrm{dR}^{\text{opt}}_{\mathcal{M},\beta}\| < \varepsilon$, and the ε-optimal scheduler σ

$\overline{\mathcal{M}_\eta} := \text{normalise}\big(\text{uniformise}(\mathcal{M}, \text{rate } \eta \longleftarrow \max_{s \in S} E(s))\big)$;

$\mathcal{D}(\overline{\mathcal{M}_\eta}) :=$ terminal MDP for $\overline{\mathcal{M}_\eta}$;

$v_0 := \overline{0}$; /* *vector of zeros* */

stop := *false*; $n := 0$;

while (¬stop) **do**

  /* Policy improvement */

  $(u^0_n, \sigma_{n+1}) :=$ `ExpectedTotalReward`$(\mathcal{D}(\overline{\mathcal{M}_\eta}), \text{rew}_{\mathcal{D}(\overline{\mathcal{M}_\eta}),v_n}, \text{opt})$;

  /* Partial policy evaluation */

  if $sp(u^0_n - v_n) < \frac{\varepsilon \cdot \beta}{\eta}$ **then**

    stop := *true*; break;

  for ($k := 0$; $k < m_n$; k++) **do**

    $u^{k+1}_n :=$ `ExpectedTotalReward`$(\mathcal{D}(\overline{\mathcal{M}_\eta}), \text{rew}_{\mathcal{D}(\overline{\mathcal{M}_\eta}),v_n}, \sigma_{n+1})$;

  $v_{n+1} := u^{m_n}_n$;

  $n := n + 1$

return $u^0_n(s_{\text{init}}), \sigma_{n+1}$;

Computation of the Expected Total Reward. Notice that the presented algorithms are guaranteed to converge whenever the expected total reward of the terminal MDP is computed precisely. Exact quantification of this value can be achieved with policy-iteration or linear programming algorithms [19]. Moreover, for models that have no cycles and consist of only probabilistic states, the expected total reward can be solved efficiently in time $O(|\rightsquigarrow| + |\hookrightarrow|)$.

6 Experiments

Here we present the empirical evaluation of the discussed algorithms. Both algorithms were implemented as part of the IMCA/MAMA toolset [10]. All experiments were run on a single core of Intel Core i7-4790 with 8 GB of RAM.

Benchmarks. We have evaluated our approach on a collection of published benchmark models: the *Polling System* [10,21], *Queuing System* [13], and the *Fault Tolerant Workstation Cluster* [16]. Discounting for the selected benchmarks naturally models the decrease of the value of costs over time. In order to address a case study with a specific set of parameters we use the same notation as in [6]. We used the tool SCOOP [20] to generate those models and for this reason the degree of variation of some parameters is restricted by its runtime/space requirements.

Table 1 shows the parameters of some of the used models. We use the symbols $\lambda^{\mathcal{M}}$ to denote the maximal number of enabled actions in probabilistic states of $\mathcal{M}$, and $E(\mathcal{M})$ shows the maximal exit rate of Markovian states of $\mathcal{M}$.

Table 1. Parameters of some of the benchmarks.

	$\lvert S\rvert$	$\lvert PS\rvert$	$\lvert MS\rvert$	$\lvert \hookrightarrow \rvert$	$\lvert \rightsquigarrow \rvert$	$\lambda^{\mathcal{M}}$	$E(\mathcal{M})$
FTWC-resp-50-40	92,819	20,806	72,013	72,007	305,613	5	6.35
PS-256-3-4	131,529	87,605	43,924	189,129	72,965	3	14
QS-256-256	465,177	398,096	67,081	530,966	200,208	2	26

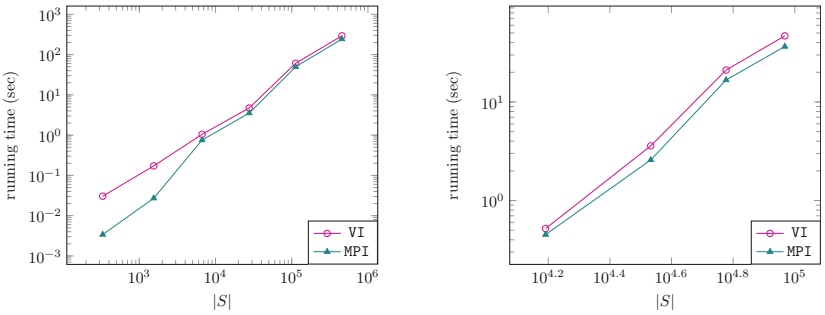
Empirical Evaluation. The space complexity of both algorithms is polynomial. Therefore, we have evaluated the effect of varying model size, precision, and the discounting rate on their runtime only. In plots, whenever the experiment covers several benchmarks, we use the symbol "X" to denote respective part of the model name, e. g. PS-2-X. In this section we will refer to the value-iteration Algorithm 1 as `VI` and to the modified policy-iteration Algorithm 2 as `MPI`.

As we have mentioned in Sect. 5, the modified policy-iteration algorithm depends on a parameter called order sequence. The optimal choice of the order sequence is an open question [19]. In this section, we present the best results we could achieve with different order sequences. Let us notice that if every element of the order sequence is 0 then `MPI` is the same as `VI`. When the values grow infinitely large, the algorithm turns into standard policy-iteration. Almost always in our experiments we could find an order sequence that led to lower running times than those of `VI`. Moreover, relatively small order sequences achieved the best value, e. g. $m_n = 100, \forall n > 0$ was sufficient for most of the models. Selecting a sequence with larger values, however, quite often led to running times significantly worse than those of `VI`.

Model size. Figure 4 shows the dependency of the running time of both algorithms on the size of the state space. Both algorithms exhibit polynomial dependency (linear in log-log scale) on the depicted size range. This agrees with theoretical expectations since the computation of expected total rewards is polynomial in the size of the state space and the convergence rate does not depend on the model size.

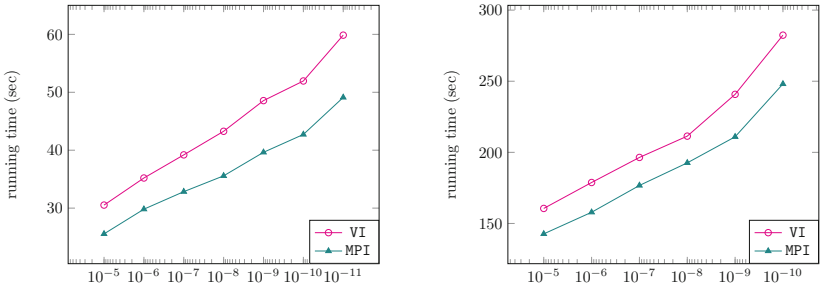
Precision. Figure 5 shows the dependency of the computation time on the precision parameter ε. The theoretical convergence rate of both algorithms resembles that of respective algorithms on CTMDPs. The expected complexity of `VI` is logarithmic in ε, which is supported by the observed results. Regarding `MPI`, we observed that the function of the running time repeats that of `VI`, possibly due to the relatively small values used for the order sequence.

Discounting rate. Figure 6 depicts the dependency of the running time of the algorithms on the discounting rate β. The observed dependency of `VI` follows the theoretical bound of $O(\frac{1}{1-\beta})$. Similarly to the previous case, the function of the running time of policy-iteration repeats that of value iteration.



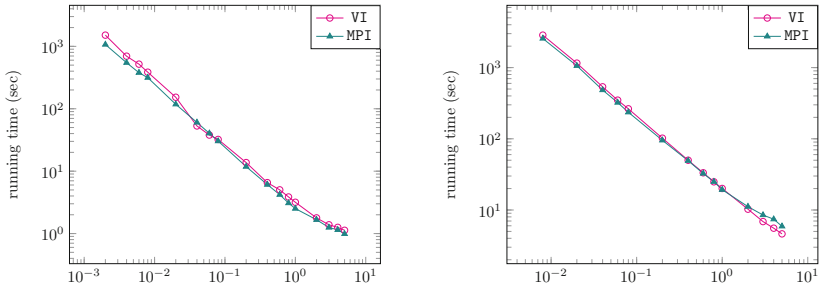
(a) PS-64-2-X (b) FTWC-forg-X-X

Fig. 4. Runtime complexity of `VI` and `MPI` w. r. t. the increase of the model size in log-log scale. For these experiments the discount rate β was set to 0.05 and $\varepsilon = 10^{-8}$.



(a) PS-256-3-4 (b) QS-256-256

Fig. 5. Observed dependency of `VI` and `MPI` on the precision parameter ε in reversed logarithmic x-axis. In these experiments $\beta = 0.1$.



(a) FTWC-50-40-resp (b) QS-256-256

Fig. 6. Observed dependency on discounting rate β in log-log scale. Here $\varepsilon = 10^{-8}$.

7 Conclusion

While discounting is a standard concept on Markov chains and Markov decision processes, this is the first paper to consider discounting for the more general model of Markov reward automata. We have discussed that computing discounted rewards on MRA can be reduced to the same task on a possibly exponentially larger CTMDP. Constructing and optimizing over this large CTMDP can be avoided by recognising the essential computation as determining the expected total reward in a linear-sized discrete-time MDP. This in turn is a well-understood problem enabling an efficient solution. Experiments clearly demonstrate the efficiency of our approach, being able to handle MRAs with hundred thousands of states.

References

1. de Alfaro, L., Faella, M., Henzinger, T.A., Majumdar, R., Stoelinga, M.: Model checking discounted temporal properties. Theor. Comput. Sci. **345**(1), 139–170 (2005)
2. de Alfaro, L., Henzinger, T.A., Majumdar, R.: Discounting the future in systems theory. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 1022–1037. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45061-0_79
3. Bertsekas, D.P.: Dynamic Programming and Optimal Control, 2nd edn. Athena Scientific, Belmont (2000)
4. Boudali, H., Crouzen, P., Stoelinga, M.: A rigorous, compositional, and extensible framework for dynamic fault tree analysis. IEEE Trans. Dependable Secure Comput. **7**(2), 128–143 (2010)
5. Butkova, Y.: Discounted Markov automata. Technical Report 2018–01, ERC Grant POWVER (695614), Universität des Saarlandes, Saarland Informatics Campus, Saarbrücken, Germany (2018). http://www.powver.org/publications/TechRepRep/ERC-POWVER-TechRep-2018-01.pdf
6. Butkova, Y., Wimmer, R., Hermanns, H.: Long-run rewards for Markov automata. In: Legay, A., Margaria, T. (eds.) TACAS 2017. LNCS, vol. 10206, pp. 188–203. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54580-5_11
7. Dehnert, C., Junges, S., Katoen, J.-P., Volk, M.: A storm is coming: a modern probabilistic model checker. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10427, pp. 592–600. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63390-9_31
8. Eisentraut, C., Hermanns, H., Katoen, J.-P., Zhang, L.: A semantics for every GSPN. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 90–109. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38697-8_6
9. Eisentraut, C., Hermanns, H., Zhang, L.: On probabilistic automata in continuous time. In: LICS 2010, pp. 342–351. IEEE CS (2010)
10. Guck, D., Hatefi, H., Hermanns, H., Katoen, J.-P., Timmer, M.: Modelling, reduction and analysis of Markov automata. In: Joshi, K., Siegle, M., Stoelinga, M., D'Argenio, P.R. (eds.) QEST 2013. LNCS, vol. 8054, pp. 55–71. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40196-1_5

11. Guck, D., Hatefi, H., Hermanns, H., Katoen, J., Timmer, M.: Analysis of timed and long-run objectives for Markov automata. Log. Meth. Comput. Sci. **10**(3) (2014)
12. Guck, D., Timmer, M., Hatefi, H., Ruijters, E., Stoelinga, M.: Modelling and analysis of Markov reward automata. In: Cassez, F., Raskin, J.-F. (eds.) ATVA 2014. LNCS, vol. 8837, pp. 168–184. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11936-6_13
13. Hatefi, H., Hermanns, H.: Model checking algorithms for Markov automata. In: Electronic Communication of the EASST, vol. 53 (2012)
14. Hatefi, H., Wimmer, R., Braitling, B., Fioriti, L.M.F., Becker, B., Hermanns, H.: Cost vs. time in stochastic games and Markov automata. Formal Aspects Comput. **29**(4), 629–649 (2017)
15. Hatefi Ardakani, H.: Finite horizon analysis of Markov automata. Ph.D. thesis, Universität des Saarlandes, Saarbrücken, Germany (2017)
16. Haverkort, B.R., Hermanns, H., Katoen, J.: On the use of model checking techniques for dependability evaluation. In: SRDS 2000, pp. 228–237. IEEE CS (2000)
17. Jansen, D.N.: More or less true DCTL for continuous-time MDPs. In: Braberman, V., Fribourg, L. (eds.) FORMATS 2013. LNCS, vol. 8053, pp. 137–151. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40229-6_10
18. Jensen, A.: Markoff chains as an aid in the study of Markoff processes. Scand. Actuarial J. **1953**, 87–91 (1953)
19. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st edn. Wiley, New York (1994)
20. Timmer, M.: SCOOP: a tool for symbolic optimisations of probabilistic processes. In: QEST 2011, pp. 149–150. IEEE CS (2011)
21. Timmer, M., van de Pol, J., Stoelinga, M.I.A.: Confluence reduction for Markov automata. In: Braberman, V., Fribourg, L. (eds.) FORMATS 2013. LNCS, vol. 8053, pp. 243–257. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40229-6_17

Markov Analysis of Optimum Caching as an Equivalent Alternative to Belady's Algorithm Without Look-Ahead

Gerhard Hasslinger(✉)

Deutsche Telekom, H.-Hertz-Str. 3-7, Darmstadt, Germany
gerhard.hasslinger@telekom.de

Abstract. Belady's algorithm defines a strategy for updating cache content in order to achieve the optimum hit rate, based on full information being available about future requests. In this work, we design a Markov state model to analyze and evaluate the hit rate of clairvoyant optimum caching.

We start deriving a new steady state solution for a cache that can store only a single object when requests are independent with arbitrary distribution. Results in literature are restricted to uniform request and we have to transfer them from paging systems to web caches, which refer to slightly different preconditions.

When we follow each specified state transition of the Markov process per request, this leads to an alternative implementation of Belady's algorithm. Therefore the Markov approach can also be applied as an alternative to Belady's algorithm for evaluating optimum caching based on request traces, in simulations and other use cases, e.g. for code optimization. Remarkably, the Markov process decides about optimum caching hits per request based on cache content, based on knowledge only of the past without look-ahead.

Keywords: Web cache strategies · Optimum caching · Belady's algorithm
Hit rate · Markov analysis

1 Introduction: Overview of Web Caching Strategies

Today's broadband Internet infrastructure mainly depends on support by distributed cache servers in the data centers of content delivery networks and clouds [6, 8]. Caches shorten the transport paths and therefore save bandwidth and shorten delays as a major concern of many web services. High availability and high throughput are also primary goals of distributed architectures for content delivery.

Therefore caching strategies exert substantial influence on the performance of Internet services. They have been developed and studied for several decades, but under changing constraints regarding storage, data forwarding technology and transport protocols. Well known caching methods are based on the *least recently used* (LRU) and *least frequently used* (LFU) principle for replacing cache content [4, 11, 16]. LRU is widely used because of its simple implementation with low update effort, whereas LFU includes statistics about past requests to increase the hit rate. When requests are independent (IRM: Independent Request Model), LFU achieves maximum hit rate by collecting the objects with highest popularity in the cache. Many variants of caching

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 35–52, 2018.
https://doi.org/10.1007/978-3-319-74947-1_3

strategies have been studied [14–16], which finally can combine simple LRU processing with flexible score rating for selecting most relevant cache content [11].

Another work stream of caching research considers optimum caching, presuming that the future request sequence is known. Then Belady's algorithm [2] is applicable to maximize the cache hit rate, which again follows a simple cache replacement principle of *farthest next request first*. While Belady's algorithm is also used in related fields like code optimization [7], it provides an upper bound of caching performance, indicating gaps in the hit rate, which may partly be overcome by prediction or when partial information is available about the future. In Sect. 6, we show how a limited look-ahead can be exploited by combining Belady's with other caching algorithms. Further approaches involving optimum caching are addressed in [1, 5, 12].

The main focus of this work is on Markov analysis of the optimum cache hit rate. As to the author's knowledge there have been only two analysis approaches in literature on optimum caching derived by Smith [18] and Knuth [13] already 30 years ago. Both refer to paging systems, assuming that requested data is always put into the cache, whereas web caches can select which data is worth to be cached. It turns out that results of [13, 18] can be straightforwardly transferred to web caches, whereas the analysis of optimum caching is more complex than for LRU or LFU policies.

In Sect. 2, we start deriving a simple but new optimum cache hit rate result for web caches of size $M = 1$ with arbitrary independent requests (IRM), which extends a corresponding result in [13] for uniform requests. Then we specify a general Markov model for arbitrary size N of the object catalogue and cache size M in Sect. 3. Compared to the approach in [18], our Markov model has smaller state space, but a steady state analysis is limited to systems with small N, M. Moreover, a transient evaluation of the Markov process per request turns out to be an equivalent alternative for evaluating Belady's algorithm. In contrast to [2, 18], the Markov approach avoids any look-ahead. Efficient implementations for Belady's algorithm and for the Markov approach are compared in Sect. 4. Section 5 shows evaluation results for the gap in the hit rates between optimum caching, LRU and LFU strategies. Section 6 is discussing how to improve caching efficiency for limited look-ahead options by combining Belady's algorithm with a non-clairvoyant caching strategy, followed by the conclusions.

2 Markov Analysis for a Single Object in the Cache

In the first part, we analyze the hit rate for a cache with a single object ($M = 1$) and for the independent request model (IRM). Then a Markov chain with N states suffices for an object catalogue O of size $|O| = N$, when we inspect the system only at cache hit events. The state k refers to the object $o_k \in O$ in the cache after the hit. We derive the transition probabilities from combinatorial relationships which finally lead to a simple steady state solution of the Markov model. Finally, the solution is compared to results in literature [13, 18], which are restricted to uniform requests.

The intervals between hits and the decisions which object to be put into the cache follow Belady's optimum caching principle [2]. Belady's algorithm has to decide for each cache miss, i.e. when an object has to be retrieved from a server. The object is loaded into the cache, if its next request comes prior to the next request to a cached

object, on account of the object with the longest time until its next request, which is replaced. This also means a state transition in the Markov chain, while the cache remains unchanged in case of a hit. We first determine the transition probabilities q_{kj} that a hit on o_k is followed by a next hit on o_j for optimum caching, from which the steady state probabilities r_j of the Markov chain are derived.

Intervals without hits refer to request sequences for m pairwise different objects excluding the object in the cache, because a request to a cached object or two new requests to the same object always lead to an option for a next cache hit by loading the external object into the cache at its first request for getting a hit at the second.

Let $\pi_m(O)$ denote the probability that a sequence of m requests is addressing m different objects of a set O. We assume independent requests (IRM) with probabilities $p_1, \ldots, p_N$ to the objects $o_1, \ldots, o_N$ reflecting their popularity. Then $\pi_m(O)$ can be computed recursively starting from subsets $O_n = \{o_1, \ldots, o_n\} \subseteq O = O_N$, $n = 1, 2, \ldots, N$ according to Eq. (1) or via the usual combinatorial formula Eq. (2) for addressing m different objects in the next m requests:

$$\pi_m(O_n) = \pi_m(O_{n-1}) + \pi_{m-1}(O_{n-1})mp_n; \tag{1}$$

where $\pi_0(O_n) = 1; \quad \pi_1(O_n) = \sum_{k=1}^{n} p_k; \forall m > n : \pi_m(O_n) = 0;$

$$\pi_m(O_n) = m! \sum_{k_1=,\ldots,k_m \le nj \ne l \Rightarrow k_j \ne k_l} p_{k_1} p_{k_2} \ldots p_{k_m}. \tag{2}$$

The recursive evaluation of $\pi_m(O)$ via Eq. (1) has complexity $O(mn)$ and therefore is preferable to computation of Eq. (2) with exponential complexity $O(n^m)$.

Let q_{kj}^m denote the probability that a hit on object o_k is followed by a next hit on o_j after m intermediate requests without a hit. In case $j = k$, requests without hit are addressing m different objects which also differ from the object o_k in the cache. We first obtain the probabilities $q_{kk}^m (m = 0, \ldots, N - 1)$ and finally the transition probability q_{kk}:

$$q_{kk}^m = p_k \pi_m(O \backslash \{o_k\}); \quad q_{kk} = p_k \sum_{m=0}^{N-1} \pi_m(O \backslash \{o_k\}). \tag{3}$$

When the next hit goes to a different object $o_j \neq o_k$, this object is the first appearing twice in the request sequence after the previous hit on o_k. Then o_j is loaded into the cache at its first request, which can be encountered before, or at any position within the

other m requests to mutually different objects in the set $O \setminus \{o_k, o_j\}$, until the $(m+2)^{\text{th}}$ request is the next hit on o_j. We obtain for $k \neq j$:

$$q_{kj}^m = (m+1)p_j^2\,\pi_m(O\setminus\{o_k,o_j\});\;\; q_{kj} = \sum_{m=0}^{N-2} q_{kj}^m. \tag{4}$$

Based on the transition probabilities of Eqs. (3–4), the steady state probabilities ρ_j of the Markov chain are determined from the equilibrium equations

$$\rho_j = \sum_{k=1}^{N} \rho_k q_{kj} \quad \text{for } j = 1,\ldots,N. \tag{5}$$

The computation of the optimum cache hit rate according to Eqs. (1–5) is subject to a computational complexity of the order $O(N^4)$, mainly for calculating all terms of Eq. (4). However, we can drastically facilitate the analysis by observing the following simple relationship

$$\rho_j = p_j^2 / \sum_{k=1}^{N} p_k^2, \quad \text{for } j = 1,\ldots,N, \tag{6}$$

i.e. the rate ρ_j of hits on object o_j is proportional to the square of its request probability p_j^2. Thus the analytic evaluation can skip Eqs. (3–4) and instead start with Eq. (6).

2.1 Proof of the Steady State Solution $\rho_j \sim p_j^2$ for the Probability of a Hit on Object o_j

We prove Eq. (6) by checking that this approach meets the equilibrium equations (5). We can restrict the verification to the last equilibrium equation for ρ_N, because then the proof is valid for $i = 1, \ldots, N-1$ as well, since the indices of the objects can be permutated, i.e. N may be replaced by i and summations excluding i instead over $1, \ldots, N-1$:

$$\rho_N = \sum_{k=1}^{N} \rho_k q_{kN} \Leftrightarrow (1-q_{NN})\,\rho_N = \sum_{k=1}^{N-1} \rho_k q_{kN} \overset{\text{Eq. (6)}}{\Leftrightarrow} 1 - q_{NN} = \sum_{k=1}^{N-1} \frac{p_k^2}{p_N^2} q_{kN}.$$

The proof proceeds with the latter relationship on the right. Next, we insert the transition probabilities q_{kN} of Eq. (4), yielding the following representation:

$$\begin{aligned} 1 - q_{NN} \overset{(?)}{=} \sum_{k=1}^{N-1} \frac{p_k^2}{p_N^2} q_{kN} &= \sum_{k=1}^{N-1} p_k^2 \sum_{m=0}^{N-2} (m+1)\pi_m(O\setminus\{o_N,o_k\}) \\ &= \sum_{m=0}^{N-2} (m+1) \sum_{k=1}^{N-1} p_k^2 \pi_m(O\setminus\{o_N,o_k\}). \end{aligned} \tag{7}$$

In order to proof Eq. (7), we evaluate each term of the sum on the right separately:

$$\sum_{k=1}^{N-1} p_k^2 \pi_m(O\backslash\{o_N, o_k\}) = \sum_{k=1}^{N-1} p_k \pi_m(O\backslash\{o_N, o_k\})(1 - \sum_{j=1;j\neq k}^{N} p_j)$$

$$= \sum_{k=1}^{N-1} p_k (\sum_{\substack{l_1,\ldots,l_m=1 \\ \forall g\neq h: l_g\neq l_h \\ \forall g\leq m: l_g\neq k}}^{N-1} p_{l_1}\ldots p_{l_m})(1 - p_N - (p_{l_1} + \ldots + p_{l_m}) - \sum_{\substack{j=1;j\neq k \\ \forall g\leq m: j\neq l_g}}^{N-1} p_j)$$

$$= \pi_{m+1}(O\backslash\{o_N\})(1 - p_N) - \sum_{g=1}^{m} \sum_{l_g=1}^{N-1} p_{l_g}^2 \pi_m(O\backslash\{o_N, o_{l_g}\}) - \pi_{m+2}(O\backslash\{o_N\})$$

$$= (1 - p_N)\pi_{m+1}(O\backslash\{o_N\}) - m\sum_{k=1}^{N-1} p_k^2 \pi_m(O\backslash\{o_N, o_k\}) - \pi_{m+2}(O\backslash\{o_N\}) \Leftrightarrow$$

$$(m+1)\sum_{k=1}^{N-1} p_k^2 \pi_m(O\backslash\{o_N, o_k\}) = (1 - p_N)\pi_{m+1}(O\backslash\{o_N\}) - \pi_{m+2}(O\backslash\{o_N\}) \Leftrightarrow$$

$$\sum_{m=0}^{N-2}(m+1)\sum_{k=1}^{N-1} p_k^2 \pi_m(O\backslash\{o_N, o_k\}) = \sum_{m=0}^{N-2}[(1 - p_N)\pi_{m+1}(O\backslash\{o_N\}) - \pi_{m+2}(O\backslash\{o_N\})]$$

The left side of the latter equation equals the right side of Eq. (7) and we can conclude:

$$\begin{aligned}\sum_{k=1}^{N-1} \frac{p_k^2}{p_N^2} q_{kN} &= \sum_{m=0}^{N-2}[(1 - p_N)\pi_{m+1}(O\backslash\{o_N\}) - \pi_{m+2}(O\backslash\{o_N\})] \\ &= \pi_1(O\backslash\{o_N\}) - \pi_N(O\backslash\{o_N\}) - p_N\sum_{m=1}^{N-1}\pi_m(O\backslash\{o_N\}) \\ &= 1 - p_N\sum_{m=0}^{N-1}\pi_m(O\backslash\{o_N\}) \overset{\text{Eq. (3)}}{=} 1 - q_{NN},\end{aligned} \tag{8}$$

where $\pi_1(O\backslash\{o_N\}) = 1 - p_N$, $\pi_N(O\backslash\{o_N\}) = 0$ and $\pi_0(O\backslash\{o_N\}) = 1$.

Finally, we have confirmed steady state probabilities $\rho_k = p_k^2 / \sum_{j=1}^{N} p_j^2$ and we already have computed the probabilities $\pi_m(O\backslash\{o_k\})$ that a hit on o_k is followed by a sequence of m requests without hit. We can combine both results to determine the mean number $E(R_k)$ of requests until the next request per state k. The hit rate h_{OPT} is reciprocal to the entire mean number $E(R) = \sum_{k=1}^{N} \rho_k E(R_k)$ of requests until the next hit. We obtain:

$$\begin{aligned}
&\Pr(R_k > m) = \pi_m(O\backslash\{o_k\}) \Rightarrow \\
&E(R_k) = \sum_{m=1}^{N} m \Pr(R_k = m) = \sum_{m=1}^{N} \Pr(R_k \geq m) = 1 + \sum_{m=1}^{N-1} \Pr(R_k > m) \\
&\quad = 1 + \sum_{m=1}^{N-1} \pi_m(O\backslash\{o_k\}) \\
&\Rightarrow h_{\mathrm{OPT}} = \frac{1}{E(R)} = \frac{\sum_{k=1}^{N} p_k^2}{\sum_{k=1}^{N} p_k^2 E(R_k)} = \frac{\sum_{k=1}^{N} p_k^2}{\sum_{k=1}^{N} p_k^2 (1 + \sum_{m=1}^{N-1} \pi_m(O\backslash\{o_k\}))}.
\end{aligned} \tag{9}$$

The entire computational complexity for the mean inter hit intervals $E(R_1)$, …, $E(R_N)$ and for the hit rate h_{OPT} is still $O(N^2)$. We first determine the series $\pi_1(O)$, …, $\pi_N(O)$ using Eq. (2), from which we obtain $\pi_1(O\backslash\{o_k\})$, …, $\pi_{N-1}(O\backslash\{o_k\})$ for each state k in linear effort $O(N)$ via backward computation using the reverted form of Eq. (1):

$$\pi_m(O\backslash\{o_k\}) = \pi_m(O) - \pi_{m-1}(O\backslash\{o_k\}) m p_k. \tag{10}$$

We consider a simple example with four objects and request probabilities $p_1 = 0.1$, $p_2 = 0.2$, $p_3 = 0.3$ and $p_4 = 0.4$. Then the probabilities for hits per object are 1/30, 4/30, 9/30 and 16/30. The mean number of requests until the next hit in each state are $E(R_1) = 2.564$, $E(R_2) = 2.252$, $E(R_3) = 2.028$ and $E(R_4) = 1.856$. Finally, the hit rate is $h_{\mathrm{OPT}} = 3000/5952 = 125/248 \approx 0.504$. For comparison, we have a gap of 20.4% between the optimum caching and the LRU hit rate $h_{\mathrm{LRU}} = \sum_k p_k^2 = 0.3$, while LFU stays in the middle $h_{\mathrm{LFU}} = \max_k(p_k) = 0.4$. Such differences are usual also for larger caches.

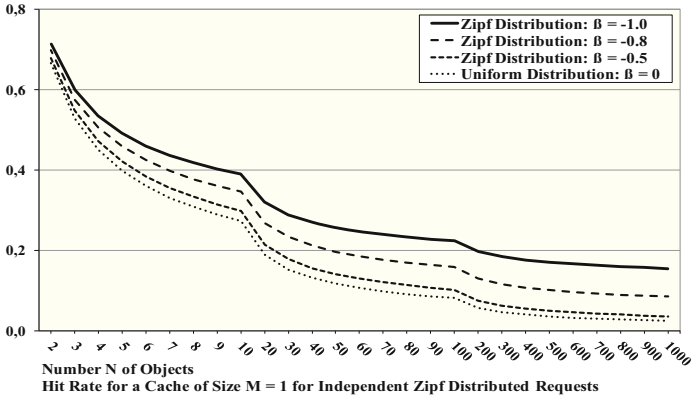


Fig. 1. Hit rate analysis for optimum caching

Figure 1 shows analysis results for Zipf distributed requests, see also Sect. 5. This includes uniform requests, i.e. $p_k = 1/N$ to all objects in the case $\beta = 0$, yielding Eq. (11):

$$\begin{aligned} \pi_m(O\backslash\{o_k\}) &= (N-1)(N-2)\cdots(N-m)N^{-m} \quad \Rightarrow \\ h_{\text{Opt}} &= 1/\sum_{k=0}^{N-2} N^{-k}(N-1)!/(N-k-1)! \end{aligned} \tag{11}$$

2.2 Comparison to Analysis Results for Optimum Caching in the Literature

Analytical approaches for optimum caching have been derived by Smith [18] and Knuth [13] over 30 years ago whereas the author is not aware of newer results.

A hit rate formula is derived by [13] for cache size $M = 2$ in the special case of uniform requests, whereas the case $M = 1$ seems to be ignored. However, a closer look reveals that [13, 18] assume that each requested object is always put into the cache, which is natural for local paging/caching systems. Then only a decision remains about which object should be replaced (cache "*replacement*" strategy [16]), whereas web caching strategies also can select the objects to be put into the cache.

Fortunately, it turns out that our hit rate results $h_{\text{WebCache}}(N, M)$ for web caches of size M and catalogue size N are directly transferrable to the hit rate of a local paging/cache system $h_{\text{PagingCache}}(N + 1, M + 1)$ [13, 18] for independent requests via the relationship:

$$h_{\text{PagingCache}}(N+1, M+1) = (1 + N h_{\text{WebCache}}(N, M))/(N+1). \tag{12}$$

A paging system reserves one of $M + 1$ cache positions for the most recently requested page and has cache size M left for unrestricted usage. Assuming independent requests, a hit on the most recently requested page is encountered with probability $1/(N + 1)$ or otherwise, with probability $N/(N + 1)$, the hit rate $h_{\text{WebCache}}(N, M)$ of a web cache of size M for N objects applies, excluding the recently requested object.

The resulting transformation of Eq. (12) yields a perfect match of results in [13, 18] with our results in the special case of uniform requests. In particular, Eq. (11) transforms into a main result for $f(2, d)$ derived by Knuth [13, p. 187] with d denoting the number of objects ($d \cong N$).

3 Markov Model for Optimum Caching Without Look-Ahead

When we consider a Markov model for arbitrary cache size, the state model has to include objects in the cache as well as all recently requested objects that would lead to a hit when addressed again in the next request. We design a model which collects all information from the past which is relevant for optimum caching decisions, but without look-ahead. State transitions are triggered per request.

3.1 Defining a General Markov Model for Optimum Caching

We assume that the cache has M positions and is fully occupied. A request to an object in the cache means a hit. Otherwise, a requested external object that is not in the cache may replace an object and thus acquires its cache position according to the optimum caching strategy, i.e. it will replace the object with the longest time until its next request, if the next request to the external object comes earlier.

The state S_T of the proposed Markov model after the T^{th} request r_T ($T \in \mathbb{N}$) is composed of a sequence I and M sets $L_1, \ldots, L_M$ of objects $S_T = (I, L_1, L_2, \ldots, L_M)$:

1. The sequence $I = (o_{k_1}, o_{k_2}, \ldots, o_{k_M})$ includes M *internal objects*, each of which caused the most recent hit in one of the M cache positions. Let $r_{T_1}, \ldots, r_{T_M}$ denote the requests for those hits. Then the internal objects $o_{k_1}, o_{k_2}, \ldots, o_{k_M}$ are ordered due to the request times $T \geq T_1 > T_2 > \ldots > T_M$. Each internal object o_{k_j} has been in the cache at its most recent hit at request r_{T_j}, but may be replaced later already before r_T.
2. Each set $L_j (j = 1, \ldots, M)$ consists of all objects, which have been requested between $r_{T_{j-1}}$ and r_{T_j} without a hit, where $r_{T_0} = r_T$. We refer to those objects as *loading options* because they may be loaded into the cache at their recent request and then would cause a hit if they would be requested again at r_{T+1}.

3.2 Properties of the State S_T

We presume and finally confirm two properties for the objects involved in the current state S_T:

(1) All loading options, i.e. objects in the sets $L_1, \ldots, L_M$ have been requested only once in the relevant time frame after r_{T_M}.
(2) S_T includes exactly those objects in $I, L_1, \ldots, L_M$, which lead to a hit due to the optimum caching principle, if one of the objects is requested again at r_{T+1}.

Next, we show that property (2) follows from property (1) and that vice versa property (1) can be confirmed for S_{T+1} from property (2) for S_T, i.e. when property (1) holds at the start for S_1 then both properties remain valid in the course of the Markov process.

Considering the next request r_{T+1}, we can distinguish three cases that r_{T+1} is addressing

1. an external object which is not included in S_T,
2. an internal object included in I, or
3. a loading option, i.e. an object in one of the sets $L_1, \ldots, L_M$.

1. If an external object (o^*) is requested at r_{T+1} then the most recent request to o^* was before r_{T_M}. After the request r_{T_M} there has been a hit in each cache position by an internal object. Consequently, o^* is not in the cache and r_{T+1} is a cache miss, because o^* has been replaced by an internal object and is not requested afterwards.

2. If an *internal object* o^* of the sequence I is requested at r_{T+1} then Belady's algorithm has not replaced o^* with a loading option. The assumption (1) that a loading option is requested only once after r_{T_M} means that the next request to the same loading option comes after r_{T+1}. Therefore Belady's algorithm gives o^* priority in the cache before all loading options. A loading option can replace another internal object (o_i), when the next request to o_i comes after r_{T+1}, but then o_i again cannot replace o^*, because of an earlier next request to o^* at r_{T+1}. Concluding, neither a loading option nor an internal object can replace an internal object with next request at r_{T+1}. Thus a request to an internal object at r_{T+1} leads to a new hit.
3. If a *loading option* o^* is requested at r_{T+1} then Belady's algorithm has put o^* into the cache at its previous request (r_{T^*}) in the considered time frame $T \geq T^* > T_M$, because at that time no more than $M-1$ other objects can have a next request before r_{T+1}. In particular, no external objects are requested between r_{T_M} and r_T and no loading option is requested again before r_{T+1}. Moreover, the internal object o_{k_M} is not requested between the time of its last hit at r_{T_M} and r_T, because a new request to o_{k_M} would lead to a new hit which would reset r_{T_M}. Therefore Belady's algorithm puts o^* into the cache, replacing o_{k_M} or another object with the longest interval until its next request. In the time span from r_{T^*} until r_T, o^* stays among the M objects with shortest interval until the next request and is kept in the cache due to Belady's criterion. Concluding, a request to a loading option at r_{T+1} leads to a hit.

Then a second request to a *loading option* always converts the loading option into an internal object, which received the most recent hit. In this way, property (1) is preserved over state transitions per request to the following state S_{T+1}.

3.3 Specification of Markov State Transitions per Request for Optimum Caching

Starting from $S_T = (I, L_1, \ldots, L_M) = ((o_{k_1}, o_{k_2}, \ldots, o_{k_M}), L_1, \ldots, L_M)$ we specify the state transitions, i.e. the following state S_{T+1} after the next request r_{T+1}. The transitions are performed according to the definitions of the sequence I, and the sets L_1, …, L_M in continuation for the next request r_{T+1}.

1. In case of a cache miss, the requested external object o^* becomes a new loading option of S_{T+1} and is added to the set L_1.

$$(Miss)\ o^* \notin \{o_{k_1}, \ldots, o_{k_M}\} \wedge o^* \notin L_1 \cup \ldots \cup L_M : S_{T+1} = (I, L_1 \cup \{o^*\}, L_2, \ldots, L_M)$$

2. If an internal object $o^* = o_{k_j} \in \{o_{k_1}, o_{k_2}, \ldots, o_{k_M}\}$ is requested then it is put in front of the sequence I as the internal object with the most recent hit. A new empty set is added to S_{T+1} for collecting loading options in upcoming requests after r_{T+1} until the next hit. Finally, the sets L_j and L_{j+1} are joined because the previous hit on o_{k_j} which

separated them is no longer relevant, i.e. o_{k_j} can no longer be replaced by a loading option in L_j after its new hit at r_{T+1}. The set L_M is removed if $j = M$:

$$(Hit_1) \quad o^* = o_{k_j} \in I, j \neq M: \\ S_{T+1} = ((o_{k_j}, o_{k_1}, \ldots, o_{k_{j-1}}, o_{k_{j+1}}, \ldots, o_{k_M}), \{\}, L_1, \ldots, L_{j-1}, L_j \cup L_{j+1}, L_{j+2}, \ldots, L_M)$$

$$(Hit_2) \quad o^* = o_{k_M} \in I: S_{T+1} = ((o_{k_M}, o_{k_1}, \ldots, o_{k_{M-1}}), \{\}, L_1, L_2, \ldots, L_{M-1})$$

3. If r_{T+1} is addressing a loading option (o^*) then r_{T+1} is a hit on o^* and o^* becomes an internal object, which is again put at the head of the sequence I.

Belady's algorithm will replace the object with the longest time until its next request, but information beyond the request r_{T+1} is not yet available for the Markov process. If $o^* \in L_j$ then o^* cannot replace the internal objects $o_{k_1}, \ldots, o_{k_{j-1}}$ because their last hit came after the previous request for o^*, when o^* was put into the cache. Therefore the object to be replaced is from the subset $R = (o_{k_j}, \ldots, o_{k_M})$.

As a transition rule for the Markov process, we preliminary choose the first internal object o_{k_j} in R to be replaced by o^*, but we still hold o_{k_j} as a loading option in S_{T+1}. In this way, the decision is kept open about which object to be replaced in R. If o_{k_j} is addressed at a later request r_{T+t} as a loading option this means that o_{k_j} is kept in the cache from r_T until a hit at r_{T+t} on account of another object in R, i.e. the replacement is then handed over to another object in R whose next request comes after r_{T+t}. If $o^* \in L_M$ then o_{k_M} is the only choice for replacement in favour of o^*.

$$(Hit_3) \quad o^* \in L_j; j \neq M: \\ S_{T+1} = ((o^*, o_{k_1}, \ldots, o_{k_{j-1}}, o_{k_{j+1}}, \ldots, o_{k_M}), \{\}, L_1, \ldots, L_{j-1}, L_j \cup \{o_{k_j}\} \cup L_{j+1}, L_{j+2}, \ldots, L_M)$$

$$(Hit_4) \quad o^* \in L_M: S_{T+1} = ((o^*, o_{k_1}, \ldots, o_{k_{M-1}}), \{\}, L_1, L_2, \ldots, L_{M-1})$$

Table 1 illustrates an example of the Markov process for optimum caching for size $M = 3$, where objects are identified by their index 1, …, 7. Each row of the table shows

- the object requested at r_T and the transition type ($Miss$, Hit_1, …, Hit_4),
- a representation of S_T with the objects $o_{k_1}, \ldots, o_{k_M}$ of the sequence I marked in boldface, and with each set L_j ($j = 1, \ldots, M$) being inserted between the objects $o_{k_{j-1}}$ and o_{k_j}, where empty sets are omitted,
- the cache content at r_T for Belady's optimum caching algorithm, which can be uniquely determined from "future requests" of the provided request sequence.

We start in the state $S_T = (o_1, o_2, o_3)$, which is entered e.g. after the request sequence o_1, o_2, o_3, o_3, o_2, o_1.

Table 1. Example of the Markov process with state transitions $S_T \rightarrow S_{T+1}$ per request

Object requested at r_T and (transition type)			State S_T represented as $L_1, o_{k_1}, L_2, o_{k_2}, \ldots, L_M, o_{k_M}$	Cache content for optimum caching at r_T
Initial cache content →			**1 2 3**	**1 2 3**
$T = 1$	4	(*Miss*)	{4} **1 2 3**	**1 2 4**
↓ 2	5	(*Miss*)	{5, 4} **1 2 3**	**1 2 4**
3	2	(Hit_1)	**2** {5, 4} **1 3**	**1 2 4**
4	6	(*Miss*)	{6} **2** {5, 4} **1 3**	**1 6 4**
5	1	(Hit_1)	**1** {6} **2** {5, 4} **3**	**1 6 4**
6	7	(*Miss*)	{7} **1** {6} **2** {5, 4} **3**	**7 6 4**
7	4	(Hit_4)	**4** {7} **1** {6} **2**	**7 6 4**
8	5	(*Miss*)	{5} **4** {7} **1** {6} **2**	**7 6 5**
9	5	(Hit_3)	**5** {4, 7} **1** {6} **2**	**7 6 5**
10	7	(Hit_3)	**7 5** {4, 1, 6} **2**	**7 6 5**
11	3	(*Miss*)	{3} **7 5** {4, 1, 6} **2**	**7 6 5**
12	6	(Hit_4)	**6** {3} **7 5**	**7 6 5**
13	7	(Hit_1)	**7 6** {3} **5**	**7 6 5**
14	5	(Hit_2)	**5 7 6**	**7 6 5**
…	…	…	…	…

3.4 State Space of the Markov Model

The considered Markov model comprises $N(N-1)\cdots(N-M+1)$ states for the ordered sequence I of M internal objects, which is combined with all distributions of $N-M$ remaining objects over the sets $L_1, \ldots, L_M$ and the set of external objects. The number of combinations for distributing $N-M$ different objects over $M+1$ multi sets is $(M+1)^{N-M}$. We conclude that the state space of the Markov model is of the size

$$\left|S_T\right| = N(N-1)\cdots(N-M+1)\cdot(M+1)^{N-M}.$$

For independent request (IRM) with non-zero request probabilities, each state can be entered with non-zero probability. Therefore we note that $I = (o_{k_1}, \ldots, o_{k_M})$ is achieved e.g. after the sequence $o_{k_1}, \ldots, o_{k_M}, o_{k_M}, \ldots, o_{k_1}$ of $2M$ requests. In this case, there are no loading options included such that property (1) is obviously fulfilled. Moreover, if the previous request sequence is continued by $o_{k_M}, L_M, \ldots, o_{k_1}, L_1$, with arbitrary sets of pairwise different objects being interspersed as loading options between requests to the internal objects, then any arbitrary state $S_T = (I, L_1, \ldots, L_M) = ((o_{k_1}, o_{k_2}, \ldots, o_{k_M})L_1, \ldots, L_M)$ will be entered. This confirms that the Markov model is irreducible and ergodic when requests are independent with non-zero probabilities, i.e. the full state space of the size $|S_T|$ is exploited.

The analysis approaches for optimum caching in [13] is focusing on uniform requests. Therefore the approach by Smith [18, pp. 745–747] seems to be the only

comparable general optimum caching model. It differs by including future references, whereas the previously presented Markov model is based only on past requests without look-ahead. The size of Smith's model is much larger for small caches [18]:

$$|S_{Smith}| = N!\binom{N}{M} = N(N-1)\cdots(N-M+1)\cdot N\cdots(M+1)$$
$$= \frac{N(N-1)\cdots(M+1)}{(M+1)^{N-M}}|S_T|.$$

Table 2 shows analysis examples for small caching systems. Independent requests are assumed with probabilities $p_1 = 0.1$, $p_2 = 0.2$, $p_3 = 0.3$ and $p_4 = 0.4$ for $N = 4$ and with $p_1 = 0.1$, $p_2 = 0.15$, $p_3 = 0.2$, $p_4 = 0.25$ and $p_5 = 0.3$ for $N = 5$, respectively.

Table 2. Analysis examples: hit rate results and model size $|S_T|$, $|S_{Smith}|$

N	Parameters	$M = 1$	$M = 2$	$M = 3$	$M = 4$
4	$h_{OPT}; h_{LFU}; h_{LRU}$ $\|S_T\| / \|S_{Smith}\|$	50.4%; 40%; 30% 32 / 96	76.5%; 70%; 58% 108 / 144	91.9%; 90%; 82.6% 96 / 96	
5	$h_{OPT}; h_{LFU}; h_{LRU}$ $\|S_T\| / \|S_{Smith}\|$	42.8%; 30%; 22.5% 80 / 600	66.5%; 55%; 44.3% 540 / 1200	82.3%; 75%; 65% 960 / 1200	93%; 90%; 84% 600 / 600

3.5 The Case of Uniform Requests

For uniform requests, the Markov model can be simplified. Actually there is no need to distinguish objects because the next request addresses each object with probability $1/N$. The simplified Markov model still refers to M internal objects and a variable number of up to $N - M$ loading options. Let $r^*_{T_1}, \ldots, r^*_{T_M}$ denote the requests of the most recent hits to all internal objects and let $n^*_1, \ldots, n^*_M$ denote the numbers of requests to pairwise different objects, which are encountered between $r^*_{T_{k-1}}$ and $r^*_{T_k}(r^*_{T_0} = r_T)$. Then $S_T = (n^*_1, \ldots, n^*_M)$ is sufficient to characterize the current state for uniform requests. This special case leads to $\binom{N}{M}$ states. We applied a steady state Markov analysis in the simplified state space to several examples for uniform requests given in [13, 18], yielding the same results when transformed via Eq. (12).

3.6 The Case $M = N - 1$

The Markov analysis of optimum caching for a fixed number of objects with IRM requests can also be simplified when almost all objects fit into the cache. For $M = N - 1$, cache misses are rare. When a cache miss happens, optimum caching can select one out of N objects to be removed from the cache, i.e. the one with the longest time until its next request. Then the next cache miss is encountered at the first instant when all N objects have been requested again and the cache miss rate is the reciprocal of the

mean time until that instant. Knuth [13] solved the mean inter cache miss time $\bar{T}_{Miss}$ and the hit rate as a coupon collection problem for uniform requests:

$$\bar{T}_{Miss} = N(1 + 1/2 + 1/3 + \cdots + 1/N); \quad h = 1 - 1/\bar{T}_{Miss}.$$

The solution can be generalized for independent requests with arbitrary probabilities p_j for each object. Therefore we compute the probabilities $p(\{o_{j_1}, o_{j_2}, \ldots, o_{j_K}\})$ that a subset of K objects gets the next requests after a cache miss and the mean time $\bar{T}(\{o_{j_1}, o_{j_2}, \ldots, o_{j_K}\})$ until all K objects of the subset have been requested. The values for subsets of size K are iteratively computed from subsets of size $K - 1$:

$$p(\{o_{j_1}, o_{j_2}, \ldots, o_{j_K}\}) = \sum_{h=1}^{K} p(\{o_{j_1}, o_{j_2}, \ldots, o_{j_K}\}/\{o_{j_h}\})p_{j_h}; \; p(\{\}) = 1; \; p(\{o_j\}) = p_j;$$

$$\bar{T}(\{o_{j_1}, o_{j_2}, \ldots, o_{j_K}\}) = \frac{1}{p(\{o_{j_1}, o_{j_2}, \ldots, o_{j_K}\})} \sum_{h=1}^{K} p(\{o_{j_1}, o_{j_2}, \ldots, o_{j_K}\}/\{o_{j_h}\})p_{j_h} \cdot$$
$$(\bar{T}(\{o_{j_1}, o_{j_2}, \ldots, o_{j_K}\}/\{o_{j_h}\}) + 1/(1 - \sum_{\substack{l=1 \\ l \neq h}}^{K} p_{j_l}))$$

In this way, we can compute the mean time between cache misses via 2^N subsets.

4 Comparing Belady's Algorithm with the Markov Approach

The steady state Markov analysis for optimum caching is feasible only for small size systems assuming independent requests, but we can follow the Markov process by one state transition per request to evaluate the cache hit rate as an alternative to Belady's algorithm [2]. Therefore we take a closer look at efficient implementation options for Belady's algorithm and for the Markov approach.

4.1 Implementation of Optimum Caching Based on Belady's Algorithm

Belady's principle for optimum caching decides about the cache content based on the time until the next request to each object. In case of a cache miss, the requested object is pushed into the cache, if its next request comes earlier than the next request to a cached object, replacing the object with longest time until its next request.

Therefore Belady's principle requires a sorted list of the objects in the cache according to the time index of their next request in the future request sequence r_T. We assume that r_T is available for $1 \leq T \leq T_{\max}$ as a list or an array, provided from a request trace that has been monitored on a web platform or by a random generator, e.g., according to an independent request model [9].

Together with the currently requested object (o_k), we also store the index $T^+(o_k)$ of the next request to the same object o_k, where $T^+(o_k) = T_{\max} + 1$ if o_k isn't requested anymore. $T^+(o_k)$ is required to decide whether to push an external object into the cache and to insert it into the sorted list of cached objects. In case of a hit, the time of the next request to the object is updated and the object is reinserted into the sorted list. $T^+(o_k)$ is determined for in a scan through all requests with effort proportional to $T_{\max}$.

Figure 2 illustrates the data sets for performing Belady's algorithm. Most of the effort in Belady's algorithm is then required for inserting objects into the sorted list of cached objects e.g. with heapsort [7, 14] at $O(\log(M))$ complexity per request.

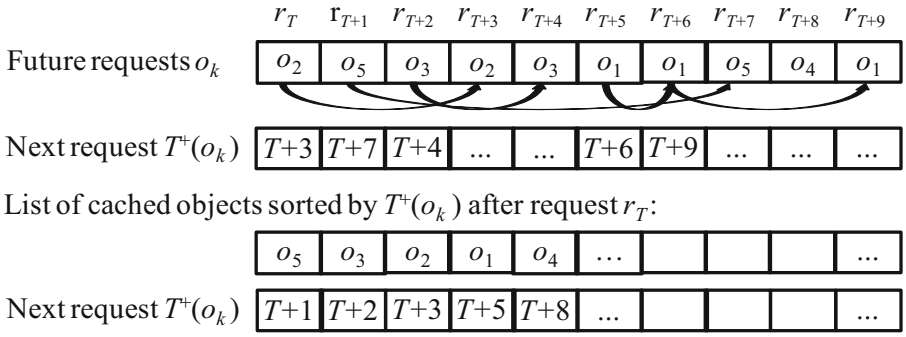


Fig. 2. Data structures for efficient support of Belady's algorithm

4.2 Implementation of Optimum Caching Based on the Markov Process

In order to follow the Markov process, a state transition has to be performed per request according to the cases (*Miss*, Hit_1, …, Hit_4) of the specified transition rules. We represent the state S_T as a double linked list, which includes all loading options and internal objects in the sequence of their most recent requests, corresponding to the middle column $L_1, o_{k_1}, L_2, o_{k_2}, \ldots, L_M, o_{k_M}$ of Table 1, with a binary marker for internal objects. Then updates per request are handled similar as in a stack for LRU caching [11], with requested objects being put on top and objects being replaced at the bottom.

- In particular, for a cache miss, i.e. a request to an external object, the only update is to add the external object as a loading option on top to the set L_1.
- In case of a hit on an internal object o_{k_j}, the object is removed from its current position in the double chained list and also put on top. In this way, L_j is merged with L_{j+1} except for the case $j = M$, where L_M is discarded.
- In case of a hit on a loading option $o^* \in L_j$, the object o^* is removed from its current position and pushed on top as an internal object. Vice versa, o_{k_j} is now marked as a loading object, where L_j is merged with L_{j+1} including o_{k_j}. In case $j = M, o_{k_M}$ and L_M are discarded.

The effort per request is constant for moving an object to the top. Discarding of objects in L_M also has constant effort in the mean, because no more than one object is added per request and therefore the mean number of discards is also less than one. More effort is needed for searching the next internal object after a requested loading option $o^* \in L_j$. We support the search by redirecting "next" pointers in the double linked list from the loading options o^* to the following internal object. Then the next internal object is found in a few steps following those pointers. In total, we experience similar update effort for efficient implementations of Belady's algorithm and the Markov approach. LRU/LFU updates are faster at constant effort per request [17].

As the main advantage and new insight from the Markov state model, no look-ahead is required to decide about optimum caching hits, in contrast to Belady's approach [2] "The optimal solution can be found by storing the program's entire sequence of references and then working backward to reconstruct a minimum replacement sequence".

The Markov process indicates in each request, whether the currently requested object is a hit and thus has been loaded previously for optimum caching. What remains open is an uncertainty about which current loading options already have replaced other objects in the cache to enable the next hits. In usual applications with unknown future request sequence, the Markov approach evaluates optimum caching on the fly, without having to wait until the complete request sequence is available for preprocessing.

5 Evaluation of Optimum Caching for Zipf Requests

We evaluate hit rates of optimum caching in comparison to least recently used (LRU) and least frequently used (LFU) caching strategies. LRU is a basic method with simple implementation, whereas LFU is known to maximize the hit rate for independent requests. We assume independent Zipf distributed requests as confirmed in many studies for web platforms [3, 10, 11] with Zipf parameter $\beta = -0.5$. For a Zipf distribution, the object in rank r in the order of highest popularity has request probability

$$z(r) = \alpha r^{\beta} \quad \text{for } r = 1, \ldots, N, \text{with normalization } \alpha = z(1) = 1/\sum_{r=1}^{N} r^{\beta}.$$

We evaluate systems with catalogue size N in the range 10^3, …, 10^6 and cache sizes M, which are varied from 0.1%–50% of the number N of objects in Fig. 3.

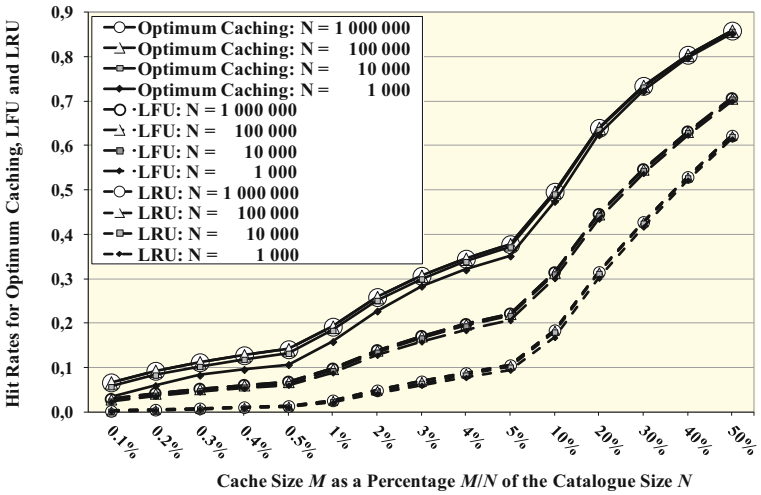


Fig. 3. Cache hit rates for Zipf law requests with $\beta = -0.5$

Each simulation runs over at least 10^8 requests, where a cache filling phase at the start is ignored. Results of the Markov approach are always exactly matching those of Belady's algorithm. For $N \geq 1000$, the hit rate mainly depends on the fraction M/N of objects in the cache. LRU, LFU and optimum caching hit rates show clearly separated curves extending results in [9, 11], where clairvoyant optimum caching can improve LFU up to 15–20% and LRU even up to 30%.

6 Limited Look-Ahead Options for Caching

The previous results show a significant gain of optimum caching hit rates over other methods with regard to future requests. Partial knowledge about upcoming requests is often available and useful for prediction and prefetching [1, 5].

Within this scope, limited look-ahead options can be exploited when a fixed number of upcoming requests $r_{T+1}, \ldots, r_{T+K}$ are known. Decisions about new cache content are taken at cache miss events. There is some transfer delay until new data is retrieved from a server or higher layer cache. Therefore the decision whether to put it into the cache or to forward it to the user without being cached, can wait until the arrival of the retrieved data, which enables for a look-ahead of at least a few hundred milliseconds.

The limited look-ahead can be exploited by combining an arbitrary "usual" caching strategy with Belady's algorithm or alternatively, with the proposed Markov analysis. Objects which are foreseen to have a cache hit within the look-ahead period are handled as a preferred class according to Belady's algorithm while all other objects are handled by the "usual" strategy without replacing objects of the preferred class.

Such a combined method may improve caching efficiency in large CDNs facing huge request workloads. Wikipedia has to handle peaks of over 50 000 requests per second [10, 19] and a peak load above 30 million requests per second is reported for Akamai's CDN [6]. A much longer look-ahead is relevant for caching of video streaming, when it takes minutes until requested data is cached in parallel to viewing the video. Then an initial decision to replace a cached video by a new one can still be changed as soon as new requests to the cached video become visible e.g. after 10–20 s, while only a small portion, e.g. 10%–20%, of the data chunks have been overwritten at that time.

7 Conclusions

The steady state Markov analysis for optimum caching requires a larger state space than for LRU and LFU caching methods. For independent requests (IRM), a simple new optimum caching hit rate solution is derived for the cache size $M = 1$. Then the hit rate per object is proven to be proportional to the square of the object's request probability. We introduce a general Markov model for optimum caching with arbitrary cache size, but feasible steady state IRM solutions beyond $M = 1$ seem to be restricted to a small number N of cachable objects and to the case $M = N - 1$.

However, the transient Markov analysis provides an alternative to Belady's algorithm for optimum caching hit rates with comparable computational effort per request. While Belady's method presumes the complete reference sequence to be available, the Markov states indicate each hit or miss per request without look-ahead or preprocessing.

Acknowledgements. This work has received funding from the European Union's Horizon 2020 research and innovation programme in the EU SSICLOPS project <www.ssiclops.eu> under grant agreement No. 644866. This work reflects only the author's views and the European Commission is not responsible for any use that may be made of the information it contains. The author would like to thank the anonymous reviewer, whose detailed report allowed to improve and correct several parts of this work.

References

1. Beckmann, N., Sanchez, D.: Maximizing cache performance under uncertainty. In: Proceedings of the 23rd Symposium on High Performance Computer Architecture (HPCA) (2017)
2. Belady, L.A.: A study of replacement algorithms for a virtual-storage computer. IBM Syst. J. **2**, 78–101 (1966)
3. Breslau, L., et al.: Web caching and Zipf-like distributions: evidence and implications. In: Proceedings of the IEEE Infocom (1999)
4. Che, H., Tung, Y., Wang, Z.: Hierarchic web caching systems: modeling, design and experimental results. IEEE JSAC **20**(7), 1305–1314 (2002)
5. Famaey, J., Iterbeke, F., Wauters, T., De Turck, F.: Towards a predictive cache replacement strategy for multimedia content. J. Netw. Comput. Appl. **36**(1), 219–227 (2013)
6. Gilmore, P.W.: Akamai & ISPs, Presentation UKNOF25 (2013). slideplayer.com/slide/10588098
7. Guo, J., Garzarán, M.J., Padua, D.: The power of Belady's algorithm in register allocation for long basic blocks. In: Rauchwerger, L. (ed.) LCPC 2003. LNCS, vol. 2958, pp. 374–389. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24644-2_24
8. Hasslinger, G., Hartleb, F.: Content delivery and caching from a network provider's perspective. Comput. Netw. **55**, 3991–4006 (2011). Special Issue on Internet based Content Delivery
9. Hasslinger, G., Ntougias, K., Hasslinger, F.: Performance and precision of web caching simulations including a random generator for Zipf request pattern. In: Remke, A., Haverkort, B.R. (eds.) MMB&DFT 2016. LNCS, vol. 9629, pp. 60–76. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31559-1_7
10. Hasslinger, G., Kunbaz, M., Hasslinger, F., Bauschert, T.: Web caching evaluation from wikipedia request statistics. In: Proceedings of the 15th Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt 2017), Paris, France (2017)
11. Hasslinger, G., Ntougias, K., Hasslinger, F., Hohlfeld, O.: Performance evaluation for new web caching strategies combining LRU with score based object selection. Comput. Netw. **125**, 172–186 (2017)
12. Jain, A., Lin, C.: Back to the future: leveraging Belady's algorithm for improved cache replacement. In: Proceedings of the 43rd Annual International Symposium on Computer Architecture (ISCA) (2016)

13. Knuth, D.E.: An analysis of optimum caching. J. Algorithms **6**, 181–199 (1985)
14. Lee, D., et al.: LRFU: a spectrum of policies that subsumes the least recently used and least frequently used policies. IEEE Trans. Comput. **50**(12), 1352–1361 (2001)
15. Megiddo, N., Modha, S.: Outperforming LRU with an adaptive replacement cache algorithm. IEEE Comput. **37**(4), 58–65 (2004)
16. Podlipnik, S., Böszörmenyi, L.: A survey of web cache replacement strategies. ACM Comput. Surv. **35**(4), 374–398 (2003)
17. Shah, K., Mitra, A., Matani, D.: An O(1) algorithm for implementing the LFU cache eviction scheme. Technical report (2010). available via dhruvbird.com/lfu.pdf or en.wikipedia.org/wiki/Least_frequently_used
18. Smith, A.J.: Analysis of the optimal, look-ahead demand paging algorithms. SIAM J. Comput. **5**(4), 743–757 (1976)
19. Wikipedia statistics and information. https://meta.wikimedia.org/wiki/Wikimedia_servers, https://wikitech.wikimedia.org/wiki/Global_traffic_routing

Intrusion Detection for Sequence-Based Attacks with Reduced Traffic Models

Benedikt Ferling[1], Justyna Chromik[2(✉)], Marco Caselli[3], and Anne Remke[1,2]

[1] Westfälische Wilhelms-Universität Münster, Münster, Germany
anne.remke@uni-muenster.de
[2] University of Twente, Enschede, The Netherlands
{j.j.chromik,a.remke}@utwente.nl
[3] Siemens AG, München, Germany
marco.caselli@siemens.com

Abstract. Securing control networks (e.g. for power and gas distribution) requires dedicated approaches. Sequence-aware intrusion detection models the network traffic under normal operation to identify malicious behavior. Unfortunately, such models are often large and difficult to handle. This paper proposes a method that generates smaller traffic models and discusses the accuracy of those reduced models in the context of a real control infrastructure employing the IEC 60870-5-104 protocol.

1 Introduction

Supervisory Control and Data Acquisition (SCADA) systems are used to monitor critical infrastructures [1], by automating communication and control of, e.g., power distribution. As shown by the US grid hack [2] and the Ukrainian grid hack [3], both causing power outages that lasted for several hours, SCADA systems are vulnerable and can cause serious damage if not properly secured.

Stuxnet [4] has demonstrated that hackers can strike critical infrastructures by directly hitting the physical process misusing internal process-knowledge. Attacks of this kind are commonly known as *semantic attacks* and are usually of higher complexity compared to standard cyber-attacks. Among all semantic attacks, *sequence attacks* can strike the infrastructure by just misplacing perfectly legal messages within a communication stream [5].

Traditional Intrusion Detection Systems (IDS) identify malicious traffic in different ways: whitelisting [6], stateful approaches [7,8], and specification-based approaches [9] exist. In contrast, we focus on the detection of so-called sequence attacks, which employ perfectly legal messages arranged in an unforeseen order and make it difficult for an IDS to detect the malicious activity.

Sequence-based intrusion detection relies on the regularity of traffic that is present in SCADA traffic and does not fit networks with large traffic variety, e.g. the Internet. However, industrial control systems show regular and consistent communication patterns [10], which can be used e.g. for anomaly detection [11]. Hence, analyzing the communication using a probabilistic state-based approach,

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 53–67, 2018.
https://doi.org/10.1007/978-3-319-74947-1_4

which keeps track of message ordering, is a solution to identify sequence attacks in control networks [12].

Sequence-based and state-based approaches close to the presented work can be found in [5] and [13], respectively. However, such analysis may be time-consuming, depending on the complexity of the communication within the network and hence, the size of the resulting models. This paper aims to reduce this complexity, while keeping the detection accuracy for most types of sequence attacks. The approach shown in [5] may result in large Discrete-time Markov Chains (DTMCs), which however contain states which represent almost identical system states. We change the generation algorithm, such that the resulting models are considerably smaller, leading to lower computation times, as well. This is done by combining states with overlapping information or by abstracting from specific packet information. We investigate the accuracy of the resulting models by performing 9 different experiments on real traffic from a Dutch gas facility which uses the protocol IEC 60870-5-104, also known as *IEC-104* [14]. IEC-104 is widely used in industrial system control, especially in the context of energy grids [15].

The protocol supports the use of *Information Objects* (IOs) to classify data. Individual packets may transfer several Information Objects, addressed using *Information Object Addresses* (IOAs). This possibly results in ranges of IOAs per packet. This paper shows that combing states that represent (almost) similar packets with overlapping IOAs and abstracting from the IOAs altogether results in much smaller traffic models. Clearly, when comparing real traffic with the previously developed traffic models, smaller models will result in smaller computation times. Our case study shows that reduced models may also decrease the number of false positive alerts.

The paper is organized as follows. Section 2 explains the basics of SCADA and IEC-104. Section 3 explains the reduction methods for DTMCs and our approach is validated in Sect. 4. Section 5 concludes the paper.

2 SCADA Systems

This section briefly introduces SCADA systems and the IEC-104 protocol.

2.1 SCADA

SCADA systems connect *field stations*, which directly control the physical process and the *control room*, that monitors the state of the system, as shown in Fig. 1. The *field station* hosts sensors, actuators and Programmable Logic Controllers (PLCs). *Sensors* measure and transmit data of the monitored process over serial wires to PLCs. *Actuators* influence the systems behavior, by, e.g., opening or closing a switch connecting a power line. *PLCs* collect the field data and send it to the data acquisition server via the communication system. They also receive commands from the control network, which are then executed on

the appropriate actuator. In the *control room*, data of the field network is evaluated and appropriate actions are triggered. The *data acquisition server* collects, stores and distributes process data.

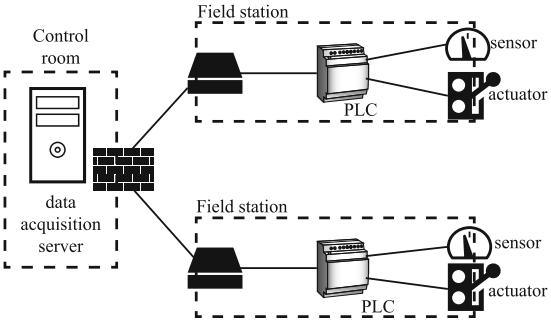


Fig. 1. SCADA in power distribution

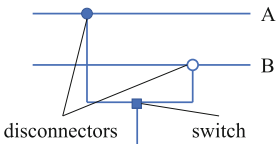


Fig. 2. Rail switching system

2.2 IEC-104 Protocol

The IEC-104 protocol is used for communication between field stations and control room [15]. It operates on top of TCP/IP. An IEC-104 packet consists of the *application protocol control information* (APCI) and the *application service data units* (ASDU) which together are called the *application protocol data unit* (APDU). APCI determines whether the packet has U-, S- or I-format. U-format initiates and terminates sessions between two devices, while the S-format acknowledges the received data. We focus on the I-format, which contains relevant information in the ASDU fields. The most important fields for our work are:

- *Type identification (TypeId)* identifies a function that the device should execute, e.g., 103 → *clock synchronization command* or 102 → *read command*.
- *Number of objects* defines the number of *information objects* found in the ASDU. This can be more than one. The *TypeId* is assigned to all these information objects inside of that packet.
- *ASDU address fields (ASDU-Address)* contains the address which all objects of the ASDU refer to.
- *Information object address fields (IOA)* refers to a specific information object, such as a reading from an element (voltage, current), a state of an element (on/off), or threshold setting. A packet may have up to 127 information objects. Multiple IOAs may come within the same response, e.g., IOAs from 1–5, and IOAs from 11–13.

While each vendor has its own implementation, the general structure follows the specification in [16] and [17].

2.3 SCADA Sequence Attacks

This section explains the concept of sequence attacks and provides an example inspired by the large power outage in North Holland on March 27th 2015 [18]. Sequence attacks are specific to industrial control systems and potentially harm a system by sending valid messages or commands, which are misplaced or out-of-order [5]. To take control of the process, an attacker can either reprogram a PLC or directly control the process from the network, e.g., by taking control over the communication channel. When controlling the process, an attacker sends commands in an order or timing not intended by the process. This potentially harmful sequence of commands is then called sequence attack [5].

The concept of *Interlocks* describes constraints on the execution of commands. In power distribution, this applies e.g. to the order in which power switches and disconnectors are used. According to IEC 60947-3, whenever a power line has to be disconnected, first, the power switch disconnects, which turns off the current on the power line. Only then, the disconnector is used to physically isolate the power line. Otherwise, a potentially dangerous electric arc is created. In order to connect a power line, first, the disconnector has to be connected, before the switch is closed.

Figure 2 shows rails A and B, which are used interchangeably depending on the switches' configuration. The disconnector on rail A and the switch are closed, while the disconnector on rail B is open. Hence, the power line at the bottom is connected to rail A. To switch from rail A to B, first the switch opens, then the disconnector on rail A opens. Next, the disconnector on rail B closes and then the switch closes.

In case of the outage that happened in the North Holland, the disconnector did not entirely connect before the power switch was turned on. This caused a short circuit, which resulted in a power outage of several hours for more than a million households and disruptions on one of the major European airports [19]. Although the incident was caused by a technical and human error, a similar situation would arise by sending legitimate commands in the wrong order to the PLC controlling the switches and disconnector. Although most of the PLCs do check said interlock internally, some operators perform this check at the central control room. Hence, if an attacker gains control over the communication channel to the remote PLC directly, the constraint check will not be performed. Even if interlocks are properly configured, any attempt to switch in a wrong order should be reported to the operator and a sequence-based IDS would fit the purpose.

3 Message Sequences

We explain the IEC-104 traffic model, sequence attacks and the performed reductions.

3.1 Representing Traffic Sequences as DTMCs

Following the approach presented in [5] traffic is represented as a sequence of exchanges in terms of a Discrete Time Markov Chain (DTMC). In the following

Data: Sequence of Events
Result: DTMC representing the sequence of events
1 **for all** $e_{t_n} \in sequence$ **do**
2 $State_{DTMC} \leftarrow$ extractAttributes(e_{t_n});
3 **if** $State_{DTMC} \in DTMC$ **then**
4 update($State_{DTMC}$, DTMC);
5 **else**
6 add($State_{DTMC}$, DTMC);
7 **end**
8 **if** $Transition_{previousState,State_{DTMC}} \in DTMC$ **then**
9 update($Transition_{previousState,State_{DTMC}}$);
10 **else**
11 add($Transition_{previousState,State_{DTMC}}$, DTMC);
12 **end**
13 $previousState \leftarrow State_{DTMC}$
14 **end**

Algorithm 1: DTMC modeling of sequences as in [5]

we use DTMC that can be defined as a tuple $\mathcal{M} = (S, T)$, where S is a finite set of states and T is a transition relation, that assigns probabilities to states $(s_1, s_2) \in T$, i.e. if there exists a transition between the states s_1 and s_2. Note that the sum of the outgoing transition probabilities per state always equals one.

States in the DTMC reflect the kind of communication that takes place. Hence an event in the sequence, i.e., the transmission of a packet, is associated with a state. Transitions model the choice between successor packets together with the respective probability of this event taking place. We transform the network traffic traces into time-ordered list of events. This paper only considers the communication between two devices and therefore an *event* e_{t_n}, which takes place at time point $t_n \in \mathbb{R}^+$ is defined as a triple <Direction, Address, Service>, which takes values from the IEC-104 specification, as follows:

- **Direction** either takes the value 'request' or 'response',
- **Address** contains the 'ASDU-address' and the 'IOAs',
- and **Service** is the 'Type-Id' that identifies a function.

A *sequence* (l) sorts events according to their time of occurrence from old to new. It is defined as a time ordered list of events e_{t_n}, such that $t_n < t_{n+1}$ for $n \in \mathbb{N}$. Algorithm 1 then builds a Discrete Time Markov Chain traffic model from a sequence of events, abstracting from possibly different inter-event times, in the following five steps:

S1 loops over all events in the sequence (c.f. Algorithm 1 l.1–14), processing its events.
S2 extracts the attributes of an event and stores them in variable '$State_{DTMC}$' (c.f. l. 2). The state to which an event leads is defined by: Request, Response, ASDU-Address, IOAs, TypeId and the 'control field format'.

S3 checks whether that state is already present in the DTMC. Then the counter indicating how often that state has been visited is increased in the corresponding state. Otherwise, a new state is added to the DTMC in line 6.
S4 updates the transitions. If the transition to '$State_{DTMC}$' is part of the DTMC the transition probabilities and the transition counter are updated (line 9). If the transition is new, line 11 adds a transition from 'previousState' to '$State_{DTMC}$' to the DTMC.
S5 updates the variable 'previousState' in line 13 with the value created in line 2.

3.2 Sequence Attacks

This section describes sequence attacks detected by comparison with a DTMC built on benign traffic. For example, reconsider the combination of switch and disconnector (c.f. Section 2.3), as presented in Fig. 3. Legal commands for the switches are 'open' and 'close'. Let the initial state of the system be an open switch and a closed disconnector.

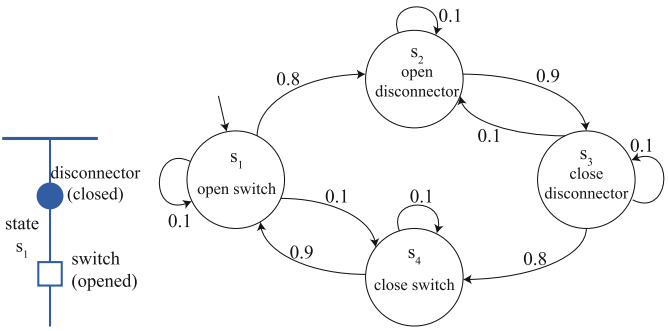


Fig. 3. Scenario with single switch and disconnector

The DTMC traffic model has four states: $S = \{s_1, s_2, s_3, s_4\}$, where

- s_1 models that a command to open the switch is sent,
- s_2 opens the disconnector,
- s_3 commands to close the disconnector, and
- s_4 requires to close the switch.

Those states and the corresponding transition probabilities are shown in Fig. 3. As discussed before, a command to open the disconnector should always be preceded by the corresponding command to open the switch. Vice-versa, the disconnector should always be closed before the switch is closed. The only bidirectional transitions between states are between open and close disconnector and open and close switch. While it may occur that a switch is, e.g., immediately

closed after being opened, this will not occur often and hence has a low transition probability. Furthermore, each state is equipped with a self-loop that happens with a relatively low probability. This corresponds to the same command being sent multiple times, which can legally happen, e.g., due to packet retransmission. Recall that IEC-104 runs on top of TCP, which can cause retransmissions due to preliminary timeouts or the loss of acknowledgements. In the following we distinguish between three types of violations:

1. **New transition violation** stems from a valid packet that is however not expected in the sequence of commands [5]. Consider the DTMC is in state s_4 of the switch example, i.e., the last packet contained the command to close the switch. If the next command would request to open the disconnector a *new transition violation* is encountered, as the DTMC model does not contain a transition that corresponds to this sequence of commands, namely *close switch* succeeded by *open disconnector*. An alert would be issued to the operator in this case to warn about a potential intrusion. Figure 4 shows the violation as a red dashed line.
2. **New state violation** occurs when an unexpected command is sent to the controller. In our example, the switch could receive the command to change some threshold value. This is a legitimate command, which however does not occur often and has not been part of the traffic used to train the DTMC. Hence, there is no state that corresponds to this command. Figure 5 shows the violation as a red dashed line and state.
3. **Anomalous transition frequency**, a so-called timing-violation occurs when a single transition is used too often. The commands arrive in an expected order, however they occur with a probability that deviates from the transition probability more than a certain predefined threshold. To achieve this, a parallel DTMC model is trained with the current sequence and compared to the previously trained model after each event. For example if the switch is opened and closed repeatedly, the transition probability between *open switch* and *close switch* will grow to exceed the transition probability of 0.1 in the originally trained DTMC. Hence, an anomaly is observed, and an alarm is issued, which could prevent the hardware from being harmed. Figure 6 shows the DTMC trained from the current sequence. The transition probabilities that differ from the original DTMC are indicated as red dashed lines.

3.3 DTMC Reduction

The traffic models that result from applying Algorithm 1 to realistic scenarios can be very large, as shown in [17]. However, many states of the model differ for just a state parameter [5]. In the case of IEC-104 traffic we can leverage this by combining states with overlapping IOAs and by completely abstracting from the information contained in IOAs. Consider a traffic capture where the first message is a server request asking for *reading* IOs with addresses 1–10, in the

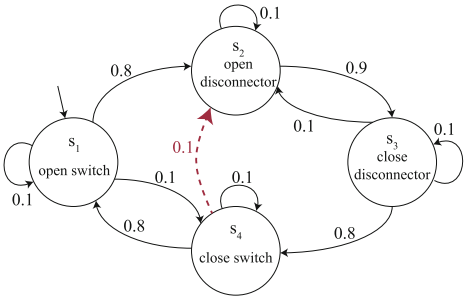


Fig. 4. DTMC with a transition violation

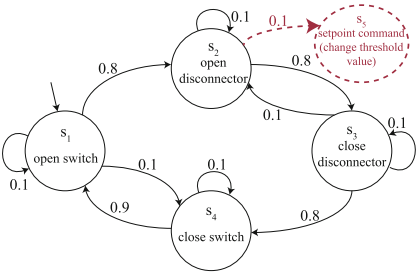


Fig. 5. DTMC with a state violation

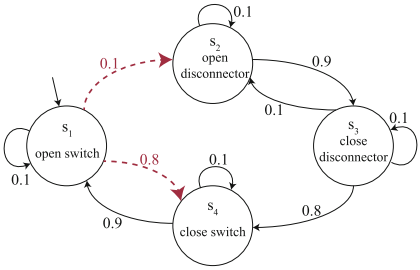


Fig. 6. DTMC with a probability violation

second message it requests readings from addresses 6–10, in the third message it asks for addresses 1–5, and in the last message it reads from address 125. The original approach explained in [5] would create four different states for these four different read requests. Using the original implementation, we noticed that many new states appear because a request is sent to a different subset of IOAs.

We therefore compare DTMCs built according to Sect. 3.1 without reduction with two types of reduced DTMCs, namely, (i) where states with *overlapping* IOAs are merged and (ii) where information contained in IOAs is not taken into account for differentiating states. While the reference case corresponds to the approach in [5], the first reduction case relies on the observation that the SCADA server asks for various IOAs, although the function (*TypeId*, c.f. Section 2.2) remains the same. In the example above, the first reduction approach corresponds to creating two states instead of four: one for IOAs 1–10, and a second for IOA 125. The second reduction case presents a more radical reduction approach, which completely abstracts from the IOAs and only takes the information 'Direction', 'Service' and 'ASDU address' into account.

For some functions like the *read command*, the process does not suffer if the reading is performed at a different place in the sequence, and merging all IOAs would greatly reduce the sizes of the DTMCs. In the mentioned example, this would mean that all reading commands refer to a single state.

We have generated traffic models from a SCADA trace obtained at a Dutch gas facility, and in the following we show the resulting DTMCs for the two private IP addresses 172.31.1.100 and 172.31.8.170. The trace consists of 10 days of traffic captured in 2011. All traffic models presented in this section have been generated using the entire available traffic capture using Algorithm 1, which has been changed slightly to implement also the reductions *overlapping* and *all*. Figure 7 shows the traffic models for the approach that takes into account IOAs in full detail and Fig. 8 shows the traffic model that does not take into account information about IOAs. The model that results from combining states with overlapping IOAs is shown in Fig. 9. Figure 7 shows a large number of transitions and states representing events in which various subsets of the same IOAs have been requested within the same function. We chose to present this graph in a small scale just to give an idea of the size and structure of this model. For better readability, Fig. 9 has been enlarged[1]. Each state is marked with some color, and contains information on the *Direction*, *TypeID*, *ASDU address*, *IOAs* which it includes and the *count* of how many packets of this type have been observed. Each transition is labeled with its probability. Reduction *all* then merges all states with the same color as shown in Fig. 8[2].

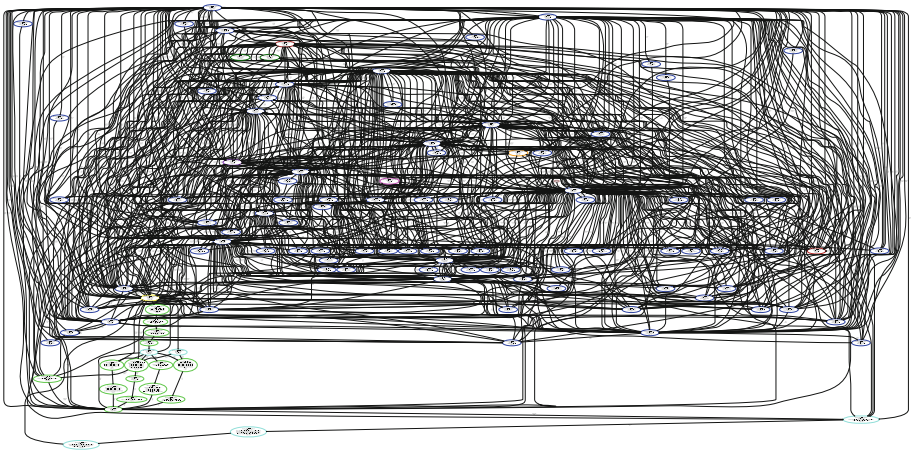


Fig. 7. Original DTMC

Figures 7, 8 and 9, show that the number of states and transitions reduces considerably, when (partly) abstracting from the IOAs. The number of states in the DTMC reduces from 117 to 17 when merging the overlapping IOAs. Further reduction decreases the number of states to 10. We tested all 148 communicating pairs present in the traffic captures and calculated their respective state reduction gain. This is defined as the number of states of the reference

[1] See also: https://github.com/jjchromik/intravis/blob/master/example/over.pdf.
[2] See also: https://github.com/jjchromik/intravis/blob/master/example/all.pdf.

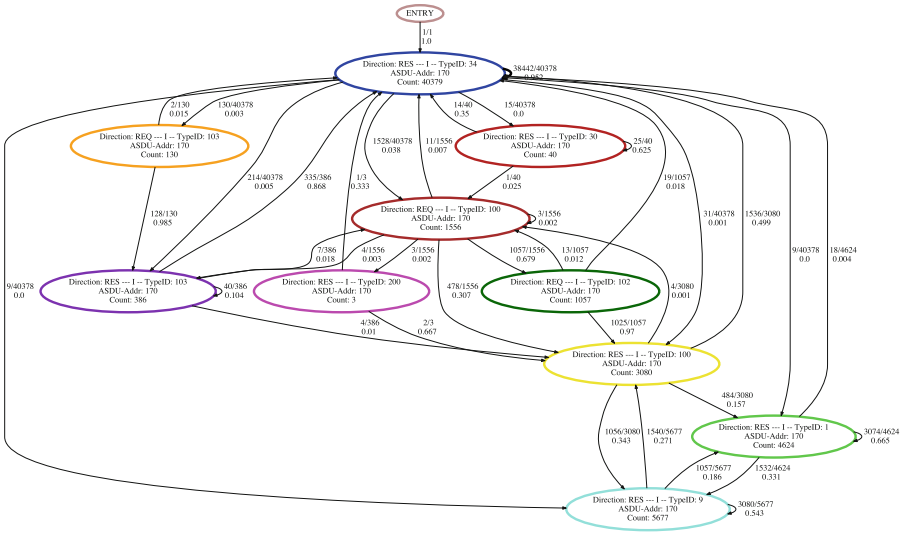


Fig. 8. Not taking into account IOAs

DTMC divided by the number of states of the reduced DTMC. The transitions reduction gain is defined analogously.

Table 1. Reduction gain for three different cases.

Communicating pair	Element	None	Overlapping	All	Overlapping - gain	All - gain
172.31.8.170 172.31.1.100 (example)	# states	117	17	11	6.88	10.64
	# transitions	896	51	39	17.57	22.97
172.31.10.230 172.31.1.100 (best case)	# states	189	19	10	9.95	18.9
	# transitions	1329	55	40	24.16	33.23
172.31.3.99 172.31.1.100 (worst case)	# states	11	11	9	1	1.22
	# transitions	22	22	22	1	1
average state gain					1.58	2.09
average transition gain					1.51	2.25

Table 1 provides the number of states and transitions for the original and the reduced traffic models for (i) the example shown in Figs. 7, 8 and 9, (ii) the DTMC with largest reduction gain (best case), and (iii) the DTMC with smallest reduction gain (worst case). We can see that while in the best case, the reduction *overlapping* decreases the number of states almost by a factor 10, and the reduction *all* almost by a factor 19. The worst case shows almost no reduction. We also provide the average state and transition gain that we observed for all 148 communicating pairs.

4 Validation

To compare the detection rates of the reduced traffic models with respect the original one, we introduce anomalies (e.g., out-of-order packets) into the traces.[3]

[3] The code used to modify the traces is available on github https://github.com/penc4ke/manipulateTraces.git.

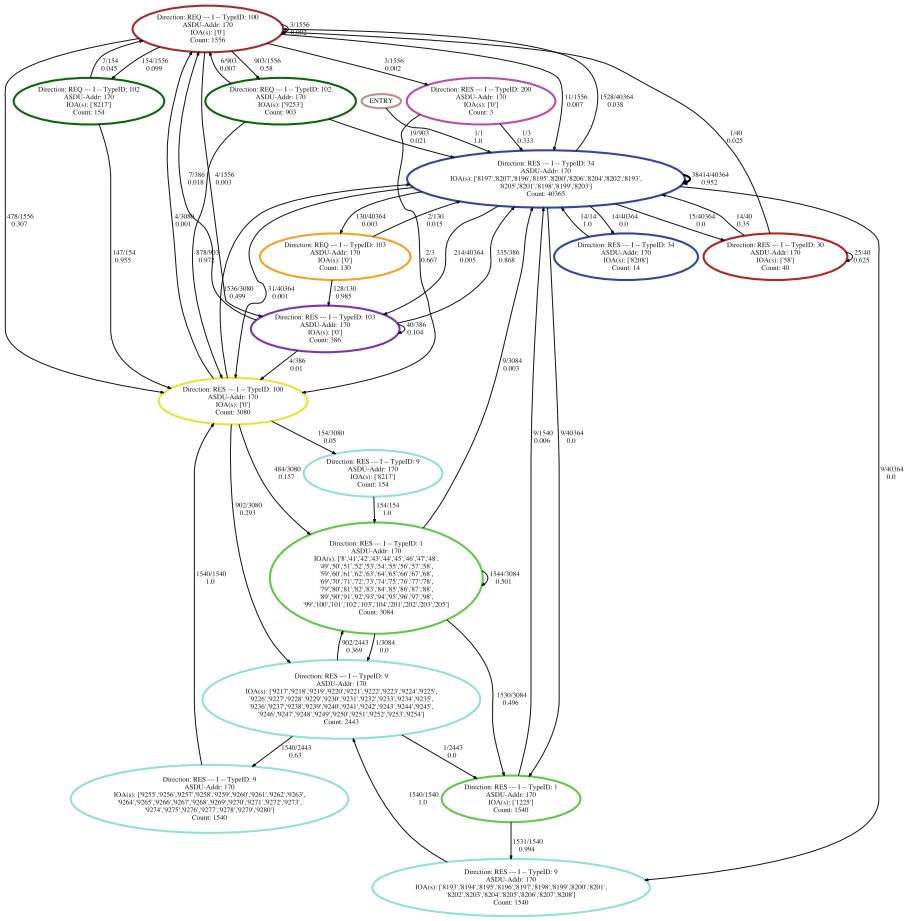


Fig. 9. DTMCs representing legit activities between devices 172.31.1.100 and 172.31.8.170 over the period of 10 days. States with overlapping IOAs are combined.

For validation we use the same traffic capture mentioned in Sect. 3.3. This data was split into two parts of 5 days each. One half is used for training a reference DTMC using either no reduction or one of the two approaches listed in Sect. 3.3. The other half is used for detecting anomalies by comparing the model obtained from the remaining (testing) trace with the training traffic model. Due to the regularity of SCADA traffic the amount of data should be enough to capture all relevant events. The following anomalies have been introduced: (i) copying a random packet from the used trace and adding it at a random position, (ii) removing a random packet from the trace, and (iii) swapping packets, i.e., interchanging the position of two random packets. We investigate applying the above changes to (i) 0.1%, (ii) 1%, and (iii) 10% of the packets from the testing trace. Note that while the added anomalies are not necessarily attacks, the results allow valuable insight into the accuracy of the reduced models.

4.1 Detection

We use the same algorithm and thresholds as explained in [12]. The thresholds for both a state violation and transition violation equal 0.1. To detect sequence violations we compare the differences between the trained DTMCs and testing DTMCs. In case the difference exceeds the above thresholds, an alert is raised. The detection mechanism checks the violations mentioned in Sect. 3.2:

New transitions violation - a transition exists in the testing DTMC, but not in the training phase. **New state violations** - a state created in the testing phase, which does not exist in the training phase. **Transition anomalies** - the transition probability in the testing DTMC differs more than the predefined threshold from the corresponding probability in the training phase.

Additionally, we provide the number of **state anomalies**, that is the number of states affected by transition anomalies.

4.2 Results and Discussion

We modify the second part of the trace 10 times and compute the median and variance of the number of detected anomalies for copying, removing and swapping respectively 0.1%, 1% and 10% of all 6079 packets. Furthermore, we compare the results of the original model without reduction to the results of the two proposed reductions.

Table 2. State anomalies: overall (new)

Reduction Type	No change		Copy			Remove			Swap		
			0.1%	1%	10%	0.1%	1%	10%	0.1%	1%	10%
none	11 (1)	median	11 (1)	11 (1)	19 (1)	11 (1)	11 (1)	14 (1)	11 (1)	11 (1)	20.5 (1)
		variance	0 (0)	0.233 (0)	2.456 (0)	0 (0)	0.1 (0)	0.667 (0)	0.4 (0)	0.178 (0)	0.933 (0)
overlapping	9 (1)	median	9 (1)	9 (1)	15 (1)	9 (1)	9 (1)	12 (1)	9 (1)	10 (1)	15.5 (1)
		variance	0 (0)	0.178 (0)	1.956 (0)	0 (0)	0 (0)	0.622 (0)	0.4 (0)	0.267 (0)	0.933 (0)
all	6 (0)	median	6 (0)	6 (0)	8 (0)	6 (0)	6 (0)	7 (0)	6 (0)	6 (0)	10 (0)
		variance	0 (0)	0.1 (0)	0.989 (0)	0 (0)	0 (0)	0.622 (0)	0.1 (0)	0.1 (0)	0.767 (0)

Table 2 shows the number of state anomalies, including the new states (given in brackets). The row providing the results for the original approach and the column showing the results without any changes in the trace are marked grey as they indicate the reference cases. For the non-modified trace, the original model detects 11 state anomalies, out of which 1 state was new. The new state corresponds, e.g., to a packet that occurs in the real traffic but was never seen in the training phase. The remaining 10 anomalous states are considered false positives, as they do not result from a modification of the original trace, but from an irregularity in the trace itself. Comparing with the two proposed reduction types, we notice that the number of false positives drops, hence, the detection accuracy of the reduced model improves, due to abstracting from the specific IOA numbers. In the reduction *all*, there are 6 anomalous states out of which

none are new. This suggests that the mentioned packet uses a TypeID that appears in the training capture, but the IOA number did not.

After introducing anomalies in the traffic sequence, we can see that only copying/removing/swapping 10% of the packets increased the number of detections. The introduced anomalies are not detected as new state anomalies, since they do not introduce any new command in the trace. For anomalies that do change the number of detections, the question remains, whether the detected state anomalies are harmless or whether they indeed can harm the system.

Table 3. Transition anomalies: overall (new)

Reduction Type	No change		Copy			Remove			Swap		
			0.1%	1%	10%	0.1%	1%	10%	0.1%	1%	10%
none	24 (10)	median	31.5 (17.5)	70 (55.5)	134 (115)	28.5 (14.5)	37 (23)	61 (46.5)	38 (24)	89.5 (75.5)	145 (127)
		variance	1.344 (1.344)	15.156 (15.211)	48.222 (36.989)	1.789 (1.789)	1.822 (1.822)	13.656 (11.333)	10.278 (10.278)	38.678 (35.333)	31.833 (32.233)
overlapping	18 (6)	median	25 (13)	57 (45.5)	98 (81.5)	22 (10)	29.5 (17.5)	49 (35)	31.5 (19.5)	71 (59.5)	104 (90.5)
		variance	2.278 (2.278)	15.956 (17.111)	34.844 (13.156)	1.433 (1.433)	0.678 (0.678)	8.278 (3.656)	8.544 (8.544)	16.233 (14.622)	12.711 (14.622)
all	14 (4)	median	17 (7)	30.5 (20.5)	49.5 (41)	15.5 (5.5)	21 (11)	33 (21)	21.5 (11.5)	37 (27)	57 (44.5)
		variance	0.267 (0.267)	8.711 (7.289)	13.733 (10.222)	0.278 (0.278)	1.333 (1.433)	2.933 (1.956)	7.122 (6.5)	15.211 (10.1)	12.622 (6.989)

Table 3 shows the number of transition anomalies and the number of new transition violations (provided in brackets). Again, the reference case is marked gray (reduction *none*). The original approach detects 24 transition irregularities in the original trace, out of which 10 are new. All those need to be considered as false positives. Possibly the training sequence was too short, as the dataset did not contain messages in this order. The gray column, representing the original trace shows that the reductions decrease the number of false positives w.r.t. the reference case.

Modifying the traces introduces additional transition anomalies. Even modifying 0.1% of all packets increases the number of new anomalous transitions considerably. When reducing the traffic models, the number of detected anomalies decreases. E.g., when copying 10% of the packets, without reduction 115 new transitions are observed, while *overlapping* results in 81.5 anomalies, and *all* in 41 anomalies. An operator may prefer fewer alerts, as too many notifications may be ignored. However, the question remains, how to distinguish an attack from a false positive alert. Note that the reduced number of detections when applying reductions stems from two sources. Fist, we lose false positives as in the reference case, which increases the accuracy of detection. Second, not every change is detected in the reduced model, which decreases the sensitivity. The current detection algorithm is not performed after each event, hence we are unable to distinguish between losing false positive or true positive.

Reconsidering the disconnector and switch attack from Sect. 2.3 shows that the reduction method should be chosen keeping the application in mind. If a single PLC would control the actuators, the same function (TypeId) referring to opening or closing respective IOs could appear in a specific order. Therefore, implementing reduction *all* could abstract away too much information. This could be preserved with the *overlapping* reduction, still reducing the size of the traffic model. In contrast, when dealing with simple reading commands, such as a General Interrogation, merging all IOAs would not result in a loss of accuracy, still reducing the size of the traffic model.

5 Conclusions

Commonly, SCADA traffic behaves quite regularly and results in packets sent in a predefined order. Hence, learning traffic models and comparing sequences of traffic to such models is a promising research direction. However, the developed models can easily become very large and it might not be feasible to maintain large models for each pair of communicating devices.

With this paper, we show that some cases exist where these models can be substantially reduced. In our use cases, states differing just for the range of Information Object Addresses, used in IEC-104, could be easily and conveniently combined in the DTMCs. We observe that completely abstracting from IOAs reduces the model size considerably while loosing accuracy. Despite lowering down the number of false positives this may cause the IDS to overlook specific attacks, like the disconnector and switch attack. For this reason a more conservative approach combining states with overlapping IOAs has the highest chance to succeed because of the higher model accuracy while still reducing model size. We conclude that when choosing reduction methods the actual purpose of the exchanged functions of the IEC-104 protocol should be taken into account. By understanding the goal of the actual functions (TypeIds), one can use specifically tailored reduction techniques for different functions. However, in most cases, the knowledge needed to the reduction function can come only from the operator side. Future work will focus on detecting actual attacks using a hybrid approach: either combining states with overlapping IOAs or abstracting from IOAs completely, depending on the TypeId. Moreover, the detection has to be performed in real-time, e.g., by using conformance testing techniques.

References

1. Zhu, B., Joseph, A., Sastry, S.: A taxonomy of cyber attacks on SCADA systems. In: International Conference on Internet of Things and on Cyber, Physical and Social Computing, pp. 380–388. IEEE CS Press, Washington, DC (2011)
2. Burke, G., Fahey, J.: AP Investigation: U.S. Power Grid Vulnerable to Foreign-hacks. http://lasvegassun.com/news/2015/dec/21/apinvestigation-us-power-grid-vulnerable-to-forei/. Accessed 06 June 2015

3. Goodin, D.: First known hacker-caused power outage signals troubling escalation. http://arstechnica.com/security/2016/01/first-known-hacker-caused-power-outage-signals-troubling-escalation/. Accessed 06 June 2015
4. Falliere, N., Murchu, L., Chien, E.: White Paper: W32. Stuxnet Dossier. Technical Report. Symantec Corporation (2011)
5. Caselli, M., Zambon, E., Petit, J., Kargl, F.: Modeling message sequences for intrusion detection in industrial control systems. In: Rice, M., Shenoi, S. (eds.) ICCIP 2015. IAICT, vol. 466, pp. 49–71. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26567-4_4
6. Barbosa, R.R.R., Sadre, R., Pras, A.: Flow whitelisting in SCADA networks. Int. J. Crit. Infrastruct. Prot. **6**(3), 150–158 (2013)
7. Goldenberg, N., Wool, A.: Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. Int. J. Crit. Infrastruct. Prot. **6**(2), 63–75 (2013)
8. Kang, B., McLaughlin, K., Sezer, S.: Towards a stateful analysis framework for smart grid network intrusion detection. In: 4th International Symposium for ICS & SCADA Cyber Security Research, pp. 1–8. BCS Learning & Development Ltd., Swindon (2016)
9. Lin, H., Slagell, A., Kalbarczyk, Z., Sauer, P., Iyer, R.: Runtime semantic security analysis to detect and mitigate control-related attacks in power grids. IEEE Trans. Smart Grid **99**, 1–16 (2016)
10. Barbosa, R.R.R., Sadre, R., Pras, A.: A first look into SCADA network traffic. In: IEEE/IFIP Network Operations and Management Symposium, pp. 518–521. IEEE CS Press, Maui, HI (2012)
11. Feng, C., Li, T., Chana, D.: Multi-level anomaly detection in industrial control systems via package signatures and LSTM networks. In: 47th IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 1–12. IEEE CS Press, Denver, CO (2017)
12. Caselli, M., Zambon, E., Kargl, F.: Sequence-aware intrusion detection in industrial control systems. In: 1st ACM Workshop on Cyber-Physical System Security, pp. 13–24. ACM (2015)
13. Fovino, I.N., Coletta, A., Carcano, A., Masera, M.: Critical state-based filtering system for securing SCADA network protocols. IEEE Trans. Industr. Electron. **59**(10), 3943–3950 (2012)
14. International Electrotechnical Commission: IEC 60870-5-104, Transmission Protocols, Network Access for IEC 60870-5-101Using Standard Transport Profiles (2003)
15. Alcaraz, C., Lopez, J., Zhou, J., Roman, R.: Secure SCADA framework for the protection of energy control systems. Concurrency Computation: Pract. Experience **23**(12), 1431–1442 (2011)
16. Clarke, G., Reynders, D.: Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems. Newnes, Oxford (2004)
17. Burke, G., Fahey, J.: LIAN 98(en): Protocol IEC 60870–5-104, Telegram Structure. http://www.mayor.de/lian98/doc.en/html/u_iec104_struct.htm. Accessed 13 December 2017
18. Nugteren, J.: ACM completes investigation into power outage in diemen. https://www.acm.nl/en/publications/publication/16469/ACM-completes-investigation-into-power-outage-in-Diemen/. Accessed 18 December 2017
19. Associated Press: Flights cancelled at schiphol airport as power outage hits amsterdam. https://www.theguardian.com/world/2015/mar/27/flights-cancelled-schiphol-airport-power-outage-amsterdam. Accessed 26 June 2017

Hidden Storage in Data Centers: Gaining Flexibility Through Cooling Systems

Robert Basmadjian[1(✉)], Yashar Ghiassi-Farrokhfal[2], and Arun Vishwanath[3]

[1] University of Passau, Innstrasse 43, Passau, Germany
robert.basmadjian@uni-passau.de
[2] Erasmus University Rotterdam, Burgemeester Oudlaan, Rotterdam, The Netherlands
y.ghiassi@rsm.nl
[3] IBM Research, Melbourne, Australia
arvishwa@au1.ibm.com

Abstract. Data centers are one of the biggest energy consumers in the ICT sector. Their cooling system accounts for almost half of the overall data center energy demand. Thus, the flexibility in shifting energy demand of the cooling systems in data centers could be a great asset for data center operators. The amount of flexibility and how it can be maximized are important yet open problems, due to the several stochastic processes involved. In this paper, we propose a novel methodology that allows data center operators to compute the flexibility of the cooling system by modeling it as an Energy Storage System (ESS). To enable such a mapping, the temperature set-points of the cooling systems must be expressed by a recursive formulation. To this end, based on thermodynamic concepts, in this paper we derive a recursive formulation for the temperature of the cooling systems and verify it empirically through a real-world data set. We then sketch (as our future work) how this mapping can be used to compute the flexibility of the cooling systems which can be efficiently leveraged during demand-response periods.

Keywords: Demand-response · Data center · Cooling system

1 Introduction

The enormous growth in the digitization of information has led to the rising need for the provision of extremely large data centers like the ones of Google, Microsoft, Amazon that consume power in the order of Mega Watts [9]. Consequently, an undesired side-effect of this extensive computing and communication activities is that data centers devour substantial amount of energy. To this end, it was shown that the energy consumption of data centers has a non-negligible share of the total energy consumption of the society: In 2007 the energy consumed by data centers in Western Europe was 56 TWh and is projected to increase to over 100 TWh per year by 2020 [8], whereas in 2014, data centers in

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 68–82, 2018.
https://doi.org/10.1007/978-3-319-74947-1_5

the U.S. consumed an estimated 70 TWh, representing about 1.8% of total U.S. electricity consumption [20]. Due to this significant energy demand on one hand and increased energy costs on the other, the operational expenditure (Opex) is becoming a dominant fraction of the total data center's cost of ownership (TCO) [10]. It is being recognized that the cooling infrastructure in a data center can account for up to 50% of the total energy demand with the remaining 50% attributable to the ICT resources [11].

Being bulk energy consumers in the energy grid, data centers can contribute to grid stability, should they present "flexibility" in shifting their energy consumption, when grid is under-supplied. Flexibility in this context is defined as the ability of a data center to provide reliably changes in power at a desired magnitude over a certain duration in response to changes in residual load. Furthermore, flexibility is indeed an integral characterization of the future energy grid, given large scale unpredictable renewable energy integration, which includes important concepts such as demand-side and supply-side management. For this purpose, demand-response (DR) [12] schemes have become the ultimate standard (e.g. OpenADR) that exploit the flexibility offered by consumers. In short, those schemes define a list of actions that need to be taken by the consumers to reduce drastically the electrical load during power shortage/outage periods. Flexibility of the data centers (in the sense of demand-response) needs to be quantified and optimally managed to benefit both stakeholders involved: data center operators (to minimize their energy cost) and energy grid (to stabilize the grid). In this context, it was shown that data centers are excellent candidates to participate in DR schemes due to their fully automated infrastructure providing inherent flexibilities (e.g. workload shifting, cooling set points alteration) on the one hand and significant power/energy demand on the other [13,14,19].

Among the several flexibility mechanisms that data centers provide, in this paper we study the one of offered by the cooling systems. We believe that energy optimization of data centers starts not only at the energy management level of internal operations (e.g. workload shifting/shedding) but also understanding the flexibility that cooling system provides, as the latter amounts to almost half of the overall energy demand. With ASHRAE's (American Society of Heating, Refrigerating and Air-Conditioning Engineers[1]) new recommendation about temperature set-points for data centers, it is now possible to operate a data center at higher temperature (until 35 °C) levels. To this end, recently Google increased the temperature set-points of its data centers which led to reduced energy consumption[2]. It was argued that data center operators can save up to 4% of energy costs by increasing the temperature set-point by 0.5 °C. Consequently, in this paper we exploit the temperature set-point flexibility that the cooling system of a data center provides and tend to model and quantify that flexibility, by mapping it to a *virtual Energy Storage System (ESS)* and further, using a novel methodology based on *network calculus*. Evaluating the flexibility

[1] https://www.ashrae.org/.

[2] http://www.datacenterknowledge.com/archives/2008/10/14/google-raise-your-data-center-temperature/.

of the cooling systems and deriving dynamic cooling strategies for data centers have not yet been studied (see Sect. 2) using known methodologies (e.g. network calculus). On the other hand, flexibility of energy storage systems (ESS) are well-studied and understood. In particular, optimal charging/discharging operation and ESS sizes have been theoretically obtained in [5,6], using the theory of network calculus. This theory provides tight bounds over the ESS overflow and ESS depletion probabilities, which can be linked, respectively, to the waste of power probability and loss of power probability. Thus, in this paper, we model the cooling system as a virtual ESS and pave the road to quantify the flexibility that the cooling systems in data centers can provide during DR periods. It is important to note that the proposed approach does not enforce turning on-off the cooling system, instead control the hot air injected to the system to maintain the system within temperature boundaries. Hence, the lifespan of the cooling system is not degraded any further beyond its regular cycle of operation.

The rest of this paper is organized as follows. We define the problem, conceptually explain the underline mapping between the cooling systems and energy storage systems (ESS), and sketch our proposed approach to solve the problem in Sect. 3. We provide a recursive temperature formulation and demonstrate its accuracy in Sect. 4. Using that model, the counterpart elements in mapping the cooling system to a virtual ESS are quantified in Sects. 5 and 6 concludes the paper.

2 Related Work

Demand-response (DR) deals with the interaction between a utility (or a different stakeholder) on the energy supply side and several consumers on the demand-side, especially regarding the question of how to incentivize them for their provided flexibilities. Consequently, in [17] demand-response was defined by the energy consumers' changes of demand-side patterns/behaviors as a reaction to dynamic prices or other incentives.

The terminology demand-side management started back in the late 1980s by Gellings and Chamberlin [16]. It is the basis for the demand-response concept where the main idea consists of making flexible the power demand as part of an integrated resource planning comprising of both supply and demand. Hence, demand-side management is aimed at the power management of facilities (e.g. in our case data centers) influencing the load shape by changing their power usage pattern (e.g. time of performing the required activity) and/or magnitude. However, in order to achieve this, the facilities need to be by themselves able to provide flexible mechanisms. Data centers are an excellent candidate to participate in demand-side management due to (1) fully automated infrastructure and (2) numerous flexibility providing mechanisms such as cooling system, workload shifting/shedding, Uninterrupted Power Supply (e.g. battery storage), and flexible contracts.

There have been several works in the literature studying the cooling impact and effect of altering temperature set points in data centers. In [15,19] the authors carried out field experiments to improve the understanding of the DR opportunities in DCs. The study evaluated an initial set of control and load migration strategies and economic feasibility. The findings show that with minimal or no impact to DC operations, a demand savings of 25% at the DC level or 10–12% at the whole building level can be achieved with strategies for cooling and IT equipment, and load migration. In [21], the authors carried out a multi-faceted study of temperature management in data centers, where a large collection of field data from different production environment was used in order to analyze the impact of temperature on hardware reliability. Based on these studies, the authors make recommendations for temperature management in data centers, that create the potential for saving energy, while limiting negative effects on system reliability and performance. In [22], it was shown the high potential of adaptive cooling in data centers using traditional computer room air conditioning (CRAC) units, where it was shown that the cooling-related energy savings enabled to avoid a multi-million dollar power build out to the data center. In [23], the author highlighted the inadequacy of traditional computational fluid dynamics (CFD) analysis of the cooling design for data centers, where the equipment population and layout changes over time, and where the heat signature changes constantly in response to workload. Furthermore, the author proposed as a viable solution adaptive cooling system by taking into account the ambient temperature, IT rack heat load, etc. In [24], opportunities for improving thermal management and energy performance of data centers with automatic control were studied. It was shown that by implementing simple and modular control strategies could significantly improve the energy performance of the data center while maintaining proper thermal management conditions. Unlike the above mentioned contributions, in this paper the main contribution is to model the cooling system using network calculus in order to analyze mathematically the flexibility extends provided by such type of a mechanism through dynamic cooling. The derived mathematical model will serve to find out the optimal cooling strategies for data centers. To the best of our knowledge, this is the first effort in the literature to achieve this goal. Similar to some of the approaches in the literature, our model takes into account important parameters such as the ambient temperature, the IT workload, power consumption of the data center as well as of the cooling system.

3 Problem Definition and Methodology

In this section, we sketch our trajectory to evaluate the maximum flexibility achieved in a data center by optimally adjusting the operation of the cooling system. In short, we map this problem into a well-studied one: energy storage evaluation. Then, borrowing the techniques from storage analysis, we enable the flexibility evaluation of the cooling system along with its optimal operation. To see the similarity of a cooling system and an energy storage system (ESS), we briefly discuss ESS technologies, operations, and constraints in the following.

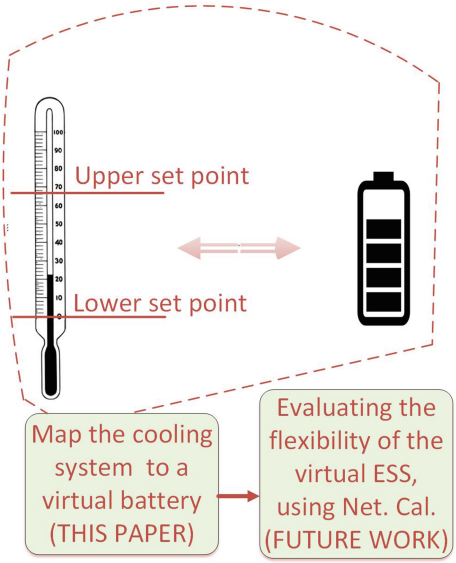


Fig. 1. The road map to evaluate the flexibility of a cooling system.

3.1 Taxonomy of Energy Storage Devices

Energy Storage Systems (ESS) have shown their potential to revolutionize the electricity grid by smoothing the variable energy generation of intermittent renewable energy sources (RES) such as photovoltaic and wind mills. In this respect, the impact of ESS to the grid has been extensively studied by considering the optimal operating strategies of when to charge and discharge the ESS. Note that an ESS can be defined as a system composed of battery modules and a commercial example can be found at [25].

There are several energy storage systems (ESS) currently in use, each of which has its own characteristics which might fit a certain application. From the technology point of view and how electricity is stored in the device, ESSs can be categorized into mechanical, thermodynamic, electrochemical, and electromagnetic [2,5].

There are certain inherent physical constraints attributed to each of these storage technologies, which could be limiting in certain applications [5]. For example, the round-trip efficiency of storing and withdrawing electricity in each of these devices is different. Some of these technologies convert electricity to other forms of energy to store it and convert it back to electricity when needed. For example, flywheels convert electricity into mechanical inertia and batteries convert electricity into chemical energy. Energy conversion is not ideal and leads to inefficiency. Some other storage technologies such as Supercapacitors have no energy conversion and hence, have high round-trip efficiency. Another important physical constraint is the leakage rate of the stored energy. In some storage

devices the stored energy such as Supercapacitors the stored energy vanishes by time due to self-discharge, whereas other devices such as batteries can keep energy for a long time without losing it considerably. Finally, the maximum charging/discharging rate is a limiting factor for some devices such as batteries, but is not for some others such as Supercapacitors or flywheels.

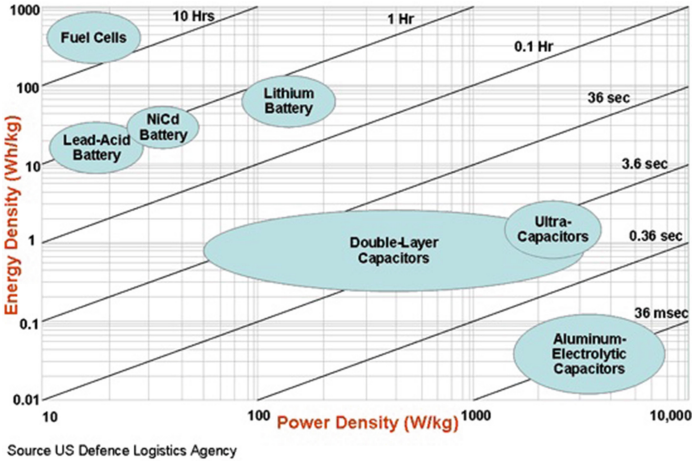


Fig. 2. Energy and power densities of storage technologies.

Many of these physical constraints are represented well by two important metrics: energy density and power density. Energy (resp. power) density is the maximum amount of energy (resp. power) that can be stored in (resp. drawn in or out of) a given storage device per unit volume. Typically, energy density and power density are contradictory and there is no technology with reasonable price that is both energy and power efficient (see Fig. 2). Each application requires a certain level of energy and power efficiency and hence, there is one or a combination of multiple storage technologies that fits each application [7].

Suppose that we have an ESS which has high power efficiency, low energy efficiency, and almost ideal round-trip efficiency. Supercapacitors are examples of this sort. In a discrete-time setting with the time unit Δ, the SCap's state of charge $b(k)$ at any time k can be, recursively, presented by [1]:

$$b(k) = \kappa b(k-1) + P_{in}(k)\Delta - P_{out}(k)\Delta \tag{1}$$

where κ is the self-discharge of the ESS, $P_{in}(k)$ and $P_{out}(k)$ are, respectively, the power input and the power output of the ESS at time k. Note that Δ must be small enough to be a good representative of the continuous time events, but also constrained to the time resolution of the dataset at hand. The state of charge of the ESS must be kept between the minimum and maximum allowable range, which is

$$0 \leq b(k) <\leq B\ . \tag{2}$$

where B is the size of the ESS. However, for some other ESSs (such as batteries) the lower bounds and upper bounds could be a fraction of the capacity of the ESS to enhance their lifetime (instead of 0 and B).

Two performance metrics are essential to keep track of for energy systems with ESS: loss of load probability (LOLP) and waste of energy probability (WOEP). The loss of load probability (LOLP) in storage devices is defined as the likelihood that at a certain time instant the energy system needs to withdraw energy from the ESS but ESS does not have enough stored energy. Mathematically speaking,

$$LOLP = \Pr\{b(k) < 0\} \tag{3}$$

Similarly, waste of energy probability (WOEP) is defined as the likelihood that at any time instant the energy that needs to be stored in the ESS cannot be stored, because the ESS is full; i.e.,

$$WOEP = \Pr\{b(k) > B\} \tag{4}$$

Recently, LOLP and WOEP have been formulated, using a mapping to data buffered queues and borrowing the techniques from Network Calculus, originally developed for data networks [1,5,6], we show how the flexibility in data center cooling system can be mapped to the operation of a SCap and hence, allowing us to model and analyze them.

3.2 Heating/Cooling System in Data Centers

The cooling system at data centers can be used efficiently to achieve flexibility in terms of demand response. The temperature of the data center increases as soon as cooling systems are off duty and that extensively decreases the energy demand of the data center during those times, given that the cooling system by itself accounts for 40% of the entire data center energy demand.

To prevent premature failures of the servers and enhance their lifetime in a data center, the operating temperature of the data center must be kept between a minimum and maximum temperature T_l and T_u, respectively. Operating servers heat up the data center and increase the temperature beyond T_u, if not controlled by the cooling system. The cooling system in the data center automatically sets its duty-cycles to keep the operating temperature point of the data center $T(k)$ at any time k in the allowable range $[T_l, T_u]$; i.e.;

$$Pr\{T(k) < T_l\} \leq \epsilon_l; \qquad Pr\{T(k) > T_u\} \leq \epsilon_u \tag{5}$$

For a target lower temperature violation probability (LTVP) and an upper temperature violation probability (UTVP), Eq. (5) can be re written as

$$LTVP = Pr\{T'(k) < 0\}; \qquad UTVP = Pr\{T'(k) > T_{max}\} \tag{6}$$

where

$$T'(k) = T(k) - T_l; \quad T_{max} = T_u - T_l \tag{7}$$

Comparing Eq. (6) with Eqs. (3) and (4), we find that the performance metrics in the cooling systems are similar to those of an ESS. However, to complete the mapping elements we should also show that $T'(k)$ can be presented in a recursive format, similar to that of an ESS as stated in Eq. (1).

Conceptually, the DR obtained through the cooling system resembles the one obtained through an ESS in the sense that the whole cooling system can be operated as an *energy renting entity*: The energy use can be reduced by turning off the cooling system and this subsequently increases the temperature beyond the set point up to the maximum of T_u. This amount of discarded energy demand from the cooling system must be added to the energy demand of the cooling system at a later time to turn the set point back to its regular value. Thus, one can think about it as a system from which we can "borrow" energy at a point but should "return" it at a later time. This is exactly what an ESS does: Energy can be borrowed from an ESS but it must be returned at a later time to attain its original state of charge (SoC).

According to the above reasoning, we propose to map the cooling system operation as a virtual ESS and then employ *network calculus* results to study the potential of the flexibility and the optimal operation of the cooling system (see Fig. 1). In this paper, we focus on the *mapping part* (the first block of modeling the flexibility of data centers) and will proceed with applying the network calculus approach to the resulting virtual ESS in *our future work* to enable flexibility evaluation of the cooling systems.

The LOLP and WOEP constraints mentioned in Eqs. (3) and (4) can be used in our mapping to quantify the flexibility subject to keeping the temperature within the allowable range with a small given violation probability (see Eq. 6). Since finding the optimal temperature set-points is a generic concept, thus the proposed methodology is independent of the cooling system type.

4 Modeling Cooling System

4.1 Setup Environment

In this section, we describe the setup environment of the carried out experimental analysis on cooling systems. The field tests were conducted at a data center (peak power demand of 200 kW) in Passau, Germany. During these test runs, the operating temperature set-points were set to be always in the range of [10 °C, 35 °C] in order to avoid any hardware failure and conform to the standards of ASHRAE. The cooling system consists of outdoor and indoor units. The 2 indoor units (see Fig. 3) are installed directly at the data center and work in parallel, which are connected to two outdoor units. The outdoor units consist of two cold water generators on the housetop. The temperature set-point of both indoor units is set to approximately 22.5 °C. The temperature set-point can be

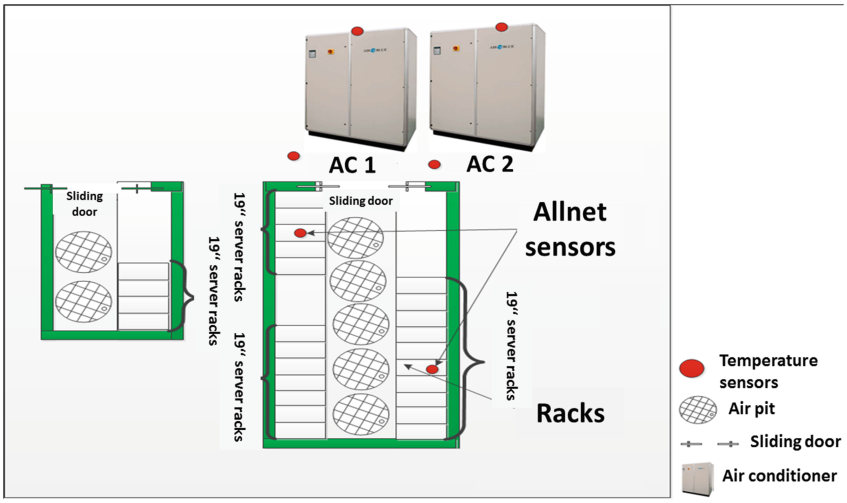


Fig. 3. Cooling system configuration.

adjusted between 12.5 °C and 25.0 °C. The corresponding data center is composed of the following servers:

- 6 HP ProLiant DL460
- 5 HP ProLiant DL360
- 2 HP ProLiant DL585
- 1 HP ProLiant DL380
- 1 HP ProLiant DL320

For the purpose of measuring temperature set-point values while testing the cooling system, we used an ALLNET IP Sensoric Appliance (ALL4500) and ALLNET temperature sensors (ALL3006).

4.2 Testing Conditions and Methodology

The test was carried out in April 24th from 8:53 AM to 3:15 PM where the weather condition was densely clouded with an ambient temperature of 19 °C. The following data were read for the corresponding experiment:

- Cold aisle temperature: Every 10 s the temperature value was recorded.
- Power consumption of cooling system: Every 5 min the energy (kWh) spent by the system was recorded. These values were converted into kW.
- Power consumption of rack servers: Every 10 s the current Watt value was recorded for each server. To get the values for a 5 min interval the mean value for this time span was calculated and converted into kW.
- Power consumption of blades: Every 5 min the current Watt value of the five blade servers in total was recorded and converted into kW.

During the carried out experiment, we investigated cooling-down, heating-up and then back to normal operation temperature set-point process and adopted the following methodology:

- Step 1: Power consumption of the two air conditioners is metered separately from the other devices in the data center. Consequently, the consumption required to keep the temperature at 21.0 °C level during a fixed period of time (Phase 1) can be determined. The power consumption of the servers for the same time span is identified as well.
- Step 2: At the end of Phase 1 the air conditioners are adjusted to maximum cooling. The time (Phase 2) it takes to reach the minimum temperature possible (12.5 °C) is measured as well as the corresponding power consumption of air conditioners and servers during this time span.
- Step 3: When the sensors indicate reaching the minimum temperature limit, the indoor units of the cooling system are turned off (end of Phase 2). Now the time (Phase 3) it takes to reach the maximum permissible temperature (30.0 °C) is measured as well as the corresponding power consumption of servers during this time span.
- Step 4: When the sensors indicate reaching the maximum temperature limit, the indoor units are turned on again and the set-point temperature of both units is set to 22.5 °C again (end of Phase 3). Now the time (Phase 4) it takes to reach the default temperature (21.0 °C) in the cold aisle is measured as well as the corresponding power consumption of air conditioners and servers during this time span.

4.3 Modeling

We model the temperature evolution inside the data center using a 'gray box' methodology, similar to the one described in [26]. This model is inspired the first law of thermodynamics, which relates temperature changes at any time to the properties of the environment, the ambient temperature, and the input/output heat power. In short, the changes in the temperature at any time is linearly proportional to the temperature difference between the temperature inside that environment and the ambient temperature, the input power and the output power. Mathematically, in a continuous-time system, the model at any time t can be represented as follows[3]:

$$C_z \frac{dT(t)}{dt} = \alpha \left(T_a(t) - T(t)\right) + \beta P_s(t) - \gamma P_c(t), \tag{8}$$

where C_z is the thermal capacitance of the zone housing the data center and T_a is the ambient temperature. Thermal capacity is the resistance of that environment to temperature changes and is a characteristic of that environment. Moreover, there are efficiency factors (α, β, γ) attributed to how ambient temperature difference, the input power, and the output power impact the temperature

[3] Please note that we use t as the time notation in a continuous-time model and k in a discrete-time model.

variations. The coefficient α is the effective heat transfer coefficient between the ambient and the zone, β is the coefficient that denotes the internal heat gain due to the servers and γ is the coefficient that maps the rate of heat efflux attributed to the cooling provided by the air-conditioning system.

By disctretizing the continuous time with unit steps Δ, the continuous-time differential equation in Eq. (8) can be approximated by the following discrete-time difference equation:

$$C_z \frac{T(k) - T(k-1)}{\Delta} = \alpha \left(T_a(k) - T(k)\right) + \beta P_s(k) - \gamma P_c(k) \tag{9}$$

Our gray box model is based on Eq. (9) and optimally finding the values of α, β, and γ which make the model the best fit for the given dataset. We also move the estimation of the heat capacity C_z from the theoretical modelling to data-fitting, by removing C_z from Eq. (9) and assuming that the modified versions of α, β, and γ, when fitted to the data according to

$$\frac{T(k) - T(k-1)}{\Delta} = \alpha \left(T_a(k) - T(k)\right) + \beta P_s(k) - \gamma P_c(k) \tag{10}$$

will account for C_z numerically. Overall, our gray box model takes as inputs a dataset that includes server room temperature, the input and the output power and finds the optimum values of the three model coefficients α, β and γ according to the MMSE approach; i.e., the coefficients minimize the following:

$$e = \sqrt{\sum_{k=1}^{N} \left(T(k) - T_{meas}(k)\right)^2} \tag{11}$$

where, e is the root mean square error for a look-ahead prediction time window of N time samples, $T(k)$ is the predicted zone temperature at discrete time instant k given by the solution of Eq. (10) for any choice of α, β and γ. $T_{meas}(k)$ is the measured zone temperature that is obtained from the experiments described in the previous subsection. We have assumed, without loss of generality, that the thermal capacitance of the zone housing the data center, i.e. C_z is unity as this suffices our modeling objective.

In Fig. 4, we compare the internal temperature predicted by the gray model with the actual temperature recorded following the experiment described earlier. We can observe from the figure that the gray box model is able to predict the temperature evolution across the different phases reasonably well. The root mean square error (RMSE) is 1.33 °C. Note that in order to assess the results obtained from RMSE methodology, another test was conducted in a different day with an ambient temperature of 3 °C (see Fig. 5). The obtained results confirm that the mean square error of 1.33 °C can be further reduced by considering moving window based regression however we keep this for our future work.

5 Mapping a Cooling Systems to an ESS

In this section, we identify the mapping elements of the virtual ESS representing the cooling system. To do so, we rephrase our temperature model from Eq. (10)

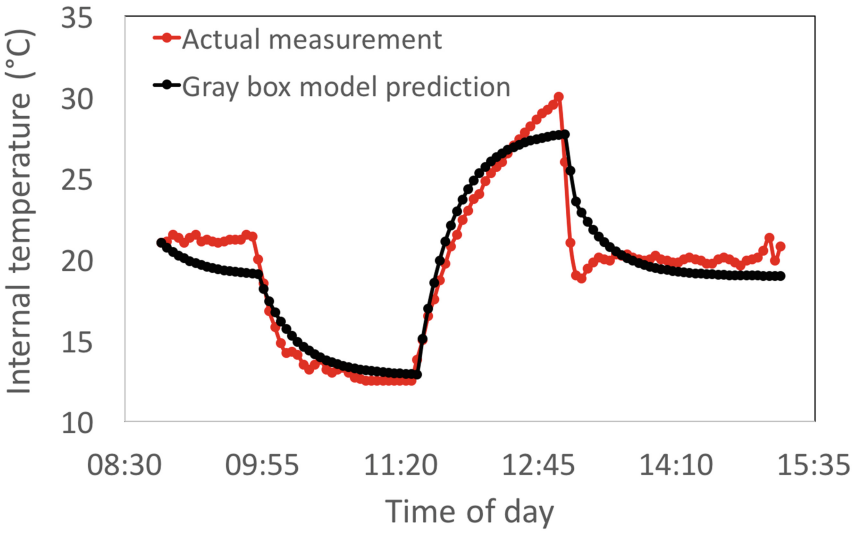


Fig. 4. Comparison between measured and modeled internal temperature in the data center.

as the following:

$$T(k) = \frac{1}{1+\alpha\Delta}T(k-1) + \frac{\alpha T_a(k) + \beta P_s(k)}{1+\alpha\Delta}\Delta - \frac{\gamma P_c(k)}{1+\alpha\Delta}\Delta \tag{12}$$

or equivalently

$$T'(k) = \frac{1}{1+\alpha\Delta}T'(k-1) + \frac{\alpha(T_a(k) + T_l) + \beta P_s(k)}{1+\alpha\Delta}\Delta - \frac{\gamma P_c(k)}{1+\alpha\Delta}\Delta \tag{13}$$

which resembles the recursive statement of the state of charge of an ESS in Eq. (1). Moreover, the temperature at any point must be kept between its lower and upper bound; i.e.,

$$0 \leq T'(k) \leq T_{max} \tag{14}$$

Comparing Eqs. (1) and (2) with Eqs. (13) and (14), we find out that the operation of the cooling system can be mapped to the operation of an ESS if we treat the elements of an ESS mentioned in Table 1 as counterparts of a cooling system.

This result analytically validates our observation in Sect. 3 that a cooling system could be interpreted as a virtual ESS. This allows us to adopt the ESS analytical results, in particular the theory of network calculus, to model and evaluate the flexibility of the data center in our future work. It is worthwhile to note that unlike ESS, the self-discharge property of the cooling system is related to the intrinsic physical properties of the housing of the data center.

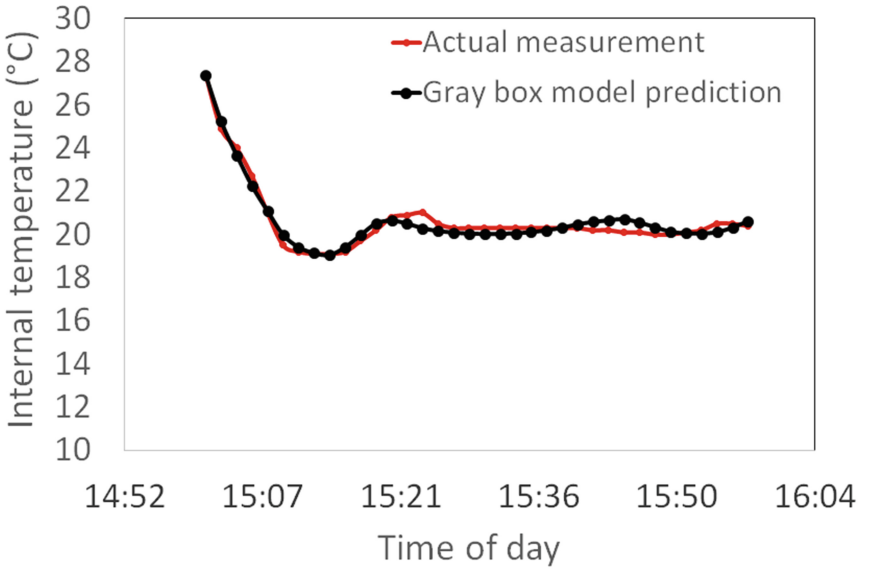


Fig. 5. Comparison between measured and modeled internal temperature in the data center.

Table 1. Mapping elements.

Cooling system	ESS
$T'(k)$	$b(k)$
$T_u - T_l$	B
$\frac{1}{1+\alpha\Delta}$	κ
$\frac{\alpha(T_a(k)+T_l)+\beta P_s(k)}{1+\alpha\Delta}$	$P_{in}(k)$
$\frac{\gamma P_c(k)}{1+\alpha\Delta}$	$P_{out}(k)$

6 Conclusion

To facilitate the integration of large scale renewable energy sources in the energy mix, flexibility has drawn much attention lately. As one of the important forms of flexibility, "demand-side management" has been recently introduced and demand-response (DR) schemes became the ultimate solution for "demand-side management" where those schemes exploit the flexibility provided by the consumers. To this end, data centers were investigated and have shown their excellent potential in participating in DR schemes due to (1) their fully automated infrastructure with negligible human intervention, (2) numerous flexibility mechanisms that they provide and (3) their significant power demand (order of Mega Watts).

In this paper, we modeled the temperature of a data center and sketched the path how this model can be used to analytically quantify the amount of

flexibility that the cooling system mechanism of a data center can provide during DR events (e.g. peak power shaving). To achieve this, we first carried out an experimental analysis by studying the effect of (1) cooling-down, (2) heating-up and (3) normal operation temperature set-point process. We model this behavior using gray-box methodology, where we showed that the root mean square error is about 1.33 °C. Based on the provided temperature model, we proposed a novel methodology of calculating cooling system flexibility using the analogy of energy storage systems (ESS) and adopting the network calculus theory. In our future work, we will use the established model to quantify the flexibility of the cooling system in a data center, subject to the temperature constraints of the data center and using network calculus.

References

1. Raeis, M., Burchard, A., Liebeherr, J.: Analysis of the leakage queue: a queuing model for energy storage systems with self-discharge. http://arxiv.org/abs/1710.09506
2. Wang, D., Ren, C., Sivasubramaniam, A., Urgaonkar, B., Fathy, H.: Energy storage in datacenters: what, where, and how much? In: Proceedings of the ACM Sigmetrics/Performance, pp. 187–198 (2012)
3. Freeh, V.W., Lowenthal, D.K.: Using multiple energy gears in MPI programs on a power-scalable cluster. In: Proceedings of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 164–173. ACM (2005)
4. Greenberg, A., Hamilton, J., Maltz, D.A., Patel, P.: The cost of a cloud: research problems in data center networks. ACM SIGCOMM Comput. Commun. Rev. **39**, 68–73 (2008)
5. Ghiassi-Farrokhfal, Y., Keshav, S., Rosenberg, C.: Toward a realistic performance analysis of storage systems in smart grids. IEEE Trans. Smart Grid **6**, 402–410 (2015)
6. Ghiassi-Farrokhfal, Y., Keshav, S., Rosenberg, C., Ciucu, F.: Solar power shaping: an analytical approach. IEEE Trans. Sustain. Energ. **6**, 162–170 (2015)
7. Ghiassi-Farrokhfal, Y., Keshav, S., Rosenberg, C., Adjaho, M.-B: Joint optimal design and operation of hybrid energy storage systems. IEEE J. Sel. Area Commun. **34**, pp. 639–650 (2016)
8. Code of Conduct on Data Centres Energy Efficiency, Version 2.0: European Commission. Institute for Energy, Renewable Energies Unit (2009)
9. Katz, R.H.: Tech titans building boom. IEEE Spec. **46**, 40–56 (2009)
10. Barroso, L.A., Clidaras, J., Hölzle, U.: The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. chapter 6, p. 96 (2013)
11. Meijer, G.I.: Cooling energy-hungry data centers. Science **328**, 318–319 (2010)
12. Assessment of Demand Response and Advanced Metering. Federal Energy Regulatory Commission (2013)
13. Basmadjian, R., Botero, J.F., Giuliani, G., Hesselbach, X., Klingert, S., de Meer, H.: Making data centres fit for demand response: introducing GreenSDA and GreenSLA contracts. IEEE Trans. Smart Grid (2016)
14. Piette, M.A., Girish, Gh., Sila, K., Koch, E., Hennage, D., Palensky, P., McParland, C.: Open Automated Demand Response Communications Specification (Version 1.0). LBNL (2009)

15. Ghatikar, G., Ganti, V., Matson, N., Piette, M.A.: Demand response opportunities and enabling technologies for data centers: Findings from field studies. Lawrence Berkeley National Laboratory, Technical report (2012)
16. Gellings, C., Chamberlin, J.H.: Demand-Side Management: Concepts and Methods. PennWell Books, Houston (1987)
17. Assessment of demand response and advanced metering. Federal Energy Regulatory Commission, Technical report (2010)
18. Aghaei, J., Alizadeh, M.-I.: Demand response in smart electricity grids equipped with renewable energy sources: a review. Renew. Sustain. Energ. Rev. **18**, 64–72 (2013)
19. Girish, G., Ganti, V., Matson, N., Piette, M.A.: Demand response opportunities and enabling technologies for data centers: findings from field studies. Lawrence Berkeley National Laboratory (2012)
20. Shehabi, A., Smith, S., Sartor, D.A., Brown, R.E., Herrlin, M., Koomey, J.G., Masanet, E.R., Horner, N., Azevedo, I., Lintner, W.: United States data center energy usage report. Lawrence Berkeley National Laboratory (2016)
21. El-Sayed, N., Stefanovici, I.A., Amvrosiadis, G., Hwang, A., Schroeder, B.: Temperature management in data centers: why some (might) like it hot. In: Proceedings of 12th International Conference on Measurement and Modeling of Computer Systems (2012)
22. Dynamic Cooling Control Provides Substantial Data Center Energy Savings. In: Air Conditioning, Heating & Refrigeration News (2013)
23. Kleyman, B.: Understanding the Benefits of Dynamic Cooling Optimization. White paper (2015)
24. Boucher, T.D., Auslander, D.M., Bash, C.E., Federspiel, C.C., Patel, C.D.: Viability of dynamic cooling control in a data center environment. J. Electron. Packaging **128**, 137–144 (2006)
25. Fenecon Energy Engineering. https://fenecon.de/en_US/page/stromspeicher
26. Vishwanath, A., Chandan, V., Mendoza, C., Blake, C.: A data driven pre-cooling framework for energy cost optimization in commercial buildings. In: Proceedings of e-Energy. ACM (2017)

Performance Benchmarking of Network Function Chain Placement Algorithms

Alexej Grigorjew(✉), Stanislav Lange, Thomas Zinner, and Phuoc Tran-Gia

University of Würzburg, Würzburg, Germany
{alexej.grigorjew,stanislav.lange,zinner,
trangia}@informatik.uni-wuerzburg.de

Abstract. The Network Function Virtualization (NFV) paradigm enables new flexibility and possibilities in the deployment and operation of network services. Finding the *best* arrangement of such service chains poses new optimization problems, comprising a combination of placement and routing decisions. While there are many algorithms on this topic proposed in literature, this work is focused on their evaluation and on the choice of reference for meaningful assessments. Our contribution comprises two problem generation strategies with predefined optima for benchmarking purposes, supplemented by an integer program to obtain optimal solutions in arbitrary graphs, as well as a general overview of concepts and methodology for solving and evaluating problems. In addition, a short evaluation demonstrates their applicability and shows possible directions for future work in this area.

Keywords: NFV · VNF Chain Placement · Optimization
Performance evaluation · Problem generation

1 Introduction

In modern networks, operators apply various network functions to their traffic flows, either due to their specific requirements or due to the network's policy in general. Packets are monitored, modified, or even dropped to perform these functions, such as firewalls, deep packet inspection, load balancers, and core gateways in LTE networks. Traditionally, they are implemented by special hardware middleboxes with high performance guarantees, but they also suffer from high costs, low flexibility, low scalability, and vendor dependence. These problems are addressed by the Network Function Virtualization (NFV) paradigm [1]. Hardware middleboxes are replaced by software instances, running in virtualized environments on cheap, vendor independent commercial-off-the-shelf (COTS) machines.

With the newly attained flexibility, new optimization problems arise in the context. In particular, the Virtual Network Function Chain Placement (VNFCP) problem deals with the orchestration of Virtual Network Functions (VNFs) in the network: (a) how many VNF instances are needed, (b) where are they located,

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 83–98, 2018.
https://doi.org/10.1007/978-3-319-74947-1_6

and (c) which traffic flow is using which instance. Thereby, various constraints are considered, such as bandwidth and resource limitations. The optimization objective usually involves the maximization of profit or the minimization of costs, for example by reducing the number of instances.

As the VNFCP problem is NP-hard [2], most publications rely on heuristics and approximations for productive use, featuring varying objectives, constraints, and levels of detail. However, assessing the quality of attained solutions is no trivial task in itself. For example, comparing the performance of two heuristics without knowing an optimal reference placement only yields limited expressiveness. In order to overcome these problems, this work investigates different approaches towards the performance evaluation of VNFCP algorithms. In particular, it discusses the generation of parameterized artificial problem instances with known optimal solutions, providing an independent reference for comparison.

The remainder of this work is structured as follows. Section 2 formally introduces a common variant of the VNFCP problem. In Sect. 3, different approaches for its solution are reflected with special regard to their respective evaluation techniques. In addition, Sect. 3.1 includes an Integer Linear Program (ILP) that obtains optimal solutions for small problem instances. Section 4 proposes artificial problem generation by means of two concrete strategies, whose results are demonstrated in Sect. 5. Related work on the evaluation of network optimization heuristics is addressed in Sect. 6. Finally, in Sect. 7, we discuss remaining challenges and conclude the paper.

An exemplary implementation of the presented ideas, such as the ILP and the problem generation strategies, is also available on GitHub[1].

2 Virtual Network Function Chain Placement

This section presents a brief overview of the VNFCP problem. Note that, as different approaches consider different details of the problem, this work only provides a generic definition of the most commonly used models. For further details, please refer to the corresponding publications in Sect. 6.1, and in particular to [3] for a more detailed version of this specific model.

2.1 Input and Output Models

The input model usually comprises three components. The network is represented by an undirected graph $G = (V, E)$. Nodes $v \in V$ may have computational resources $v_{\text{cpu}} \in \mathbb{R}$, while links $e \in E$ have bandwidth and delay properties $e_{\text{bw}}, e_{\text{d}} \in \mathbb{R}$. The available network function types $t \in T$ require a certain amount of resources $t_{\text{cpu}} \in \mathbb{R}$ and possess similar attributes as links: $t_{\text{bw}}, t_{\text{d}} \in \mathbb{R}$. Finally, each traffic request $r \in R$ consists of source and destination nodes $r_{\text{src}}, r_{\text{dst}} \in V$, bandwidth and maximum latency requirements $r_{\text{bw}}, r_{\text{d}} \in \mathbb{R}$, and a sequence of network functions $r_{\text{c}} \in T^{|r_{\text{c}}|}$ which represent the requested function chain.

[1] https://github.com/lsinfo3/vnfcp-benchmarking.

The output model contains the set of placed VNF instances $z \in \mathcal{I}$ which possess a type $z_{\text{type}} \in T$ and location $z_{\text{node}} \in V$. In addition, for every traffic demand $r \in R$, the assigned route $r_{\text{path}} \in V^{|r_{\text{path}}|}$ and the used instances $r_{\text{inst}} \in (\mathcal{I} \cup \{\emptyset\})^{|r_{\text{path}}|}$ are given. Hereby, r_{inst} has the same length as r_{path}, and $\emptyset$ indicates that no VNF is applied at the respective location of the route.

2.2 Constraints

The considered constraints can be split into two categories. On the one hand, consistency between different variables must be ensured. For example, the location of the i-th instance in r_{inst} must equal the i-th node in r_{path}, or more fundamentally, for each subsequence (v_i, v_{i+1}) in r_{path}, there must be an edge in the network graph, i.e., $\{v_i, v_{i+1}\} \in E$. On the other hand, resource and delay requirements must be met. This includes CPU capacities v_{cpu} on nodes, bandwidths e_{bw} on links, capacities t_{bw} of the instances $z \in \mathcal{I}$, and maximum flow latencies as defined by r_{d}. Further details are omitted in this work, as they differ between different approaches.

2.3 Objectives

The range of considered objectives varies greatly among existing work in literature. Typical atomic objective functions include the number of placed instances, the amount of consumed computational resources, the number of active nodes, cumulative delay and number of hops for all requests, and the number of violated service level agreements. These may either be considered individually, consolidated by applying weights, or be treated by a multi-objective optimizer.

Note that the choice of objective functions determines the optimal solutions. In order to assess the quality of placements, they need to be specified beforehand.

3 Approaches for Solutions and their Assessment

This section outlines different strategies to solve the VNFCP and common ways to evaluate their performance.

3.1 Exhaustive Optimization

The most evident approach is to compute an exact solution to a given optimization problem, or the exhaustive Pareto frontier in a multi-objective context. This strategy is sometimes implemented in a simple brute force manner, but more frequently realized by Integer Linear Programs (ILPs) for performance reasons.

Given their exact results, a qualitative evaluation is not necessary for exhaustive approaches. They always provide the *best* results with regard to their respective optimization objective. However, since the VNFCP problem is NP-hard, these methods are usually limited in applicability. Hence, they can be evaluated

with respect to the supported details of their model, their required computational resources, and the runtime of an optimization, while also considering the supported scale of problem input. More importantly, the results of these exhaustive solvers can be used as a reference for the evaluation of faster, empirical approaches, as explained in Sect. 3.3.

Example ILP. In the following, an example ILP is presented that primarily minimizes CPU utilization. It can be adjusted and used on small problem instances for the evaluation of heuristics with similar problem models.

Variables. The program is based on three types of decision variables: $c_{r,f,n}$ and $z_{r,f,i}$ indicate the location and number of an instance, and $a_{r,f,e}$ indicates the path of a request r, with f being the number of the corresponding function in the request's chain, n being a node in the network graph, e being an edge, and i the number of the instance on the respective node. In addition, several auxiliary variables are used to ease the definition of constraints and objectives: $m_{r,f,n,i}$ indicates whether the i-th instance of its type on node n is used for function f of request r. Similarly, $m_{t,n,i}$ indicates whether *any* request with a function of type t uses this instance. Finally, $m_{t,n}$ contains the *number* of instances of type t on node n. All used variables and indices are summarized in Table 1.

Table 1. Variables and indices used in the ILP formulation.

Index	**Description**
$r \in \{1, ..., \lvert R\rvert\}$	Traffic request
$f \in \{1, ..., \lvert r_c\rvert + 1\}$	Function in the chain of a request
$n \in \{1, ..., \lvert V\rvert\}$	Node in the network graph
$e \in \{1, ..., \lvert E\rvert\}$	Edge in the network graph
$t \in \{1, ..., \lvert T\rvert\}$	Type of the respective network function
$i \in \{1, ..., \lvert R\rvert\}$	Instance of the respective type on the respective node
Variable	**Description**
$c_{r,f,n} \in \{0, 1\}$	Indicates whether function f of request r is served in node n
$z_{r,f,i} \in \{0, 1\}$	Indicates whether function f of request r is served on the i-th instance of its type
$a_{r,f,e} \in \{0, 1\}$	Indicates whether edge e is used by function f of request r
$m_{r,f,n,i} \in \{0, 1\}$	$c_{r,f,n} \wedge z_{r,f,i}$
$m_{t,n,i} \in \{0, 1\}$	$\max_{(r,f) \in \{(r,f) \mid \text{ funct. } f \text{ of req. } r \text{ is of type } t\}} \{m_{r,f,n,i}\}$
$m_{t,n} \in \mathbb{N}$	$\sum_{i=1}^{\lvert R\rvert} m_{t,n,i}$

Note that each subpath from function $f - 1$ to function f is modeled separately, with $f = |r_c|$ being the last function, and $f = |r_c| + 1$ representing the destination of the traffic request. The latter is only used by the variables $a_{r,f,e}$ for the last subpath.

Objective. The primary objective is to minimize the CPU utilization, but on a second priority, the number of hops was included, e.g., to avoid loops in the paths. Using the variables above, the objective is expressed as follows.

$$\text{Minimize} \quad 0.99 \sum_{t=1}^{|T|} \sum_{n=1}^{|V|} m_{t,n} \cdot t_{\text{cpu}} + 0.01 \sum_{r=1}^{|R|} \sum_{f=1}^{|r_c|+1} \sum_{e=1}^{|E|} a_{r,f,e} \tag{1}$$

Thereby, the weights are chosen such that the amount of hops does not impair the minimization of the primary objective, i.e., the number of used CPU resources.

Constraints. The following constraints are used by the ILP to ensure path consistency and to respect resource and capacity utilization. Given an edge $e \in \{1, ..., |E|\}$, let e_1 and e_2 be the respective nodes that it connects. Further, let $L(n)$ be the set of incident edges of node n, $F(t)$ be the set of all functions (r, f) with type t, and $\tau(r, f)$ be the type t of the f-th function of the request r. Note that special cases, such as the subpath towards the final destination of a request, are omitted to retain clarity. For possible values for the indices r, f, n, e, t, and i, see Table 1.

$$\forall r, f : \sum_{n=1}^{|V|} c_{r,f,n} = 1 \text{ and } \sum_{i=1}^{|R|} z_{r,f,i} = 1 \tag{2}$$

$$\forall r, f, e : a_{r,f,e} \leq \sum_{x \in L(e_1) \setminus e} a_{r,f,x} + c_{r,f,e_1} + c_{r,f-1,e_1}$$
$$\text{and } a_{r,f,e} \leq \sum_{x \in L(e_2) \setminus e} a_{r,f,x} + c_{r,f,e_2} + c_{r,f-1,e_2} \tag{3}$$

$$\forall r, f, n : c_{r,f,n} \leq \sum_{x \in L(n)} a_{r,f,x} + c_{r,f-1,n}$$
$$\text{and } c_{r,f-1,n} \leq \sum_{x \in L(n)} a_{r,f,x} + c_{r,f,n} \tag{4}$$

$$\forall n : \sum_{t=1}^{|T|} m_{t,n} \cdot t_{\text{cpu}} \leq n_{\text{cpu}} \tag{5}$$

$$\forall e : \sum_{r=1}^{|R|} \sum_{e=1}^{|E|} a_{r,f,e} \cdot r_{\text{bw}} \leq e_{\text{bw}} \tag{6}$$

$$\forall t, n, i : \sum_{(r,f) \in F(t)} m_{r,f,n,i} \cdot r_{\text{bw}} \leq t_{\text{bw}} \tag{7}$$

$$\forall r : \sum_{f=1}^{|r_c|+1} \sum_{e=1}^{|E|} e_{\text{d}} \cdot a_{r,f,e} + \sum_{f=1}^{|r_c|} \tau(r, f)_{\text{d}} \leq r_{\text{d}} \tag{8}$$

Hereby, Eq. 2 ensures that, for every requested function, there is exactly one location and one instance assigned. Equation 3 ensures that the paths are connected: for every incident node of a link e, there is at least one other link x in use,

or the node is an endpoint of this subpath. With Eq. 4, all used instance locations enforce the usage of an incident link, ultimately leading to another instance location (or the request's destination) with the above constraints. The set of Eqs. 5, 6, and 7 respect the available CPU resources on nodes, the available bandwidth on links, and the available computational power of instances, respectively. Finally, Eq. 8 maintains the service level agreements with respect to the flows' delays.

3.2 Approximation Algorithms

Approximation algorithms provide solutions with *guaranteed* quality bounds. They are designed such that their results are provably within a multiplicative (or sometimes additive) distance from the optimal solution. However, the approximability of the VNFCP problem varies greatly with its details and assumptions, such as the actual objective of the optimization and considered constraints. There are only few publications in literature that propose an approximation algorithm for the VNFCP problem in polynomial time (cf. Sect. 6).

Given their provable bounds, the evaluation of approximation algorithms is primarily of analytical kind and deals with their worst case performance. Assessing their expected, average performance by analytical means is difficult as the definition of an *expected* problem scenario itself is unclear. However, they can be evaluated by empirical means with respect to selected representative problem characteristics from real world scenarios. Therefore, similar approaches as described in Sect. 3.3 can be applied here as well.

3.3 Heuristics

Heuristic algorithms attempt to find sufficiently good solutions within feasible computational efforts and time limits. These approaches vary greatly in applied strategy, requirements, and runtime, and therefore, also in their results' quality. Their underlying ideas include simple greedy heuristics, pre-calculation of desired parameters, iterative improvements, relaxation of ILPs, fixing and optimizing only parts of the whole problem, and finally meta-heuristics such as simulated annealing and evolutionary algorithms. A selection of related work in this category is presented in Sect. 6.

An empiric evaluation of heuristic algorithms is based on a comparison of the resulting objective values with chosen reference solutions. The selection of these references affects the expressiveness of the evaluation.

Comparison with other heuristics. When comparing two or more heuristic algorithms, the main statement is which algorithm performs better than the other with respect to the selected problem scenarios and objectives. However, there is no meaningful *quantitative* estimation of the overall performance of both algorithms without an *independent* reference point. Even with statements such as *"algorithm A performs x% better than algorithm B"*, they could still both be significantly worse than the optimal attainable value, which should be reflected by the evaluation. In many cases, simple baseline algorithms are used for the comparison, which further reduces its expressiveness. Nevertheless, as only

heuristics are used, the scale of the evaluation is only limited by the algorithms' capabilities, as opposed to the limitations of acquiring optimal references.

Ideal reference points. The ideal solution (or true Pareto frontier for multi-objective problems) can be used to overcome the above issues. As it is not tied to the used algorithm but only to the problem itself, it serves as an independent representative for the investigated heuristics. However, obtaining these ideal points is not trivial in itself. Exhaustive solvers as described in Sect. 3.1 are only applicable for small problems due to their runtime, but the investigation of more complex scenarios is usually more interesting. Hence, the generation of synthetic problem instances with predefined optimal solutions is proposed in this work. This enables an empirical evaluation of larger scaled problems with parameterized characteristics. At the same time, the generation of such problems is tied to predefined strategies and objectives. Two simple generation strategies are presented in Sect. 4.

4 Synthetic Problem Generation

In general, there are two conceivable techniques for the generation of problems with known solutions. On the one hand, the problem structure could be restricted in such a way that, for an aware algorithm, the computation of solutions is facilitated significantly. An example for such a strategy is given by the grid graphs in Sect. 4.1. On the other hand, the problem could be created in conjunction with, or even based on, the actual solution instead of dealing with its acquisition subsequently. This type of strategy is applied for the dynamic resource distribution in Sect. 4.2.

In any case, the presence of solutions for a problem instance requires that the objective is fixed beforehand. Consequently, most generation strategies are tied to a specific objective. On the contrary, other characteristics of the problem can be defined more flexibly. Having more parameters for the generation process provides more control of the scale and difficulty of the problem, which helps to reveal properties of the investigated algorithms.

However, in every meaningful performance evaluation, the results should be reviewed with respect to the synthetic problem's structure and its influence on them. General statements are difficult to obtain from empirical evaluations, but the observed behavior shall not only be caused by the choice of evaluation scenario. Therefore, it can be supplemented by an independent evaluation with different, smaller problem instances and optimal references obtained by an ILP, such as in Sect. 3.

4.1 Grid Graph Problem

The grid graph problem (GGP) is designed to be a simple multi-objective placement problem, minimizing both CPU utilization and number of hops simultaneously. Due to its simple structure, these measures are directly proportional to

the number of instances and the overall delay, respectively. Therefore, they may be minimized here as well.

An overview of the graph's layout is presented in Fig. 1. The positions of the nodes are based on a grid layout. There are m source and destination nodes in the network, located at the first and last stage of the graph, respectively. In each of the the n intermediate stages, there are k nodes with computational resources. Each node is only connected to its direct neighbors in the grid.

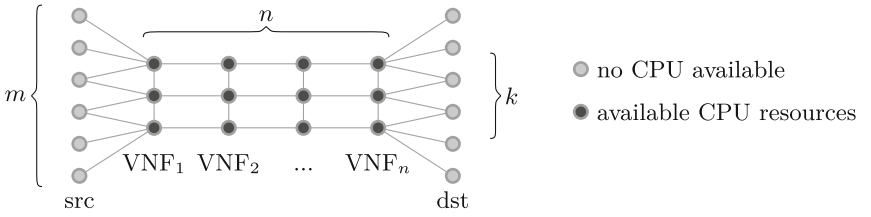


Fig. 1. Grid graph problem scenario.

The idea of this scenario is to define very similar traffic requests, with equal properties except for their source and destination, and all traffic flowing into the same direction. The requests include n different VNF types, one for each of the inner stages. Each of them has the same properties, and a single instance of each type is sufficient to satisfy all traffic requests. Each node in the n intermediate stages can host exactly one instance. Each link in the network has the same delay and sufficient bandwidth to support every request $n+1$ times. Finally, each traffic demand requests all of the n VNFs in the same order and its maximum tolerated delay allows it to visit all of them, regardless of their position.

In addition to the above parameters k, m and n, the load parameter $\rho \in [0, 1]$ controls the amount of requests in the scenario, i.e., there will be $\rho \cdot m^2$ traffic demands generated with random choices from the m available source and destination nodes. The dimensions of other measures, such as bandwidths and CPU resources, may be tweaked as well if necessary.

Optimal Solutions. In this example, the optimal placement is computed after the problem generation. The k horizontal rows of length n in the intermediate stages are referred to as *lanes*. All Pareto optimal solutions place all of the n functions in a straight lane, exactly in the order they are requested, from left to right. This avoids the introduction of unnecessary hops. As this is a multi-objective scenario, the CPU utilization is minimized as well, hence, all possible numbers $k' \in \{1, ..., k\}$ of occupied lanes are tested. For each such k', there are $\binom{k}{k'}$ possibilities for chosen lanes, thus, a total of $2^k - 1$ tests must be done.

The optimality of these solutions can be proved by contradiction: assuming the use of an incomplete row of VNFs at some point, there must be additional hops to reach the remainder of the requested service chain, which could have been avoided otherwise. Due to the limited space, only the idea is outlined here though.

4.2 Dynamic Resource Distribution

The idea of the dynamic resource distribution is to decouple the generation of the network graph from the traffic demands in order to enable a more versatile evaluation. In particular, any graph can be used without restrictions on its structure, including real topologies and randomly generated ones, however, the available resources and link bandwidths are altered in the process.

The intended objective is to minimize the number of placed instances on the graph. In its current implementation, the strategy is limited to a single VNF of the same type for each request, however, improvements are planned for future work. The generation process is comprised of the following steps.

1. Generate or choose an existing network topology.
2. Distribute a predefined number of instances on a predefined number of nodes in the network.
3. Generate traffic demands with random source-destination-pairs until all instances are fully used.
   (a) Pick a random source-destination-pair from all node pairs.
   (b) Pick one of the VNFs with remaining capacity that implies the smallest detour from the shortest source-destination-path.
   (c) Select the requested bandwidth for this traffic demand from a predefined range. Ensure optimality by filling instances up: If the remaining capacity of the instance would be too small to allow another traffic demand, increase the bandwidth such that it is fully utilized.
   (d) Define the maximum tolerated delay within a factor of the selected path's latency, e.g., the twofold.
4. Distribute sufficient CPU resources in the network such that all intended instances can be placed. Add further resources to allow more variation from the evaluated algorithms. This includes increasing existing resources as well as adding new resources to previously unused nodes.
5. Define the link bandwidths such that all selected paths are supported. Multiply the required amount by a predefined factor to enable more versatile results from the evaluated algorithms.

As evident from this process, the optimal solution is constructed in conjunction with the problem. More accurately, it is defined in steps (2) and (3b). Optimality of this problem-solution-pair is achieved by (3c), which ensures that all intended instances are fully utilized, and therefore, this is the minimum required number.

Despite its flexible definition, the generated problems are currently limited in variation with regard to requested VNF chains. The generation of longer chains and the possibility to support more objectives will be investigated in future work.

5 Evaluation

In order to show the benefits of synthetic problem generation, the strategies from Sect. 4 are demonstrated by means of a few examples. Therefore, the algorithms from [2,3] are applied.

Multi-Objective Quality Indicators. For the evaluation of multi-objective problem scenarios such as the grid graph problem, all n obtained solutions $s_{1..n} \in \mathbb{R}^n$ are aggregated into a single quality indicator value $q \in \mathbb{R}$ and compared directly. As this accompanies a loss of information, multiple types of indicators that represent different performance aspects are usually used. In this work, the hypervolume I_H and epsilon indicator I_ϵ are applied [4–6]. Hereby, the hypervolume indicator measures the volume in the solution space enclosed by the returned set and a reference point, while the epsilon indicator measures how much a reference solution set (e.g., the real Pareto frontier) must be stretched so it becomes worse than the evaluated set.

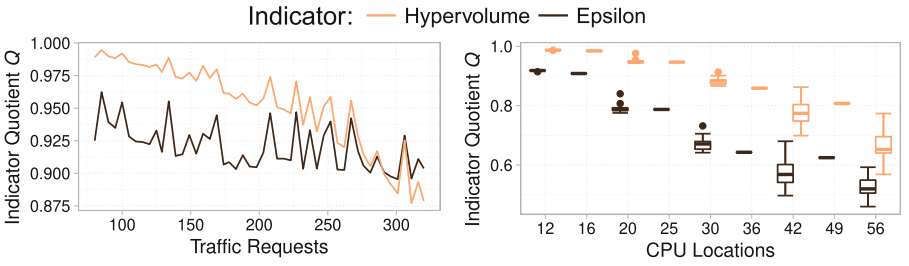
For a direct comparison with the optimal values, their ratio is computed. Note that for the hypervolume, bigger values indicate a better performance, while the opposite applies to the epsilon indicator. Hence, the indicator quotients for heuristic solutions S and optimal solutions O are defined as follows.

$$Q_H(S,O) := \frac{I_H(S)}{I_H(O)}; \qquad Q_\epsilon(S,O) := \frac{I_\epsilon(O)}{I_\epsilon(S)} \tag{9}$$

Thereby, all values are within the range $[0,1]$, where 1 represents the optimal performance with respect to this indicator.

5.1 Grid Graph Problem

With the GGP scenario, the effects of different parameters on the performance of the MO-VNFCP algorithm [3] are investigated. Firstly, Fig. 2 contains the resulting indicator ratios.



(a) Influence of the traffic requests. (b) Influence of the CPU locations number.

Fig. 2. Influence of GGP's parameters on the placement quality of MO-VNFCP.

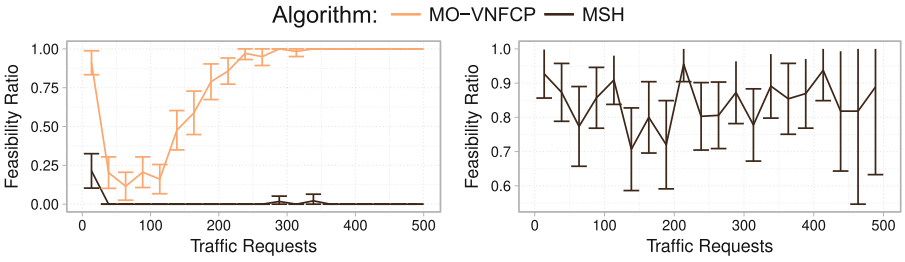
Note that, without the optima available for comparison, displaying the pure indicator values in a similar way would be significantly less expressive. By increasing the parameters of the scenario, the optimized problem instances differ between two runs, and so does their difficulty. As the optima change in a similar way, they are used to provide a relative view on the performance and a consistent overall representation of attained quality.

Figure 2a displays the influence of the number of traffic requests on the indicator quotient. The load parameter ρ is increased from 0.2 to 0.8 with $m = 20$ source and destination nodes, resulting in $\rho \cdot m^2 = 80$ to 320 requests. The parameters for the intermediate stages are $k = 4$ and $n = 3$. Both indicators show a decreasing trend in quality values, however, the hypervolume is affected stronger than the epsilon indicator, which implies that they indeed represent different aspects of performance.

In Fig. 2b, the number of nodes in the intermediate stages is varied instead, with $k \in [3, 7]$ and $n \in [4, 8]$. This leads to an increased number of locations with CPU resources available for the algorithm to choose from, but also increases the length of the embedded VNF chains. The figure displays box plots which enclose the first and third quartile of attained indicator values. The whiskers extend to the furthest point, but at most to the 1.5-fold of their boxes' height. Similarly to before, increasing the scale of the problem leads to a decrease in solution quality shown by both indicator types. However, increasing the intermediate nodes from 12 to 56 has a greater impact on both indicators compared to raising the number of requests from 80 to 320. This analysis helps to identify the algorithm's capabilities and shows opportunities for improvement.

5.2 Dynamic Resource Distribution

With the dynamic resource distribution scenario, the behavior of the algorithms MO-VNFCP [3] and MSH (Multi-Stage Heuristic) [2] are compared. Note that, while this scenario only considers the number of instances, both algorithms still try to optimize other measures simultaneously, such as the number of hops. All measurements were conducted on a real topology, namely the Germany graph [7].

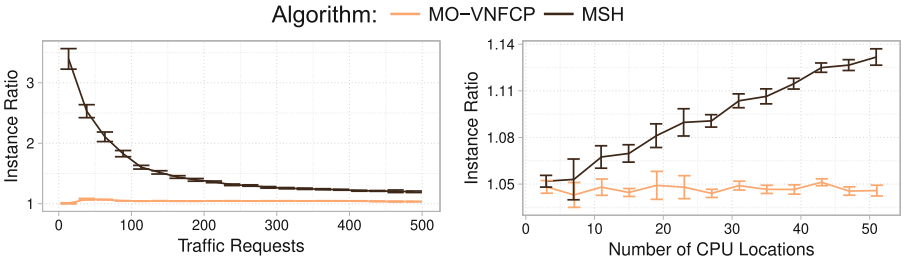


(a) Feasibility ratios with multiplier 1.5. (b) Feasibility ratios with multiplier 5.0.

Fig. 3. Feasibility ratios with different bandwidth multipliers.

As an initial quality measure, Fig. 3 displays the influence of the number of traffic demands on the feasibility ratio (i.e., the relative amount of feasible solutions within all optimization runs) for two different bandwidth parameters with 95% confidence intervals. In Fig. 3a, the available link bandwidths were set to the 1.5-fold of what the optimal solution required. Here, MSH did not return

a feasible solution in most cases as it tends to congest links. For MO-VNFCP, increasing the scale of the problem improved its solvability as with more traffic requests there are also more available resources in the graph, allowing more variation in the selected paths. In Fig. 3b, MO-VNFCP is always able to return a feasible solution, while there is no significant dependency observable for MSH. Its 95% confidence intervals range from 0.6 to 1.0 for multiplier 5.0. The following measurements were also conducted with this configuration.



(a) Influence of traffic requests.

(b) Influence of CPU locations.

Fig. 4. Influence of problem parameters on the number of instances.

Figure 4 shows the number of used VNF instances in relation to the optimal value. In Fig. 4a, the influence of the number of requests is shown. Interestingly, the *relative* performance of the MSH improves significantly with larger problems. This is caused by a broad distribution of instances by the heuristic, and its relative quality improves when the optimal placement also requires more instances. On the other hand, Fig. 4b shows the dependency on the number of nodes with available CPU resources. The more choices exist, the more difficult the problem becomes, but the optimal number of instances stays the same. Hence, the relative performance of the MSH deteriorates in this example. Figure 4 also shows that in both cases, the performance of the MO-VNFCP heuristic was very close to the optimal solution with no significant dependency on the investigated scale. Taking its longer runtime into account, it presumably requires larger or more complex scenarios to reveal its behavior, e.g., by adding more VNFs to the requested chains.

6 Related Work

This section provides an overview of publications that deal with different aspects of the VNFCP problem and focuses on the problem instances that are used for the quality assessment of the proposed algorithms. Furthermore, we present works that discuss methodologies for generating synthetic problem instances whose optimal solutions are known beforehand.

6.1 VNF Chain Placement

In [8], the authors provide one of the first formal problem statements for the VNFCP. Their evaluation is performed on a network comprised of four core routers, five switches, and 10 edge nodes. Since an ILP-based approach is utilized, optimal results are obtained. Unfortunately, such approaches can not be applied to large problem instances due to the fact that the solution space of the VNFCP grows exponentially.

For this reason, the authors of [9,10] propose heuristics in addition to ILP-based algorithms. As they analyze only a subset of all possible placements, these heuristics are capable of handling large problem instances. As a downside, they provide no guarantees with regard to the optimality of returned solutions. In order to evaluate the heuristics, the authors generate synthetic topologies using the Barabási Albert (BA) model [11] with up to 1,000 nodes and randomly generated demands. In a similar fashion, both an ILP-based approach as well as a heuristic are proposed in [12]. However, only a small network of 12 nodes and 3 function chains is considered. In [3], a multi-objective approach is used to handle conflicting objectives. A set of solutions is continuously improved similarly to the simulated annealing approach. The results are evaluated by comparing them with those of another heuristic from literature, by means of three real-world topologies and artificial demands.

In contrast to optimizing the placement of all demands simultaneously, Bari et al. [2] propose an algorithm that adds newly arriving demands to the current placement and instantiates new instances on demand. The evaluation is performed on real world networks whose size ranges from 12 to 79 nodes. Similarly, Sahhaf et al. [13] consider the dynamic scenario and evaluate their algorithm using two network graphs from the Internet Topology Zoo [14].

Rather than addressing the entire VNFCP, [15,16] address subtasks like routing of demands or mapping and scheduling them to existing VNF instances, respectively. Both approaches work in the dynamic scenario. While the former uses different graph generation models like BA or Waxman [17], the latter does not require a topology due to the assumption that delays between nodes are negligible. Furthermore, demand arrivals follow a uniform distribution and are composed of random permutations of available VNF types.

In summary, most works in literature use either synthetic or real world graphs in conjunction with artificial demand sets in order to evaluate their proposed VNFCP heuristics. However, in the context of large problem instances, optimal solutions can not be determined and thus, a quantitative statement regarding the performance of heuristics is not possible.

Approximation Algorithms. A special case is provided by the few approximation algorithms in this context [18,19]. They prove deterministic bounds for their results and may therefore omit the empirical evaluation of their algorithms.

6.2 Synthetic Problem Generation

In many areas of optimization, synthetic problem instances are used in order to perform algorithm benchmarks and tweak their performance [20–22]. To the

best of our knowledge, there are no corresponding frameworks for the VNFCP, yet. However, Virtual Network Embedding (VNE) problems [23,24] overlap with the VNFCP in terms of the chaining and placement aspects, and algorithms for problem generation are researched in a recent publication [25]. Its authors develop mechanisms for both static and dynamic scenarios whose solution is known beforehand. Since it is possible to change the number of network nodes, requests, and resources, algorithms can be analyzed in terms of aspects like scalability and behavior under different load levels.

7 Conclusion

In order to fully benefit from the flexibility of Network Function Virtualization, algorithms that tackle the arising optimization problems, in particular the Virtual Network Function Chain Placement problem, are required. Despite the large number of recent publications in this area, there is no commonly accepted standard approach yet. Therefore, an unbiased methodology for the evaluation and comparison of VNFCP algorithms is necessary to obtain meaningful statements on their performance.

This work presents an overview of the VNFCP problem, followed by possible strategies for its solution. Different concepts towards their evaluation are discussed, and the importance of optimal solutions for their comparison is emphasized. In particular, an Integer Linear Program is proposed to obtain optimal reference solutions for small problems, along with artificial problem generation strategies with known optima for the assessment of larger topologies.

Two strategies are implemented to demonstrate the applicability of the concept. Their evaluation shows the influence of problem parameters on the algorithms' performance and provides valuable insights for their improvement. However, it also reveals current limitations. Hence, future work will include the extension of existing strategies to use more complex function chains, and the support for more objective functions. Nevertheless, by using artificial problems, the expressiveness of performance assessments can be raised significantly with regard to parameterization and absolute solution quality.

Acknowledgments. This work has been performed in the framework of the CELTIC EUREKA project SENDATE-PLANETS (Project ID C2015/3-1), and it is partly funded by the German BMBF (Project ID 16KIS0474). The authors alone are responsible for the content of the paper.

References

1. Network Functions Virtualisation – Update White Paper (2013). https://portal.etsi.org/NFV/NFV_White_Paper2.pdf
2. Bari, M.F., Chowdhury, S.R., Ahmed, R., Boutaba, R.: On orchestrating virtual network functions. In: 2015 11th International Conference on Network and Service Management (CNSM), pp. 50–56 (2015)

3. Lange, S., Grigorjew, A., Zinner, T., Tran-Gia, P., Jarschel, M.: A multi-objective heuristic for the optimization of virtual network function chain placement. In: 29th International Teletraffic Congress (ITC 29), pp. 152–160 (2017)
4. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. In: IEEE Transactions on Evolutionary Computation, pp. 117–132 (2003)
5. Knowles, J., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers. Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, Technical report 214, February 2006. http://www.tik.ee.ethz.ch/pisa/?page=bugs.php
6. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point. In: Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms, pp. 87–102. ACM (2009)
7. Zuse-Institute Berlin (ZIB): SNDlib: Survivable fixed telecommunication network design library. http://sndlib.zib.de/. Accessed 29 Oct 2017
8. Moens, H., De Turck, F.: VNF-P: a model for efficient placement of virtualized network functions. In: 10th International Conference on Network and Service Management (CNSM) and Workshop, pp. 418–423 (2014)
9. Luizelli, M.C., Bays, L.R., Buriol, L.S., Barcellos, M.P., Gaspary, L.P.: Piecing together the NFV provisioning puzzle: efficient placement and chaining of virtual network functions. In: IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 98–106 (2015)
10. Luizelli, M.C., da Costa Cordeiro, W.L., Buriol, L.S., Gaspary, L.P.: A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining. Comput. Commun. **102**, 67–77 (2017)
11. Albert, R., Barabási, A.-L.: Topology of evolving networks: local events and universality. Phys. Rev. Lett. **85**(24), 5234–5237 (2000)
12. Mehraghdam, S., Keller, M., Karl, H.: Specifying and placing chains of virtual network functions. In: 3rd International Conference on Cloud Networking (CloudNet), pp. 7–13 (2014)
13. Sahhaf, S., Tavernier, W., Rost, M., Schmid, S., Colle, D., Pickavet, M., Demeester, P.: Network service chaining with optimized network function embedding supporting service decompositions. Comput. Netw. **93**, 492–505 (2015)
14. Knight, S., Nguyen, H.X., Falkner, N., Bowden, R., Roughan, M.: The internet topology zoo. IEEE JSAC **29**(9), 1765–1775 (2011)
15. Dwaraki, A., Wolf, T.: Adaptive service-chain routing for virtual network functions in software-defined networks. In: Proceedings of the 2016 workshop on Hot Topics in Middleboxes and Network Function Virtualization, pp. 32–37 (2016)
16. Mijumbi, R., Serrat, J., Gorricho, J., Bouten, N., De Turck, F., Davy, S.: Placement and scheduling of functions in network function virtualization, arXiv preprint arXiv:1512.00217 (2015)
17. Waxman, B.M.: Routing of multipoint connections. IEEE J, Sel. Areas Commun. **6**(9), 1617–1622 (1988)
18. Rost, M., Schmid, S.: Service chain and virtual network embeddings: Approximations using randomized rounding, arXiv preprint arXiv:1604.02180 (2016)
19. Lukovszki, T., Schmid, S.: Online admission control and embedding of service chains. In: Scheideler, C. (ed.) Structural Information and Communication Complexity. LNCS, vol. 9439, pp. 104–118. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25258-2_8

20. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. Springer, London (2005). https://doi.org/10.1007/1-84628-137-7_6
21. Huband, S., Barone, L., While, R.L., Hingston, P.: A scalable multi-objective test problem toolkit. In: EMO, vol. 3410, pp. 280–295 (2005)
22. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans. Evol. Comput. **10**(5), 477–506 (2006)
23. Fischer, A., Botero, J.F., Beck, M.T., De Meer, H., Hesselbach, X.: Virtual network embedding: a survey. IEEE Commun. Surv. Tutor. **15**(4), 1888–1906 (2013)
24. Fajjari, I., Aitsaadi, N., Pujolle, G., Zimmermann, H.: VNR algorithm: a greedy approach for virtual networks reconfigurations. In: Global Telecommunications Conference (GLOBECOM), pp. 1–6 (2011)
25. Fischer, A., de Meer, H.: Generating virtual network embedding problems with guaranteed solutions. IEEE Trans. Netw. Serv. Manag. **13**(3), 504–517 (2016)

Towards Optimal Placement of Monitoring Units in Time-Varying Networks Under Centralized Control

Sounak Kar(✉), Rhaban Hark, Amr Rizk, and Ralf Steinmetz

Technische Universität Darmstadt, Darmstadt, Germany
sounak.kar@kom.tu-darmstadt.de

Abstract. The increasing penetration of software-defined communication networks with centralized control has made network management a highly demanding task. Common monitoring approaches in the context of such convoluted high-speed networks have become a serious challenge in terms of complexity and resource management. Management functions rely on monitoring information such as the flow size distribution (FSD), to perform crucial activities such as load balancing and resource provisioning. In this paper, we propose a solution as to how one can utilize limited monitoring resources to estimate the FSD for distinct flows characterized by origin-destination pairs. We provide a method to dynamically adapt placement of monitoring units with some extracted knowledge about the change in FSD's with time.

1 Introduction

With the rise of software-defined networking (SDN), logically centralized network management and control have become a de facto standard architecture in communication networks. SDN enables segregation of the data plane from the control plane by separating switching and forwarding tasks from the higher context control decisions, such as routing and processing monitoring information. This results in a higher network flexibility and performance as well as a more efficient network management.

Network monitoring is an important component that feeds information into the SDN controller for decision-making. This is presented in Fig. 1, which depicts a logically centralized controller collecting monitoring information from different switches across an SDN network. The collected data may include packet counts, loss rates, packet timestamps among other information. Traffic monitoring can provide valuable insights on the timing and size of data flows, i.e., estimates of the flow size distribution (FSD) across the network. Before we elaborate upon usefulness of FSD estimation, it should be noted that there exist alternative definitions of FSD. Flow size can be characterized in terms of bytes, packets, or time-length. For our experiments, we have used the time-length definition of FSD.

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 99–112, 2018.
https://doi.org/10.1007/978-3-319-74947-1_7

The estimation of flow size distributions does not only play a role in Software-defined networks but also in the context of automated embedding of virtualized service function chains. Network function virtualization promises an embedding of virtual service functions that basically replace middleboxes and other traditional network functions in a softwarized way. Virtual network functions (VNF) are hence embedded on top of commodity hardware and may be connected through SDN slices such that network resources, now also in terms of functions, can be pooled, scaled, and migrated. The continuous optimization of this embedding, i.e., the mapping of network functions to physical resources, and hence including function migrations and scaling depends largely on the knowledge of the network state. This enables central entities, denoted as orchestrator, to control such networks of virtual function chains. Here, monitoring the flow size distribution as well as further VNF metrics is essential for network resource optimization algorithms.

In addition to the above, FSD estimation enables adaptive traffic engineering, DDoS attack detection as well as developing billing models. However, monitoring can introduce considerable overhead on the data plane switches, and on the concerned SDN controllers too. As evident from Fig. 1, collecting monitoring information from *all switches all the time* leads to accumulation of both valuable and redundant information. The problem of handling this monitoring overhead is aggravated not only by the size of the network but also by the dynamics of the traffic flows.

In this work, we study extraction of the FSD from network traffic given a limited amount of monitoring resources, i.e., a limited number of vantage points. We consider a restricted number of monitoring points to capture the impact of the overhead that arises with monitoring an increasing number of SDN switches. Here, we make use of the information on flow routing on top of the network graph to place a limited amount of monitors on the available SDN switches. These "monitors" can either be logically persistent tasks that are devolved to some SDN switches or simply the SDN switches that are polled continuously by the controller. These monitors are placed to minimize the estimation error of the FSD. In addition, given that network flows are known to possess dynamics on different time scales, we investigate the problem of adapting the placement of monitoring points in an SDN network with time. Here we probe/learn from the observations made by the placed monitors about the "importance"/contribution of the monitoring units to the estimates. Based on that we rearrange the monitoring units to maximize the precision of the estimation, which is measured by the Bhattacharyya Distance, a metric of divergence between the estimated and actual FSD.

The remainder of the paper is structured as follows: In Sect. 2 we discuss related work on the estimation of the FSD in communication networks before introducing the problem of static and dynamic monitor placement to obtain the FSD estimates in Sect. 3. Subsequently, simulation results are presented in Sect. 4 and our contribution has been summarized in Sect. 5.

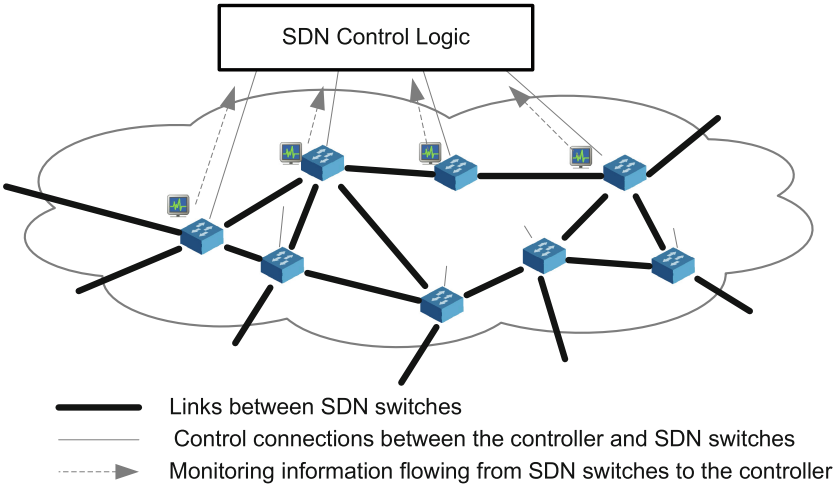


Fig. 1. Sketch of acquiring monitoring information in SDN.

2 Related Work

In this section we discuss related work on monitoring in centrally controlled networks such as SDN and orchestrated virtualized service function chains. We particular discuss the estimation of the flow size distribution in communication networks. The architecture of SDN provides the possibility for traffic engineering and DDoS detection based on monitoring tasks that are devolved to network switches. In this work we are concerned with so called passive probing techniques [1–3] which, in essence, do not inject probing traffic into the network to infer about its characteristics. Here, we rather *passively* observe the network state through polling the state on the SDN switches, which can be achieved through polling packet counters [4–6]. Extracting the monitoring information from SDN switches can also be based on offloading tasks to switches as in [7,8]. In contrast to previous works we are not particularly investigating *how* the SDN switches can be polled to extract information but rather *which* SDN switches need to be polled to obtain an accurate monitoring result - in this case - an estimate of the flow size distribution. In addition to its value for SDNs, monitoring is essential for orchestrated virtual service function chains. Here, network Function Virtualization (NFV) promises the deployment of network functions on platforms usually built of commodity hardware [9–11]. Here, auto-embedding frameworks [12,13] for virtual network slices take monitored network information to automatically assign resources to deployed network functions. Here too, estimates of the FSD can play an important role in optimizing the placement of such network functions on different physical network substrates.

It is well known that estimation of the FSD is a data intensive procedure. Most of the literature is consequently oriented towards estimating the FSD by sampling. Flow sampling was shown to be more effective than packet sampling

in the seminal work in [14]. Later, a more refined approach called dual sampling described in [15] was shown to outperform previous approaches. However, the temporal aspect of FSD estimation has not been well studied. Recently, while discussing network traffic characteristics of data centers in [16,17], authors briefly mention that the FSD can reasonably vary depending upon when and where we monitor the flows in the network. In this paper, we focus on addressing this aspect of FSD in the context of flow monitoring in an SDN enabled network. Note that positioning of monitoring units is a crucial part of this process, as increased presence of monitors entails undesirable load on the controller and additional cost. On monitor positioning, our static approach is similar to that of [18] as described in Sect. 3. However, we adapt our monitor positioning at certain intervals as described in Sect. 3. In addition, we focus on the FSD in terms of flow length and observe the entire flow until it finishes.

3 From Monitor Placement to Estimates of the Flow Size Distribution

In this section, we formally describe the problem of placing monitors in a communication network topology for the estimation of the FSD before elaborating on our resource positioning and the temporal placement adaptation strategy.

3.1 Problem Formalization

A snapshot at time t of a given network topology can be described as a collection $G(V, E, F_t)$, where the set of vertices $V = \{1, 2, 3, .., N\}$ denotes the set of nodes existing within the network, E denotes the edge set, a subset of the set of unordered pairs from V, and set of flows is defined as: $F_t = \{f_1^t, f_2^t, ..., f_{n_t}^t\}$, where n_t is the number of flows present in the network at time t. A typical element of F_t can be expressed as $f_j^t = \{a_1, a_2, ..., a_k; T_j^t\}$ with $\{a_1, a_2, .., a_k\} \subseteq \{1, 2, .., N\}$ and $T_j^t \leq t$. That is at time t, we have a directed flow f_j^t that originated from a_1 at time T_j^t with destination a_k and it passes through the path $a_2, a_3, ...$; in that order. In Fig. 2 we provide an example of such a snapshot and list out corresponding flows.

Since we consider controllers tasked with placing monitors on network nodes, we further assume the routing protocol is known and there are M available monitoring units to be placed at certain nodes in the network. In an SDN setting this corresponds to controllers devolving monitoring tasks on switches while in an Network Function Virtualization setting this corresponds to an orchestrator placing monitors on virtual function chains. In this work we consider that these monitoring units examine flows passing through them and are capable of discerning between flows with different origin-destination labels and/or origination time. Through these monitoring units, we try to infer about the flow size distribution for specific sender-receiver pairs of nodes, which we express as distribution of flow lengths in time. To emphasize, flows are defined in this work in a generic fashion, i.e., given header information flows can be identified using the

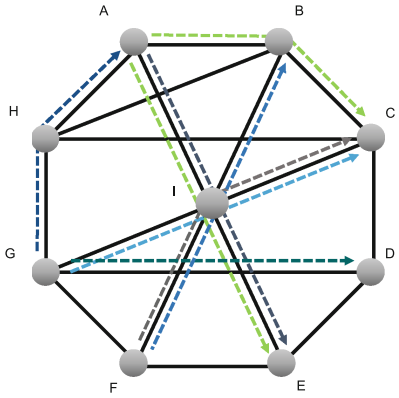


Fig. 2. Snapshot of a network topology at a particular time t with 9 nodes $\{A, B, ..., I\}$ and 9 active flows. A dark solid edge denotes connectivity between two nodes whereas colored dashed arrows represent the 9 flows. $\{G, H, A; t_l\}$ is one such flow with origination time $t_l \leq t$. Note that flows $\{A, I, E; t_m\}$ and $\{A, I, E; t_o\}$ with the same sequence of incident nodes $\{A, I, E\}$ are distinguishable by their origination times t_m and t_o, respectively.

standard 5-tuple. The framework also applies (with less computational complexity) to aggregated form identified only using sender/destination addresses. Our objective is to minimize the divergence between our estimate of the FSD and the actual FSD for all active flows. We aim to achieve this through the following heuristic algorithm:

- Identify active flows within the network and allocate resources to ensure maximum monitoring coverage.
- Adapt to changing flows as fast as possible. A specific resource placement remains in force only for a necessary time required to obtain a good estimate for the corresponding FSD.

We describe this algorithm in detail in the following subsections. We emphasize that the second step is more crucial due to the fact that the flow size distribution naturally changes over time.

3.2 Static Placement at a Fixed Time

Placing monitoring units in a network can be shown to be equivalent to the set-cover problem where the flows represent elements and the nodes can be thought as relevant subsets. This is known to be NP-complete. We adapt a greedy algorithm for finding the maximal cover, to our context. To this end, we define following quantities: number of flows passing through a node at time t and the corresponding set of flows.

Definition 1. *Given network topology $G(V, F_t)$ at time t, number of flows passing through the node i, is defined as $f(i \mid F_t) = \sum_{f_j^t \in F_t} \mathbb{1}(i \in f_j^t)$ for $i \in V, F_t = \{f_1^t, f_2^t, ..., f_{n_t}^t\}$.*

Definition 2. *Given network topology $G(V, F_t)$ at time t, the set of flows passing through node i, is defined as $C(i \mid F_t) = \{f_j^t \in F_t : i \in f_j^t\}; i \in V, F_t = \{f_1^t, f_2^t, ..., f_{n_t}^t\}$.*

We note that the direction of a flow only helps us distinguish it from the rest and barely has any implication when it comes to monitoring. Thus, with the objective of maximizing the number of flows covered, we should sequentially place the monitors at 'busy' nodes, as described in the following algorithm.

Algorithm 1. Placement algorithm (Greedy) at a fixed time t

Input : Network topology $G(V, F_t)$, number of monitoring units M
Output: Set of monitoring nodes $P^* \subset V$ such that $|P^*| \leq M$

$P^* = \emptyset, A = \emptyset, k = 1$;
while $A \neq F_t$ *and* $k \leq M$ **do**
  $a = \arg\max_{j \in V \setminus P^*} f(j | F_t \setminus A)$;
  $P^* = P^* \cup \{a\}$;
  $A = A \cup C(a | F_t \setminus A)$;
  $k = k + 1$;
end

In essence, Algorithm 1 seeks to maximize coverage with resources at its disposal sequentially. At the outset, it identifies the node incident with maximum number of active flows and places the first monitor there, ensuring all the flows passing through it (and originating from/ending at it) are covered. In the next step, the algorithm focuses on the residual flows and finds the node incident with the maximum number of residual flows, thereby maximizing the total number of flows covered so far. The same operation is carried out iteratively until we exhaust all our monitoring units or the flows in the network. At any step, if we have more than one candidate node to place our monitor, we pick one uniformly at random. We also provide an illustration of Algorithm 1 in Fig. 3.

Under the usual assumption of shortest path routing with edge weights representing the capacities of the links, the overlay network is defined as the collection of shortest path trees (SPT). It is noteworthy that in absence of multiple flows between the same sender-receiver pair, the performance guarantee of the first monitoring unit is closely related to highest betweenness centrality of the overlay network.

3.3 Adapting Monitor Placement with Time

As flow composition changes over time, ideal placement of monitoring units changes as well. This demands a rearrangement of the placed monitoring units.

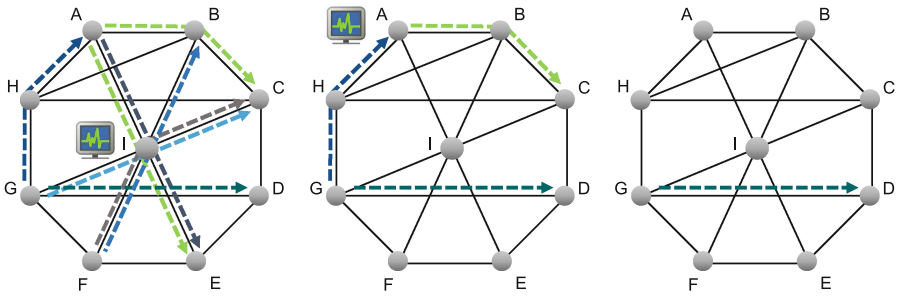


Fig. 3. We start with the snapshot mentioned in Fig. 2 and with 2 monitors to place. We see that I is incident with maximum number of flows (5) and by placing a monitor at I we are left with 3 residual flows. Thereafter, A and G both have 2 flows passing through them and we randomly pick A. Since we have exhausted our monitoring resources, we can't monitor $\{G, D\}$.

Again, to estimate the FSD with a certain level of precision, we need to keep these units stalled for a certain period of time to gather requisite amount of flow size samples. In this work, we measure the precision of the estimation using the *Bhattacharyya distance* between the estimated and the actual distribution as defined in the Appendix. The distance assumes zero value when the estimated distribution coincides with ground truth and higher value indicates greater dissimilarity.

Further, even with the same set of flows through the network, we need to update our estimate of the FSD owing to its temporal shifts. Hence, we move through a cycle of probing and estimation phases as sketched in Fig. 4. In the probing phase, we discover flows present in the network and accordingly decide on the placement of monitors using Algorithm 1. Note that with M monitors, it takes $\lceil N/M \rceil$ phases to probe all nodes in the network. At the end of each estimation phase, we estimate the FSD of all incident flows through the monitors, where the time-length required for the estimation is defined in Definition 3. Note that, in one estimation phase, we do not take observations from previous estimation phases into consideration as they might already be stale. However, information gathered over the estimation phase is utilized for probing as well, as highlighted through the decreased number of probing slots in Fig. 4.

Definition 3. *Given prior knowledge of m possible candidates for FSD, $\{F_1, F_2, ..., F_l\}, X_j \sim F_j$; essential time to estimate FSD is defined as time required to observe p^{th} quantile of $X_j, j \in \{1, 2, ..., l\}$ in the worst case, i.e., $T(\{F_1, F_2, ..., F_l\}, p) = \max_{i \in \{1,2,...,l\}} \mathbf{E}(X_i)/(1-p)$,*
where the last expression follows from Wald's Identity mentioned as Lemma 1 in the Appendix.

The definition above is based on the idea that until the time we observe a value beyond a large quantile, it is expected that we observe lower quantiles meanwhile. Hence, the sample observed would be representative of the population. The less

we observe, the more prone the method becomes to losing out higher quantiles. *It is, of course, more useful to observe longer.* However, in the context of flow changes and temporal variation in the FSD's, we need to keep updating the gathered estimates. Hence we wait for a defined essential time before making rearrangements, to come up with a relevant estimate. Note that the observation obtained until this essential time leads to an unbiased estimate only when the actual FSD does not change within the observation window. To deal with this, one may discard undesired samples within this window by using change detection principles [19], discussed further in Sect. 4.3.

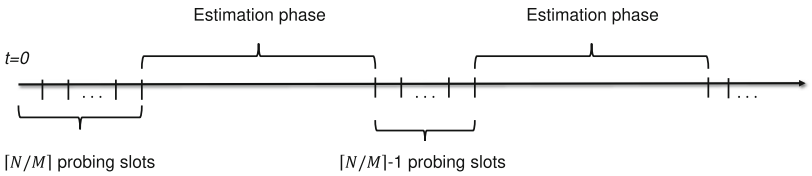


Fig. 4. Successive phases of probing and estimation. Initially, $\lceil N/M \rceil$ probing rounds are required to probe the whole network. In later phases, we can use data from estimation phase to reduce required number of rounds by 1. Length of estimation phase is determined by Definition 3.

4 Numerical Evaluation

In this section, we discuss the used evaluation setup before presenting numerical results that show the strength of our FSD estimation algorithm. We measured the performance of our approach first in terms of coverage and then by showing the divergence of the estimated FSD from the ground truth.

4.1 Evaluation Setup

We simulate the underlying network topology as an Erdös-Rényi graph $G(n, p)$, with parameters $n = 20, 50, 100$ and $p = 0.4, 0.8$. Recall that $p > \log(n)/n$ asymptotically guarantees connectedness. After ensuring connectedness, we simulate edge-weights independently as uniform distributed $U(0, 1)$. We subsequently find the shortest-path tree corresponding to each source node using Dijkstra's algorithm and form the overlay routing trees, denoted as $G_{\cup SPT}(n)$.

Given the topology, we simulate flows within the network for a discretized time interval of 1−200 s. At each time point, we uniformly choose a certain fraction of the $2\binom{n}{2}$ sender-receiver pairs to be active and initiate a flow in the corresponding routing path in $G_{\cup SPT}(n)$. The length of the flow (in time) is simulated using a shifted Poisson distribution (P), where the parameter of the distribution varies over time according a Markov Chain with a diagonally

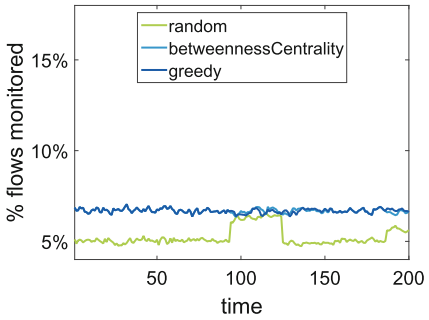
dominated transition matrix. *This is done to reflect the change in FSD over time with a reasonable amount of stickiness.* In our case, $\mathbf{E}(P) \in \{1, 2, 3\}$ and thus expectation of actual flow length $L = 1 + P$, belongs to the set $\{2, 3, 4\}$. This is to ensure that we do not include zero as a valid flow length. Thus, we have the simulated flows in the network at each time point. For computational simplicity, we assume that at a given point in time, all the sender nodes use the same Poisson parameter to generate a flow. Note that a flow generated at time t with length k persists till time $t + k - 1$ and we assume our monitoring units are capable of distinguishing this from other flows between the same sender-receiver pair initiated in the meantime using the standard 5-tuple of sender/receiver IP address and ports as well as the protocol identifier.

4.2 Performance of Adaptive Flow Size Distribution Estimation

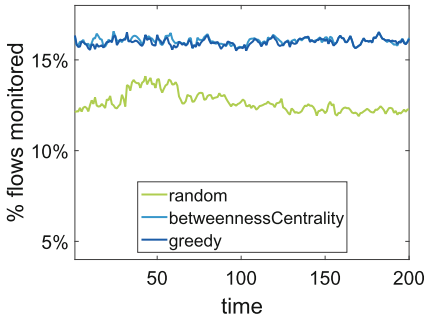
We present in Fig. 5 sample illustrations that compare our strategy (denoted in the following figures as **greedy**) to *(i)* a random placement algorithm and *(ii)* to the policy where we place monitors at the nodes with highest betweenness centrality. We again highlight that for greedy strategy, available monitoring units are placed according to Algorithm 1 and incident flows are observed for a time period determined using Definition 3. At the end of estimation phase, we again probe and accordingly rearrange the monitors as highlighted in Fig. 4. Note that the rearrangement of monitoring units takes place at the same interval for the random placement scheme as well. This is because the third strategy is agnostic of the flows in the network and hence the placement remains static. Strategies are compared in terms of the fraction of flows covered over time. Note that Fig. 5 compares strategies only in terms of flow coverage over time and thus focuses more on the efficacy of monitor placement. Effectiveness of the 'greedy' strategy becomes more conspicuous as we have greater number of monitoring units in place. The accuracy of FSD estimation for the greedy strategy is further presented in Fig. 6. For our simulations, we have explored the effect on performance for following factors:

- The number of nodes n in the network. We take $n \in \{20, 50, 100\}$.
- Connectivity of the underlying network characterized by the edge probability p of $G(n, p)$. For our simulations, $p \in \{0.4, 0.8\}$.
- Flow probability q between a given sender-receiver pair. We choose $q \in \{0.2, 0.4, 0.7\}$ to denote moderate, high and very high regimes of traffic.
- Number of available monitoring units. We have simulated for $\{1, 2, ..., 10\}$.

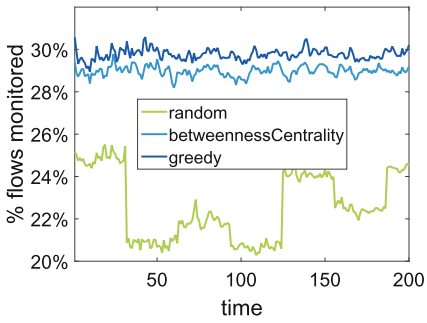
Next we illustrate the divergence from the ground truth FSD for a randomly chosen sender-receiver pair that has an active flow. This is dependent on the flow probability as it dictates the number of flows we observe before one (re)arrangement. Further, it is also influenced by the assumptions of flow structure and hence, the amount of time we wait before rearranging. Note that while calculating the distance, we use the distribution used to simulate flows at the



(a) Number of monitoring units = 2



(b) Number of monitoring units = 5



(c) Number of monitoring units = 10

Fig. 5. Performance comparison of positioning strategies in very high flow regime. Simulation setting: $n = 100$, $p = 0.4$, $q = 0.7$. Our strategy is marked as 'greedy' in the legend. Advantage of 'greedy' strategy becomes noticeable as we tend to increase number of monitoring units.

beginning of a phase as ground truth. The corresponding Bhattacharyya distance has been plotted in Fig. 6. As mentioned in Sect. 3.3, the smaller the value of the distance, the closer the estimated FSD is to the actual FSD. Instances of higher distance may be due to distribution shifts occurring in the middle of an estimation phase, which does not let resulting sample remain a true representative of actual FSD anymore.

We further investigate the incremental benefit with an increasing number of monitoring units under different parameter conditions. Figure 7 shows the incremental benefit of placing an additional monitor through box plots where the underlying variable of interest is the fraction of flows in the network covered over time.

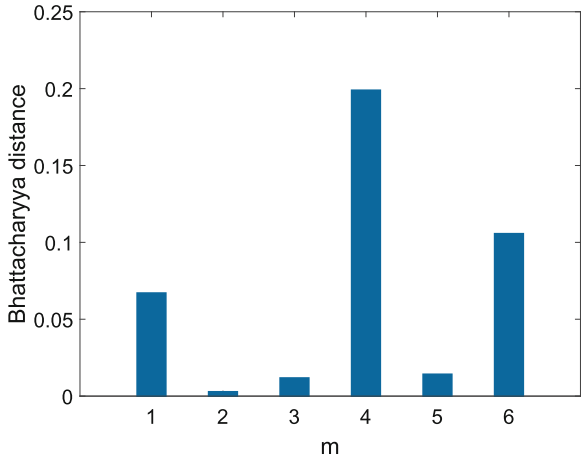


Fig. 6. Bhattacharyya distance vs. the estimation phase number m. Simulation settings $n = 100$, $p = 0.4$, $q = 0.4$. Higher distance corresponds to less proximity between the estimated flow size distribution and the ground truth. Increasing Battacharyya distance can be attributed to distribution shifts within an estimation phase.

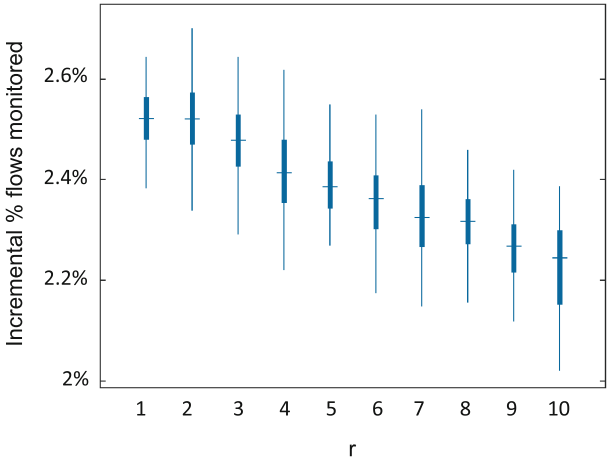


Fig. 7. Incremental benefit in terms of coverage by placing $r{-}th$ monitor with $r \in \{1, 2, ..., 10\}$. Simulation setting: $n = 100$, $p = 0.4$, $q = 0.4$. Benefit accrual rate starts to diminish beyond a certain point as the next 'busiest' node tends to get considerably less 'busy'.

4.3 Discussion

As apparent in Fig. 6, the divergence of the estimated FSD from the actual FSD is high for some estimation phases. This is due to the fact that the actual FSD changes in between causing inclusion of samples from different distributions.

We are aware that our estimation algorithm is precise only when distribution shifts do not occur in between rearrangements. A good way to get around this problem would be to detect regime shift as described in [19] and discard the samples beyond the detected change point. In general, this approach has the possibility of ending up with too little samples. A further open question is finding a performance guarantee for Algorithm 1, especially in presence of multiple monitors.

5 Conclusion

In this work, we explored a method for time adaptive monitor placement in centrally controlled communication networks such as SDN, for estimation of the FSD. In centralized networks, network control relies on monitoring information like the FSD to perform certain activities, such as traffic engineering and resource provisioning. However, an increased amount of monitoring for information accumulation inevitably entails significant overhead, especially when monitoring data is gathered from *all* switches in the network. In this work, we proposed a method for utilizing a limited number of monitoring resources to estimate the FSD for distinct flows in SDNs. Further, we provided a method to dynamically adapt the placement of monitors with time using extracted knowledge about the change in FSD's with time. Our results show that adapting the monitoring placement with time yields better performance than static centrality based monitoring placement when the number of monitors increases. In case of small number of monitors, static centrality based methods yield comparable results, which we contribute to the structure of the shortest path routing. Finally, we show decreasing incremental benefit of placing additional monitoring units in the network.

Acknowledgments. This work has been funded in parts by the German Research Foundation (DFG) as part of project B4 within the Collaborative Research Center (CRC) 1053 – MAKI. This work has been performed in parts in the framework of the CELTIC EUREKA project SENDATE-PLANETS (Project ID C2015/3-1), and it is partly funded by the German BMBF (Project ID 16KIS0471).

Appendix

Lemma 1. ***Wald's Identity*** *[20]: Suppose for a sequence of real-valued random variables* $\{X_n\}$ *and nonnegative integer-valued random variable* N *following conditions hold,*

1. $E[\mid X_n \mid] < \infty$,
2. $E[X_n \mathbb{1}(N \geq n)] = E[X_n] \cdot P(N \geq n), \forall n$ *and*
3. $\sum_{n=1}^{\infty} E[X_n \mathbb{1}(N \geq n)] < \infty$.

Then, the random sums $S_N := \sum_{n=1}^{N} X_n$ *and* $T_N := \sum_{n=1}^{N} E[X_n]$ *are integrable and* $E[S_N] = E[T_N]$. *Additionally, if*

4. $E[X_n] = E[X_1] \ \forall n$ *and*
5. $E[N] < \infty$

$E[S_N] = E[N] \cdot E[X_1]$.

In our case, $X_i's$ are i.i.d according to a distribution F with finite mean and N is the time to observe $p-th$ percentile of F. Hence, $E[X_n \cdot \mathbb{1}(N \geq n)] = E[X_n \cdot \mathbb{1}(X_1 < F^{-1}(p), ..., X_{n-1} < F^{-1}(p))] = E[X_n] \cdot E[\mathbb{1}(X_1 < F^{-1}(p), ..., X_{n-1} < F^{-1}(p))] = E[X_n] \cdot P(N \geq n)$, owing to independence of $X_i's$. Condition (3) is satisfied as the summands form a geometric series with $n-th$ term being $E[X_1] \cdot p^n$. Thus, $E[S_N] = E[X_1] \cdot E[N] = E[X_1]/(1-p)$.

Definition 4. ***Bhattacharyya Distance:*** *Bhattacharyya distance between two probability mass functions (pmf) p and q over same domain X, is defined as* $D(p, q) = -\log\left(\sum_{x \in X} \sqrt{p_x \cdot q_x}\right)$.

Compared to the popular metric *Kullback-Leibler divergence*, this distance is defined even when empirical pmf assigns zero mass to a point that belongs to the support of actual or hypothesized distribution.

References

1. Benko, P., Veres, A.: A passive method for estimating end-to-end TCP packet loss. In: Proceedings of the IEEE GLOBECOM, vol. 3, pp. 2609–2613 (2002)
2. Papadogiannakis, A., Kapravelos, A., Polychronakis, M., Markatos, E.P., Ciuffoletti, A.: Passive end-to-end packet loss estimation for grid traffic monitoring. In: Proceedings of the CoreGRID Integration Workshop, pp. 79–93 (2006)
3. Friedl, A., Ubik, S., Kapravelos, A., Polychronakis, M., Markatos, E.P.: Realistic passive packet loss measurement for high-speed networks. In: Papadopouli, M., Owezarski, P., Pras, A. (eds.) TMA 2009. LNCS, vol. 5537, pp. 1–7. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01645-5_1
4. Tootoonchian, A., Ghobadi, M., Ganjali, Y.: OpenTM: traffic matrix estimator for openflow networks. In: Krishnamurthy, A., Plattner, B. (eds.) PAM 2010. LNCS, vol. 6032, pp. 201–210. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12334-4_21
5. Hark, R., Stingl, D., Richerzhagen, N., Nahrstedt, K., Steinmetz, R.: DistTM: collaborative traffic matrix estimation in distributed SDN control planes. In: Proceedings of the IFIP Networking, pp. 82–90 (2016)
6. Bozakov, Z., Rizk, A., Bhat, D., Zink, M.: Measurement-based flow characterization in centrally controlled networks. In: Proceedings of the IEEE INFOCOM (2016)
7. Shirali-Shahreza, S., Ganjali, Y.: FleXam: flexible sampling extension for monitoring and security applications in openflow. In: Proceedings of the ACM HotSDN, pp. 167–168 (2013)
8. Yu, M., Jose, L., Miao, R.: Software defined traffic measurement with OpenSketch. In: Proceedings of the USENIX NSDI, pp. 29–42 (2013)
9. OPNFV. https://www.opnfv.org/
10. T-NOVA: Project. http://www.t-nova.eu/
11. SONATA: Project. http://www.sonata-nfv.eu/

12. Dietrich, D., Rizk, A., Papadimitriou, P.: AutoEmbed: Automated multi-provider virtual network embedding. In: Proceedings of ACM SIGCOMM 2013 (Demo track), pp. 465–466 (2013)
13. Houidi, I., Louati, W., Ben Ameur, W., Zeghlache, D.: Virtual network provisioning across multiple substrate networks. Comput. Netw. **55**(4), 1011–1023 (2011)
14. Hohn, N., Veitch, D.: Inverting sampled traffic. In: Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement. IMC 2003, New York, pp. 222–233. ACM (2003)
15. Tune, P., Veitch, D.: Fisher information in flow size distribution estimation. IEEE Trans. Inf. Theory **57**(10), 7011–7035 (2011)
16. Benson, T., Akella, A., Maltz, D.A.: Network traffic characteristics of data centers in the wild. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement. IMC 2010, New York, pp. 267–280. ACM (2010)
17. Roy, A., Zeng, H., Bagga, J., Porter, G., Snoeren, A.C.: Inside the social network's (datacenter) network. SIGCOMM Comput. Commun. Rev. **45**(4), 123–137 (2015)
18. Yoon, S., Ha, T., Kim, S., Lim, H.: Scalable traffic sampling using centrality measure on software-defined networks. IEEE Commun. Mag. **55**(7), 43–49 (2017)
19. Tartakovsky, A.G., Moustakides, G.V.: State-of-the-art in Bayesian changepoint detection. Seq. Anal. **29**(2), 125–145 (2010)
20. Karlin, S., Taylor, H.: A First Course in Stochastic Processes. Academic Press, New York (1975)

Estimating the Flow Rule Installation Time of SDN Switches When Facing Control Plane Delay

Anh Nguyen-Ngoc(✉), Stanislav Lange, Stefan Geissler, Thomas Zinner, and Phuoc Tran-Gia

University of Würzburg, Würzburg, Germany
{anh.nguyen,stanislav.lange,stefan.geissler,
zinner,trangia}@informatik.uni-wuerzburg.de

Abstract. The software defined networking (SDN) paradigm has numerous benefits for network operators, including cost aspects, flexibility, and programmability. Using the SDN approach of a separated control and data plane, the controller can order the installation of flow rules in the switches it manages, e.g., via FlowMod messages in case of the OpenFlow protocol. Since the processing time of these messages has a direct impact on the reaction time of the network, it is a key performance indicator for switches and quantifying it in a reliable manner is required for ensuring state consistency between the control and the data plane. Furthermore, real world deployments not only consist of different data plane hardware, but may feature varying control plane delays.

Hence, in this work, we investigate the impact of such a delay on the FlowMod processing time of OpenFlow switches. Firstly, we identify a significant heterogeneity between data plane hardware in terms of processing times as well as the underlying TCP-level behavior. Secondly, we show that despite this heterogeneity, combining switch specific information with delay measurements at the controller can be used to reliably infer FlowMod processing times.

We confirm our results with measurements in a dedicated testbed that is comprised of three different hardware switches, three different SDN controllers, and several high precision measurement devices.

Keywords: OpenFlow · FlowMod · Transmission delay · SDN

1 Introduction

Background. Several aspects of today's networks are affected when the paradigm of software defined networking (SDN) is employed. In addition to the separation of control and data plane, the SDN architecture is characterized by a logically centralized control plane. The latter is achieved by migrating control plane functionality from the network devices to a dedicated controller,

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 113–126, 2018.
https://doi.org/10.1007/978-3-319-74947-1_8

i.e., software that runs on commercial off-the-shelf (COTS) hardware. The two planes communicate via protocols like OpenFlow [1], which implement the open southbound API [2].

Goal. Prior to migrating their network to an SDN-based deployment, operators need to assert that the resulting network meets the performance requirements for the particular use case. On the one hand, such requirements include data plane aspects like packet forwarding speed. On the other hand, control plane performance characteristics such as the processing time of FlowMod messages in OpenFlow switches play an important role. In our previous work [3], we investigate different approaches for assessing these processing times during the network run time. These include measurements within the SDN controller module, packet dumps on the controller host, as well as measurements from dedicated wiretaps that are placed in the network. However, even a dedicated control plane channel might be subject to latency in a real world deployment. Consequently, the goal of this work consists of analyzing the impact of control plane delay on the FlowMod processing times and the estimation accuracy of proposed methods.

Key insights. Firstly, our experiments show that given knowledge regarding the current control plane delay and switch hardware, it is possible to accurately infer the time until FlowMods are active in the data plane of the switch. Secondly, we observe that the limited buffer size of many hardware switches leads to TCP flow control behavior that significantly reduces the throughput of FlowMod messages and thus, the time until flow rules are installed. Finally, we show that the controller implementation can also have a significant impact on the performance w.r.t. the flow setup time due to different sending and packetization behavior.

Testbed. We obtain our results in a dedicated testbed with three different hardware switches and three different SDN controller implementations. In these measurements, we use wiretaps as well as the Spirent C1 testing platform and traffic generator to ensure a reliable ground truth with high precision.

The remainder of this work is structured as follows. We discuss related work in Sect. 2. In Sect. 3, the possible communication schemes for exchanging OpenFlow FlowMod messages are presented alongside the testbed setup and the resulting measurement options. Measurement results are covered in Sect. 4 and 5 concludes the paper.

2 Related Work

This section covers two main areas of related work. On the one hand, approaches for evaluating the performance of different aspects and components of an SDN architecture are presented. On the other hand, an overview of mechanisms for identifying and addressing the heterogeneity of SDN switches is provided.

Techniques for testing SDN-based networks in a holistic fashion are discussed in [4]. Before addressing the long term goal of integrated tests, it is necessary to understand the behavior of the individual network elements, i.e., controllers and

switches. In an effort to provide means to test switch behavior with respect to compliance with the OpenFlow protocol specification, the authors of [5] present the OFTest suite. In contrast to this work, they focus on functional testing rather than performance tests.

The study conducted in [6] features a dedicated hardware traffic generator in order to test the data plane performance of Linux-based OpenFlow switching. In a similar setup, the authors of [7] investigate the characteristics of virtual switches and underlying virtualization techniques. In both works, the main interest lies in the data plane performance of the different switch implementations. This work, on the other hand, investigates the control plane performance of OpenFlow-enabled switches under varying network conditions.

OFLOPS [8] is a software framework for testing OpenFlow switch performance in the data plane as well as in the control plane. Its extension, OFLOPS-Turbo [9] is capable of 10 GbE traffic generation and utilizes the open-source NetFPGA-based OSNT [10] traffic generator and capture system. In contrast, we focus on the processing time of FlowMod messages in order to assess the effects of control plane delays on the resulting performance.

Analytical approaches like [11,12] investigate the influence of different network parameters on the performance of an OpenFlow architecture. Since such models are often based on measurements, the accuracy of these measurements also positively affects the quality of the resulting models. Therefore, one key aspect of our analyses is the accuracy of the available measurement mechanisms. A methodology for assessing the accuracy of measurements in the SDN context is presented in [13]. In addition to measurements performed by an SDN controller module, wiretaps installed at both ends of a communication channel serve as a means of providing the ground truth. This technique is also applied in the experiments that are conducted during the course of this work.

Several previous works highlight the heterogeneity of SDN switch hardware in terms of functionality, performance, and OpenFlow protocol compliance [14,15]. Unexpected or unreliable behavior such as additional delays and inconsistency between control and data plane pose several risks with respect to security as well as correct forwarding behavior. Hence, this heterogeneity needs to be taken into account for proper planning and design of real world deployments.

Some aspects of the heterogeneity, e.g., OpenFlow protocol compliance, are addressed by approaches such as TableVisor [16] and FlowConvertor [17] that introduce abstraction layers to translate given OpenFlow messages to device specific directives that take into account the behavior of individual switch hardware. While their focus is on maintaining functional homogeneity, we address the performance aspect. Finally, methods for data plane verification and consistency checks between data and control plane are proposed in [18]. However, rather than focusing on the identification of faulty switches, we are interested in performance prediction.

3 Methodology

In this section, two mechanisms for sending FlowMod messages from an SDN controller to OpenFlow switches are presented. Furthermore, we provide an overview of the experimental setup alongside measurement parameters and the configuration of the hardware that is used.

3.1 Mechanisms for Sending FlowMod Messages

There are two types of methods that are used for installing OpenFlow rules in a switch: asynchronous and synchronous or *addFlowAsync* and *addFlow*, respectively. In the first case, every FlowMod is followed by a BarrierRequest, and the next FlowMod is sent to the switch if and only if the corresponding BarrierReply has already been received and thus, the controller is informed that the previous FlowMod is successfully installed. On the other hand, the *addFlowAsync* mechanism generates a set of FlowMod messages and sends only one BarrierRequest afterwards. The difference between the two mechanisms is illustrated in Fig. 1, together with the measurement parameters that are considered in this work.

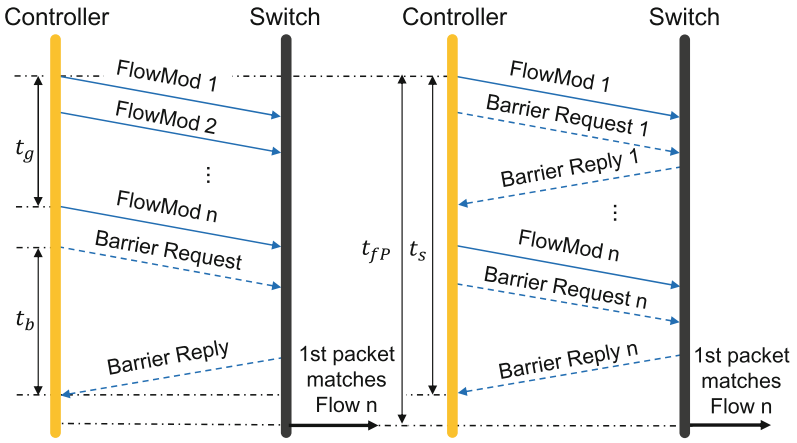


Fig. 1. Asynchronous and synchronous methods for adding flows to an Openflow switch.

On the left side of Fig. 1, t_g represents the time that the controller needs to generate n FlowMod messages in case of *addFlowAsync* and t_b is the duration between BarrierRequest and BarrierReply. The time between the first FlowMod and the last BarrierReply indicates how long it takes the switch to finish setting up n rules and is denoted as t_s in both cases. Finally, t_{fP} denotes the time difference between the first FlowMod message and the first data plane packet that is forwarded by the switch according to the last FlowMod it received. This verifies that the corresponding flow entry is actually installed in the data plane of the switch.

Note that $t_s > t_{fP}$ can occur in real systems due to two reasons. On the one hand, the BarrierReply message is subject to the delay between the switch and the controller. On the other hand, switches may already have installed flow rules in their data path prior to actually sending the corresponding BarrierReply. Both of these scenarios may lead to the BarrierReply arriving after the first packet is already forwarded in the data plane.

3.2 Testbed Setup

In order to investigate the impact of transmission delay on the estimation of FlowMod message processing times, experiments are performed in a testbed that is set up according to Fig. 2. In addition to a computer[1] which runs the SDN controller that is connected to an OpenFlow switch, two dedicated hosts[2] act as traffic source and traffic sink.

Furthermore, a computer with two 1 Gbps network interface cards (NICs) runs Ubuntu 16.04 and emulates the transmission delay in both directions, i.e., from the switch to the controller and vice versa. The red lines indicate links with delay, which is set via the tc command[3]. A Net Optics tap device[4] is inserted between the controller and Netem PC with the purpose of mirroring all traffic that passes through the corresponding link to the monitoring machine, an HP Proliant DL32 server. This server is equipped with an Endace DAG (Data Acquisition and Generation) 7.5G2 card, which has 2 Gigabit Ethernet ports, to capture every incoming packet.

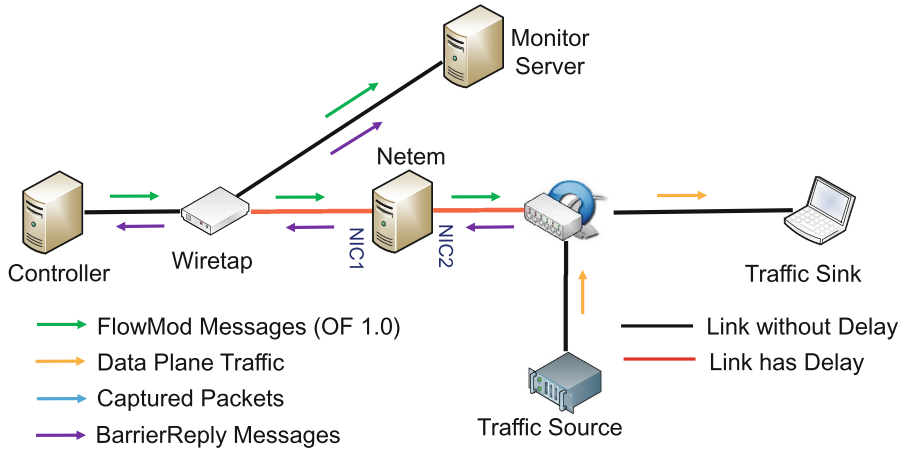


Fig. 2. Logical testbed setup.

[1] Intel(R) Core(TM) i7-2600 CPU @ 3.40 GHz/16 GB RAM.
[2] Intel(R) Core(TM)2 Duo CPU E8500/4G RAM.
[3] `sudo tc qdisc add dev [interface] root netem delay [delayValue]`.
[4] http://www.ixiacom.com/products/ixia-gig-zero-delay-tap/.

Three different controllers are utilized in this work. Firstly, the latest version of the Python-based Ryu controller[5] is used in conjunction with an additional module that allows the generation of FlowMod messages according to the two aforementioned methods. Secondly, a module with similar function is built for the Java-based OpenDaylight controller[6]. Finally, the Spirent C1 testing platform[7] with the OpenFlow Testing Package allows emulating an SDN controller. Furthermore, the Spirent C1 is capable of emulating the traffic source and sink, simplifying the testbed setup.

In this work, a total of three OpenFlow switches are used. Their specifications are displayed in Table 1. Previous work [14] has already demonstrated that in addition to pure hardware specifications, the fill level of the TCAM also has an impact on FlowMod processing times. During our measurements, we were able to confirm this behavior. However, we omit the detailed results since in this paper, we focus on the network delay between switch and controller.

Table 1. Switches used in this work.

Switch	CPU	Memory	Flow table size	Software
Pronto 3290	MPC8541 825 MHz	512 MB	3840	PicOs 2.0.14 (Open vSwitch v1.10.0)
Quanta T1048	MPC8541 825 MHz	1024 MB	2046	PicOs 2.6 (Open vSwitch v2.3.0)
NEC Pf5240	PowerPC 667 MHz	1024 MB	2816	OS-f3PA v5.0.0.1

3.3 Experiment Procedure

At the beginning of each measurement, the OpenFlow table in the switch is guaranteed to be empty. This is achieved by sending corresponding FlowMod messages to the switch before starting an experiment. Then, the controller sets up basic rules for exchanging ARP packets between the traffic source and sink. Later, these rules allow the traffic to be forwarded correctly to the destination without additional interaction with the controller. After that, another rule for dropping all packets that do not match any entry in the OpenFlow table of the switch is installed. Doing this prevents interference with the controller's performance due to an enormous number of PACKET_IN messages being forwarded to it. Meanwhile, the traffic source sends UDP traffic to the specific UDP port of

[5] http://osrg.github.io/ryu/, v4.18.

[6] https://www.opendaylight.org/software/downloads/hydrogen-base-10, Hydrogen release.

[7] http://www.spirent.com/Test-solutions_datasheets/Broadband/PAB/Spirent_TestCenter/STC_C1-Appliance_Datasheet, Spirent TestCenter Application v4.69.986.

the traffic sink using the Iperf[8] software. However, the packets can not reach the traffic sink due to the lack of a matching entry in the switch and are dropped. Afterwards, the controller generates either a batch of FlowMods followed by a BarrierRequest message or a series of alternating FlowMod and BarrierRequest messages. In both cases, the last FlowMod matches the aforementioned UDP traffic.

The results are collected by means of several approaches. Firstly, capture files are obtained by running a packet analyzer in the controller during the experiment, such as *tcpdump* or the Spirent C1's capture tool. The second approach relies on reports that are generated by the controller modules. Finally, the combination of wiretap devices and the DAG card offers the capability to capture and analyze packet timestamps at nanosecond precision.

4 Results and Discussion

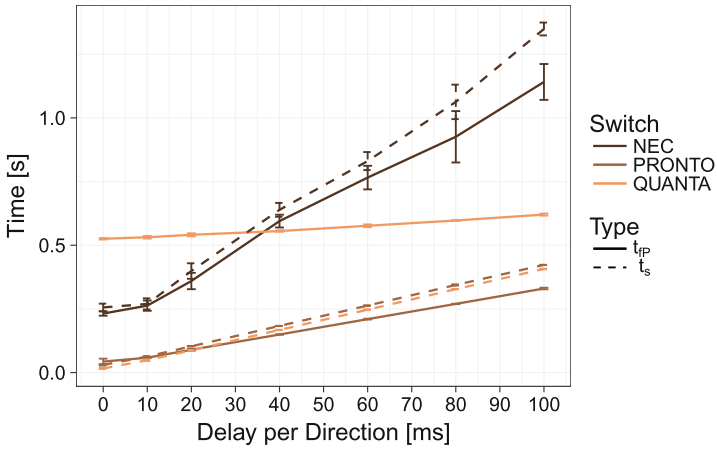
In this section, we present the results of the experiments that are described in Sect. 3. First, we demonstrate the heterogeneous of different hardware switches. This is achieved by comparing the FlowMod processing times of different switches when installing different numbers of flows and applying different amounts of control plane delay. Afterwards, we show that using prior information on the hardware specific characteristics and controller-based delay measurements, it is possible to achieve a high degree of correlation between the flow setup time, t_s, and the time until flow rules are active in the data plane, t_{fP}. This outcome highlights that reliable estimations of t_{fP} are possible at run time. Finally, we present results regarding the impact of the controller implementation on the FlowMod processing time.

Note that we omit results regarding the synchronous *addFlow* mechanism due to the fact that each flow rule is affected by the delay between switch and controller, resulting in non viable total delays even for small round trip times.

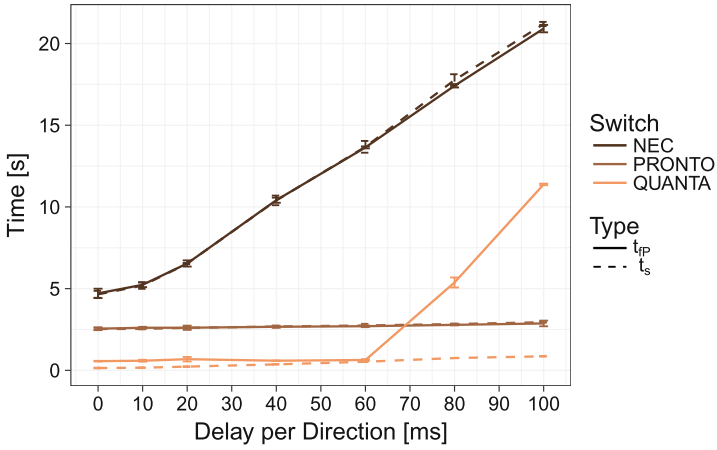
4.1 Impact of Switch Hardware

The two graphs of Fig. 3 highlight the individual behavior of the three switches that are used in this work with respect to their sensitivity to parameters such as the amount of control plane delay and the number of flows that are installed. Their x-axes represent the control plane delay that is set in each direction between switch and controller, i.e., a value of 10 ms corresponds to a round trip time of 20 ms. The y-axes denote the flow setup times t_s and t_{fP} that are recorded by means of the wiretap devices and are represented by dashed and solid lines, respectively. Finally, differently colored curves correspond to different switches. For each parameter combination, five experiment runs are performed in order to construct 95% confidence intervals that are indicated with error bars. The results in the figures are based on measurements with the OpenDaylight controller. Experiments with the other two controllers yield qualitatively similar results and are discussed in Sect. 4.2.

[8] https://iperf.fr/.



(a) 100 flows.



(b) 1800 flows.

Fig. 3. Influence of the control plane delay on the FlowMod processing time when using different switches and different numbers of flows. Scenario details: OpenDaylight controller and *addFlowAsync* mechanism.

Figure 3a displays results from experiments in which a total of 100 FlowMod messages are sent to the switch via the *addFlowAsync* mechanism, i.e., 100 FlowMods are followed by one pair of BarrierRequest and BarrierReply messages. Three observations can be made. First, the three switches operate at different time scales. With processing times that are lower than 500 ms for all delay values, the Pronto switch consistently outperforms the other two switches in this scenario. Second, the sensitivity of the switches towards the control plane delay varies significantly, as indicated by the different slopes of the individual

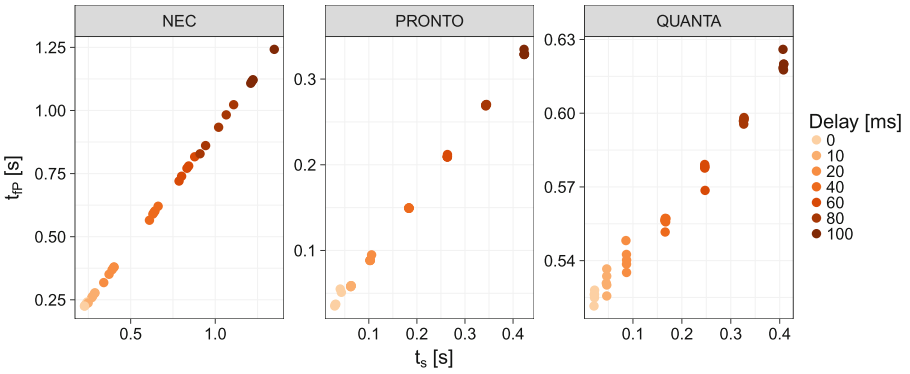
curves. Consequently, the NEC switch achieves lower values of t_{fP} than Quanta in scenarios with a low delay, whereas the Quanta switch is least affected by the increasing delay and gives better results for delays that are larger than 40 ms. Third, while the NEC and Pronto switch send their barrier reply after having installed all flow rules into the data plane, i.e., $t_s > t_{fP}$, the Quanta switch sends out the confirmation before the rules are active. Hence, a window of inconsistency of up to half a second can occur if the controller is unaware of this behavior.

Increasing the number of installed flows to 1800 exposes additional differences between the switches. The corresponding results are shown in Fig. 3b. For all switches, the increased number of FlowMod messages that need to be processed results in larger setup times. Furthermore, the high delay sensitivity of the NEC switch is even more pronounced in this scenario, with setup times ranging from 5 to over 20 s. In the case of the Quanta switch, a significant increase of the installation time is observed for delay values larger than 60 ms. Combined with the premature barrier reply message, this can be a major threat to state consistency. Only the Pronto switch is able to maintain nearly constant t_s and t_{fP} values for all delay settings. The differences in terms of performance between the three switches could stem in part from the heterogeneity of the hardware (cf. Table 1) as well as from vendor specific implementations of various features in the used firmware. Consequently, these differences in combination with internal queues of varying size inside the switches also affect the sending behavior of the controller which adheres to TCP. Therefore, the corresponding rate control mechanisms can lead to lower transmission rates, which ultimately leads to higher flow setup times.

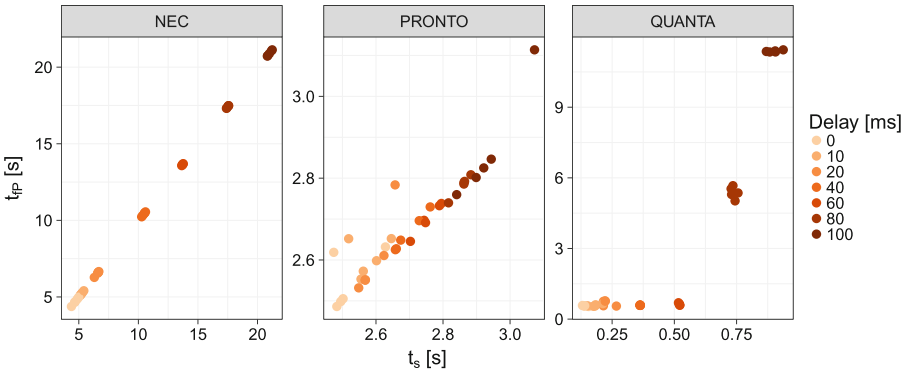
While the wiretap-based measurements that are presented in the previous figures demonstrate the differences between the hardware switches, there is a high pairwise similarity between t_s and t_{fP} values. We use this relationship to derive a mechanism that can be used to infer t_{fP} from information regarding the particular switch model that is in use and measurements at the controller. These measurements include tcpdump on the controller machine to obtain t_s and a simple round trip time measurement like ping to determine the control plane delay.

For each of the three switches, the graphs in Fig. 4 show the measurement of the flow setup time t_s at the controller on the x-axis and the actual time until the first data plane packet t_{fP} at the wiretap on the y-axis. Differently colored dots denote different control plane delays.

In Fig. 4a, results that are obtained when installing 100 flow rules are displayed. Although the times that are recorded for the three switches have significantly different ranges, a high linear correlation between t_s and t_{fP} can be observed. Hence, using switch-specific information regarding its sensitivity towards control plane delay in conjunction with round trip time and t_s measurements is sufficient for an accurate estimation of the flow installation time in the data plane.



(a) 100 flows.

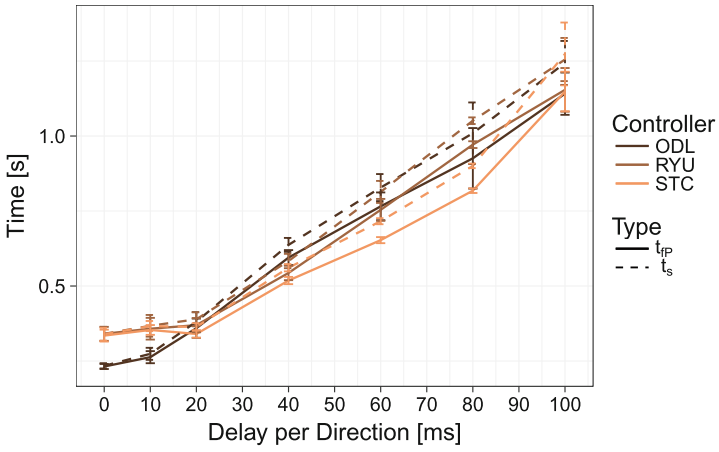


(b) 1800 flows.

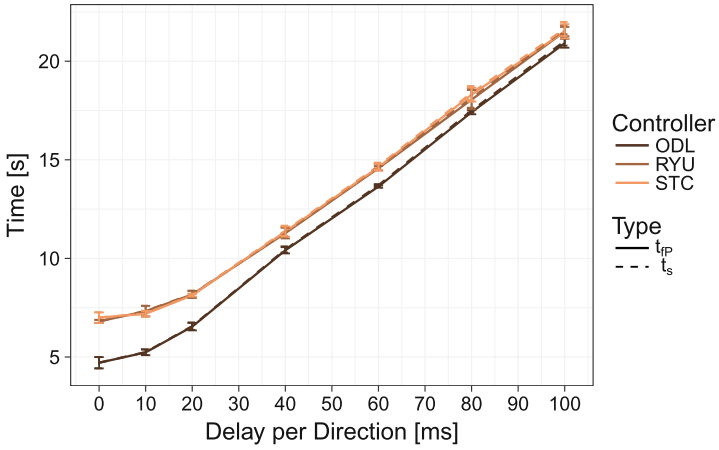
Fig. 4. Flow setup time t_s recorded at the controller and time to first data plane packet t_{fP} recorded via the wiretap for the three different switches. Scenario details: OpenDaylight controller and *addFlowAsync* mechanism.

Similar results are obtained in case of the installation of 1800 flow rules, as presented in Fig. 4b. While the NEC switch has the highest setup and processing times, it also has the most consistent behavior and an almost perfect linear correlation. Except for few outliers, the Pronto switch also shows a high degree of correlation, even with the increased number of flows. Finally, the Quanta switch produces outliers for high control plane delays. Nevertheless, this behavior is observed consistently - qualitatively as well as quantitatively - in multiple repetitions of our experiments, as indicated by clusters of dots in the scatter plot. Therefore, this switch-specific characteristic can also be taken into account by the controller when making predictions regarding the data plane state.

Summarizing, our findings show that using simple controller-based measurements in combination with switch properties that can be determined prior to deployment can be used for performing accurate prediction of the FlowMod installation time in the data plane of OpenFlow switches.



(a) 100 flows.



(b) 1800 flows.

Fig. 5. Impact of the controller choice on flow setup times for different numbers of installed flows. Configuration details: NEC switch and *addFlowAsync* mechanism.

4.2 Impact of Controller Choice

While the previously shown results focus on the peculiarities of different data plane hardware, this section is devoted to the influence of the controller implementation on the performance. To this end, experiments with the NEC switch are conducted with three different controllers. The NEC switch has been selected due to the stability of observed setup times in the results above. The three controllers include the Java-based OpenDaylight controller, the Python-based Ryu controller, as well as the controller implementation that is provided by the

OpenFlow Testing Package of the Spirent C1. In case of the OpenDaylight and Ryu controller, the same host machine is used to ensure that the results are not affected by a heterogeneity of the underlying hardware. The graphics in Fig. 5 show t_s and t_{fP} values for different numbers of flows on the y-axis and the control plane delay on the x-axis. Differently colored lines correspond to the three controllers.

When 100 flows are installed, the majority of confidence intervals in Fig. 5a overlap. This indicates that for control plane delays that are larger than 10 ms, no statistically significant difference between the three controllers can be identified. In the case of control plane delays that are lower than 20 ms, using the OpenDaylight controller leads to setup times of roughly 0.23 s as opposed to setup times of roughly 0.34 s that are observed for Ryu and the Spirent-based controller.

These phenomena are even more pronounced in the case of 1800 flows. Figure 5b shows that using the OpenDaylight controller leads to consistently faster flow setup times than Ryu and Spirent. Differences between 1 and 2 s are observed for setup times that range between 4.7 and 21.6 s.

The aforementioned results indicate that the controller is not merely a generator of FlowMod messages but can also affect the performance. In-depth analyses of the corresponding packet dumps show that the sending behavior of the OpenDaylight controller and the corresponding packetization of OpenFlow messages differs from the other two controllers. Hence, controller developers should be aware of such mechanisms in order to adapt to switch capabilities and opportunities to improve the overall performance.

5 Conclusion

In this work, we investigate the influence of control plane delay on the performance of OpenFlow switches in terms of their FlowMod processing time. Our testbed setup features hardware from NEC, Quanta, and Pronto as well as three different SDN controller implementations. These include the Java-based OpenDaylight controller, the Python-based Ryu controller, and the controller implementation that is available in the OpenFlow Testing Package of the Spirent C1 platform. Additionally, we use wiretap devices in order to obtain highly precise measurements.

The contribution of this work is threefold. Firstly, we confirm the heterogeneity of OpenFlow switching hardware. This includes not only varying processing times but also different degrees of sensitivity towards the control plane delay between controller and switch. In particular, the latter is not only affected by common hardware specifications such as the CPU rate and amount of RAM, but also by details like queue sizes that can lead to TCP rate control phenomena. Secondly, we demonstrate that switch-specific characteristics that can be extracted prior to deployment can be used in conjunction with simple measurements at the controller in order to accurately predict the data plane state and performance of switches. Such a prediction mechanism can significantly reduce

the window of inconsistency between an SDN controller and the switches it manages. Finally, we show that implementation details of the SDN controller can also have an impact on the overall FlowMod processing performance due to sender-side behavior. This leads to optimization potential that can be taken into account by both, controller developers who want to improve the general performance of their controller as well as network operators who want to maximize compatibility and reliability of the components in their particular network.

Several directions for future work are available. On the one hand, the impact of dynamically fluctuating control plane delays can be analyzed. Depending on the amount and frequency of the fluctuation, the measurement frequency at the controller needs to be adapted. On the other hand, more in-depth analyses of the packet dumps might reveal different classes of switch-side behavior that can be used to infer more generic and robust models.

Acknowledgments. This work has been performed in the framework of the CELTIC EUREKA project SENDATE-PLANETS (Project ID C2015/3-1), and it is partly funded by the German BMBF (Project ID 16KIS0474). The authors alone are responsible for the content of the paper.

References

1. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks. SIGCOMM CCR **38**, 69–74 (2008)
2. Jarschel, M., Zinner, T., Hossfeld, T., Tran-Gia, P., Kellerer, W.: Interfaces, attributes, and use cases: a compass for SDN. IEEE Commun. Mag. **52**, 210–217 (2014)
3. Nguyen-Ngoc, A., Lange, S., Gebert, S., Zinner, T., Tran-Gia, P., Jarschel, M.: Performance evaluation mechanisms for flowmod message processing in openflow switches. In: 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE) (2016)
4. Kuźniar, M., Canini, M., Kostić, D.: OFTEN testing OpenFlow networks. In: European Workshop on Software Defined Networking (EWSDN) (2012)
5. OFTest –Validating OpenFlow Switches, Big Switch Networks. http://www.projectoodlight.org/oftest/
6. Bianco, A., Birke, R., Giraudo, L., Palacin, M.: Openflow switching: data plane performance. In: IEEE International Conference on Communications (ICC) (2010)
7. Emmerich, P., Raumer, D., Wohlfart, F., Carle, G.: Performance characteristics of virtual switching. In: IEEE 3rd International Conference on Cloud Networking (CloudNet) (2014)
8. Rotsos, C., Sarrar, N., Uhlig, S., Sherwood, R., Moore, A.W.: OFLOPS: an open framework for OpenFlow switch evaluation. In: Taft, N., Ricciato, F. (eds.) PAM 2012. LNCS, vol. 7192, pp. 85–95. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28537-0_9
9. Rotsos, C., Antichi, G., Bruyere, M., Owezarski, P., Moore, A.: OFLOPS-Turbo: testing the next-generation OpenFlow switch. In: European Workshop on Software Defined Networks (EWSDN) (2014)

10. Shahbaz, M., et al.: Architecture for an open source network tester. In: Proceedings of the Ninth ACM/IEEE Symposium on Architectures for Networking and Communications Systems (2013)
11. Jarschel, M., Oechsner, S., Schlosser, D., Pries, R., Goll, S., Tran-Gia, P.: Modeling and performance evaluation of an OpenFlow architecture. In: Proceedings of the 23rd International Teletraffic Congress (2011)
12. Azodolmolky, S., Nejabati, R., Pazouki, M., Wieder, P., Yahyapour, R., Simeonidou, D.: An analytical model for software defined networking: a network calculus-based approach. In: IEEE Global Communications Conference (GLOBECOM) (2013)
13. Jarschel, M., Zinner, T., Höhn, T., Tran-Gia, P.: On the accuracy of leveraging SDN for passive network measurements. In: Australasian Telecommunication Networks & Applications Conference (ATNAC) (2013)
14. Kuźniar, M., Perešíni, P., Kostić, D.: What you need to know about SDN flow tables. In: Mirkovic, J., Liu, Y. (eds.) PAM 2015. LNCS, vol. 8995, pp. 347–359. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15509-8_26
15. Huang, D.Y., Yocum, K., Snoeren, A.C.: High-fidelity switch models for software-defined network emulation. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (2013)
16. Geissler, S., Herrnleben, S., Bauer, R., Gebert, S., Zinner, T., Jarschel, M.: Tablevisor 2.0: Towards full-featured, scalable and hardware-independent multi table processing. In: 2017 IEEE Conference on Network Softwarization (NetSoft) (2017)
17. Pan, H., Xie, G., Li, Z., He, P., Mathy, L.: FlowConvertor: enabling portability of SDN applications. In: IEEE Conference on Computer Communications-INFOCOM 2017. IEEE (2017)
18. Zhang, P., Li, H., Hu, C., Hu, L., Xiong, L., Wang, R., Zhang, Y.: Mind the gap: monitoring the control-data plane consistency in software defined networks. In: Proceedings of the 12th International on Conference on emerging Networking Experiments and Technologies (2016)

Dynamic Control of Running Servers

Esa Hyytiä[1,2(✉)], Douglas Down[3], Pasi Lassila[2], and Samuli Aalto[2]

[1] Department of Computer Science,
University of Iceland, Reykjavik, Iceland
esa@hi.is
[2] Department of Communications and Networking,
Aalto University, Espoo, Finland
[3] Department of Computing and Software,
McMaster University, Hamilton, Canada

Abstract. Motivated by a data center setting, we study the problem of joint dispatching and server sleep state control in a system consisting of two queues in parallel. Using the theory of Markov decision processes and a novel lookahead approach, we explicitly determine near-optimal control policies that minimize a combination of QoE costs, energy costs, and wear and tear costs due to switching. Guidelines are provided as to when these combined policies are most effective.

Keywords: Data centers · Queueing systems · Dynamic control
Markov decision processes · Lookahead techniques

1 Introduction

Server clusters comprise the core of modern data centers and cloud computing systems. Stochastic queueing models, such as multiserver systems with a central queue or distributed systems consisting of multiple parallel servers with their own queues, are suitable for the performance analysis of such systems [11]. Traditional mechanisms for their control include job scheduling and dispatching (a.k.a. task assignment). In a distributed system, the dispatcher decides to which server an arriving job is routed, and the local scheduler decides on how the service capacity of the server is dynamically shared among its jobs.

When optimizing the control of such queueing systems, an important measure is the response time, i.e., the total delay of a job. Typical objectives related to the delay performance are minimization of the mean response time or its tail probability. However, in current computing systems, a significant additional factor is energy efficiency [1]. It is not enough to optimize the delay performance, but one should also take into account the energy aspect. Both dispatching and scheduling decisions affect not only the delay performance but also the energy efficiency. An additional dimension in this joint control problem is related to the sleep states of servers. One should decide on when to put a server into a sleep

This work was supported by the Academy of Finland in the FQ4BD and TOP-Energy projects (grant nos. 296206 and 268992).

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 127–141, 2018.
https://doi.org/10.1007/978-3-319-74947-1_9

mode and when to wake it up again. While sleeping tends to reduces energy costs, it has a negative effect on the delay performance, since, after the wake-up, there is typically a relatively long setup time before the server is back in full operation. In addition, the more often a server is switched on and off, the more vulnerable to failures it becomes.

In this paper, we consider distributed systems consisting of multiple parallel servers with their own queues. The servers are assumed to apply the First Come First Served (FCFS) scheduling policy. Our goal is to develop near-optimal joint control policies for dispatching and sleep state control of servers when the trade-off between delay performance and energy efficiency is described by a composite objective function. Our approach is based on the theory of Markov decision processes (MDPs) [25]. More precisely said, we apply the *policy improvement* method and combine it to a *lookahead* technique [13]. We demonstrate that the combination of these control mechanisms can yield significant savings, in particular when the system is under moderate load.

2 Related Work

While queueing theory has been applied for decades to evaluate the performance of computing systems, Chen et al. [2] and Sledgers et al. [26] were the first to use queueing models to study the problem of energy-aware control of server clusters. Since then, energy-aware multiserver systems with a central queue have been analyzed in many papers [4–6,18–20,23,24]. However, the optimal control problem in such energy-aware multiserver systems has proved difficult. Exact solutions have been found for the single server case [5,8–10,16], but with multiple servers only structural properties of the optimal policy are obtained [17].

For static dispatching policies (such as Random Routing), the analysis of energy-aware distributed systems consisting of multiple parallel servers with their own queues is straightforward, since the parallel queues can be analyzed separately. However, dynamic policies (such as Join the Shortest Queue) are mathematically tractable only under restrictive assumptions. Moreover, exact optimality results are scarce, being available only for some specific setups. Near-optimal solutions for dispatching problems have been developed by applying the policy iteration approach from the theory of Markov decision processes [12,25,28]. Such an approach has been utilized for composite objective functions that take into account the performance-energy trade-off [14,15,22].

With respect to sleep state control in a multiserver setting, it has been observed in [5,7] that putting servers to sleep aggressively can be harmful. If a server is turned off e.g., immediately when it becomes idle, energy costs may be saved in the short term, but significant response time degradation may result, in particular when setup times are significant. They suggest that an idle server waits a period of time before being put to sleep. They choose a state-independent timer for this wait and design a dispatching policy that takes this sleep state control mechanism into account. In [21], it is shown that dispatching and sleep state control of this form is optimal (simultaneously minimizes mean response time

and energy costs) in a many-servers asymptotic regime. In contrast, we allow the sleep state control to be state dependent, considering the joint control problem in an MDP framework. Consistent with [5,7,21], we see that gains can be made by conscious turn off decisions, but for the small system that we consider, we further identify system parameters under which such gains are significant.

3 Joint Control Problem

Consider a distributed computing system consisting of two parallel FCFS servers with their own queues. The servers are assumed to be homogeneous, i.e., they have the same service rate. Service times of jobs, denoted by X, are assumed to be independent and generally distributed with finite first two moments. Jobs arrive according to a Poisson process with rate Λ. The load per server is denoted by $\rho = \Lambda \mathbb{E}[X]/2$. For stability, we assume that $\rho < 1$. Each job is dispatched to one of the two servers upon its arrival. The dispatching decisions are assumed to be *dynamically controllable*, i.e., they may depend on the state of the system.

The servers are assumed to be *energy-aware*, and there are four different *operational states* for the servers: (i) *busy*, (ii) *idle*, (iii) *off*, and (iv) *set up*. A server is busy when it is processing jobs. As soon as the service of all available jobs is completed, the server becomes idle. The server remains idle as long as one of the following events takes place. Either a new job is dispatched to it, in which case the server becomes again busy and starts serving the new job, or the *switch-off timer* (associated with the idle server) expires, in which case the server is immediately switched off. The length of the switch-off timer, denoted by τ, is assumed to be dynamically controllable. If the switch-off timer expires, the server remains off until a new job is dispatched to it, at which time the server is switched on (set up). After a *setup time*, the server becomes again busy and starts serving the jobs waiting in its queue. Setup times of servers, denoted by D, are assumed to be independent and generally distributed with finite first two moments. In particular, we are, however, interested in the case where the setup times are deterministic, $D = d$. When busy or set up, the power consumption of a server is e [watts], but when idle, its power consumption is ϵ [watts], which is assumed to be less than e, i.e., $\epsilon = \gamma e$, where $\gamma < 1$. In the sequel, we will use e as our power unit. When off, a server does not consume any power. In line with [5], such an energy-aware server is called *DelayedOff*. Special cases are *NeverOff* ($\tau \to \infty$) and *InstantOff* ($\tau = 0$).

The cost structure comprises both QoE and system specific cost components including both energy and switching costs. The QoE metric in our model is the mean response time $\mathbb{E}[T]$. Note that, due to the well-known Little's formula, minimizing the mean response time is equivalent to minimizing the mean total number of jobs in the system, $\mathbb{E}[N] = \Lambda \mathbb{E}[T]$. Energy costs are related to the mean total power consumption $\mathbb{E}[P]$, and switching costs take into account wear and tear costs of switching a server off and on. More precisely, we assume that the *mean cost rate* of the whole system is given by

$$r = r_T + r_P + r_S = \mathbb{E}[N] \cdot c_T + \mathbb{E}[P] \cdot c_P + \Lambda_S \cdot c_S, \tag{1}$$

where Λ_S denotes the aggregate switch-on (and off) rate of the two servers. The constants (c_T, c_P, c_S) map each component to a common unit.

Now the problem is to find a *joint dispatching and sleep state control* that minimizes the mean cost rate (1). Dispatching decisions are made when new jobs arrive, and the sleep states are controlled when a server becomes idle or a new job arrives. We allow dynamic control that is based on the current state of the system, together with the service time of the arriving job if we are about to make a dispatching decision. We assume that the state of a server is described by its *virtual backlog*, switch-off timer value, and energy state. The virtual backlog u refers to the time needed to complete the service of all jobs currently in the system (without any new arrivals). If the server is busy, the virtual backlog is just the ordinary backlog, i.e., the sum of remaining service times, but if the server is in set up, it also includes the remaining setup time. For an off or idle server, the virtual backlog equals 0. The current value of the switch-off timer, t, refers to the time that the server has been (continuously) idle, $0 \le t \le \tau$. In the following section, we tackle this optimal control problem by the policy improvement method combined with the lookahead technique.

4 Policy Improvement and Lookahead

For the policy improvement method, we need a *basic control policy* that can be analyzed *explicitly*. Such a policy is attained if we apply random routing to dispatching and deterministic switch-off timers for the sleep state control. In this paper, we choose uniform routing probabilities (1/2 for each server) so that the load is balanced, which is a reasonable basic dispatching policy. As a result, there are two independent single-server queues with Poisson arrivals at rate $\lambda = \Lambda/2$. We need to derive (for each queue i) the so-called *relative value function* $v_i(u_i, t_i) - v_i(0, 0)$, which gives the difference in the mean accumulated costs if the system starts from states (u_i, t_i) and $(0, 0)$, respectively, where u_i refers to the virtual backlog and t_i the switch-off timer value of server i. Formally, the *value function* is defined as

$$v_i(u_i, t_i) := \lim_{t \to \infty} \mathbb{E}\left[C_i(u_i, t_i, t) - r_i t\right], \tag{2}$$

where $C_i(u_i, t_i, t)$ denotes the costs queue u incurs during time $(0, t)$ when initially in state (u_i, t_i), and r_i is the mean cost rate of queue i. The relative value function for the DelayedOff M/G/1-FCFS queue is derived in Sect. 5. In addition, we assume that the sleep state control of the basic policy is such that server 1 is an ordinary NeverOff server ($\tau_1 \to \infty$) and server 2 is an energy-aware InstantOff server ($\tau_2 = 0$), which is a reasonable compromise for all traffic load situations.

Below we show how to improve this static (i.e., state-independent) basic policy by developing a dynamic control policy that utilizes the state information. We start from the dispatching decisions. For the sleep state control, we consider separately two different cases: first the case when a server becomes idle, and thereafter the case when a server is already off and a new job arrives. For simplicity, the results in this section are given for deterministic setup times d.

4.1 Improving Dispatching Decisions

Recall first that the basic policy assumes that server 1 is NeverOff ($\tau_1 \to \infty$) and server 2 InstantOff ($\tau_2 = 0$). Thus, the state of server 1 is completely described by the virtual backlog u_1 (which, in this case, is the same as the ordinary backlog). From Proposition 1 (presented and justified in Sect. 5), we get its relative value function:

$$v_1(u_1) - v_1(0) = \frac{\lambda\, u_1^2}{2(1-\rho)}\, c_T + u_1(1-\gamma)c_P. \tag{3}$$

On the other hand, to describe the state of server 2, it is enough to specify the virtual backlog u_2 and indicate whether the server is switched off (s) or running (r), i.e., in set up or busy. From Proposition 1, we again get the corresponding relative value function:

$$v_2^{(r)}(u_2) - v_2^{(s)}(0) = \frac{\lambda\, u_2^2}{2(1-\rho)}\, c_T + \frac{u_2}{1+\lambda d}\left[c_P - \lambda c_S - \frac{\lambda d(2+\lambda d)}{2(1-\rho)}\, c_T\right]. \tag{4}$$

The dispatching decisions of the static basic policy can be improved by choosing the server i for which the expected *admission costs* $a_i(u_i, x)$ are minimized, where x denotes the service time of the arriving job. The expected admission costs can be calculated as follows. For server 1, we have

$$\begin{aligned} a_1(u_1, x) &= (u_1 + x)c_T + v_1(u_1 + x) - v_1(u_1) \\ &= \left(u_1 + x + \frac{\lambda x(2u_1 + x)}{2(1-\rho)}\right) c_T + x(1-\gamma)c_P, \end{aligned}$$

and, for server 2, we have

$$\begin{aligned} a_2^{(r)}(u_2, x) &= (u_2 + x)c_T + v_2^{(r)}(u_2 + x) - v_2^{(r)}(u_2) \\ &= \left(u_2 + x + \frac{\lambda x(2u_2 + x)}{2(1-\rho)}\right) c_T + \frac{x}{1+\lambda d}\left(c_P - \lambda c_S - \frac{\lambda d(2+\lambda d)}{2(1-\rho)} c_T\right), \end{aligned}$$

and $a_2^{(s)}(0, x) = c_S + a_2^{(r)}(0, d + x)$, obviously, as one switching cost is saved. In each case, it is easy to identify the immediate cost consisting of the response time of the new job and the possible switching cost c_S.

4.2 Lookahead for Server Switch-Off

The static decision to switch server 2 off whenever it becomes idle is obviously suboptimal. Next we apply the lookahead technique to tackle this [13].

Suppose that server 1 has backlog u_1 when server 2 becomes idle. By default, we would switch server 2 off at this point. However, we can consider the following two alternative actions:

A: Switch server 2 off immediately and route the next job, given it arrives before time τ, to server 1.

B: Keep server 2 running idle for time τ hoping that a new job arrives soon, which would then be routed to server 2.

In both cases, after time τ, we return back to the default routing and switch-off policies. In general, τ is a free parameter less than u_1.

With these, one can compute *in closed-form* the expected cost of the alternative actions, denoted by d_A and d_B, respectively,

$$\begin{aligned} d_A &= \int_0^\tau \Lambda e^{-\Lambda t}\Big[(c_P - r)t + (u_1 - t + \mathbb{E}\,[X])c_T + \mathbb{E}\,[v_1(u_1 - t + X)] + v_2^{(s)}(0)\Big]\,dt \\ &\quad + \; e^{-\Lambda\tau}((c_P - r)\tau + v_1(u_1 - \tau) + v_2^{(s)}(0)), \\ d_B &= \int_0^\tau \Lambda e^{-\Lambda t}\Big[((1+\gamma)c_P - r)t + \mathbb{E}\,[X]\; c_T + v_1(u_1 - t) + \mathbb{E}\left[v_2^{(r)}(X)\right]\Big]\,dt \\ &\quad + \; e^{-\Lambda\tau}(((1+\gamma)c_P - r)\tau + v_1(u_1 - \tau) + v_2^{(s)}(0)). \end{aligned}$$

Then we choose to keep server 2 idle if that action yields a smaller expected cost, $d_A - d_B > 0$.

As time passes without an arrival, u_1 gets smaller, and the benefits from keeping server 2 running idle become smaller. This suggests that we can consider a differential time step. In particular, we find that

$$\begin{aligned} f(u_1) &:= \lim_{\tau\to 0} \frac{d_B - d_A}{\tau} \\ &= \frac{\lambda}{1-\rho}\left(\frac{(\rho d(2+\lambda d))}{1+\lambda d} + 2u_1\right)c_T + \frac{2\lambda\rho(d\,c_P + c_S)}{1+\lambda d} - (1+2\rho)\gamma\,c_P, \end{aligned}$$

and then solving $f(u_1) = 0$ yields the critical backlog above which server 2 can be kept idle instead of being switched off,

$$u_1^* = \frac{1+\rho-2\rho^2}{2\lambda c_T}\gamma\,c_P - \frac{\rho(2(1-\rho)(d\,c_P + c_S) + d(2+\lambda d)c_T)}{2c_T(1+\lambda d)}.$$

Note that u_1^* depends only on the first moment of the service time distribution. Moreover, the second term is always negative, and therefore if $\epsilon = 0$, i.e., $\gamma = 0$, then $u_1^* < 0$, which suggests that it is preferable to keep server 2 on, which of course makes sense if idling incurs no energy costs.

Alternatively, one can determine the critical energy cost denoted by c_P^* above which server 2 should be switched off.

$$c_P^* = \frac{\lambda}{\gamma(1+2\rho)(1+\lambda d) - 2\rho\lambda d}\left(2\rho c_S + \frac{2u_1(1+\lambda d) + d\rho(2+\lambda d)}{1-\rho}c_T\right),$$

and in the special case of $\epsilon = e$, i.e., $\gamma = 1$, we have

$$c_P^* = \frac{\lambda}{1+\lambda d+2\rho}\left(2\rho c_S + \frac{2u_1(1+\lambda d) + d\rho(2+\lambda d)}{1-\rho}c_T\right).$$

Numerical Example. Let us next assume unit service time, $\mathbb{E}[X] = 1$, unit response time cost, $c_T = 1$, unit setup delay, $d = 1$, and no switching costs, $c_S = 0$. Moreover, the energy cost in busy/setup states is $c_P = 1$, and in the idle state γc_P, where $\gamma \in \{0.5, 1\}$. Server 1 has backlog u_1, and server 2 becomes empty. Then we vary the offered load ρ and evaluate when one should keep server 2 running. The results are depicted in Fig. 1(a). We can see that as the load increases, the critical backlog decreases eventually becoming zero, i.e., if the system is heavily loaded, then the response time costs starts to dominate (cf. the knee in the response time curve).

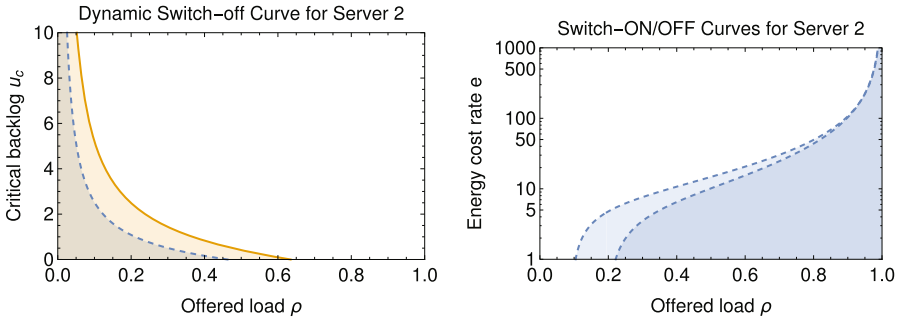


Fig. 1. Dynamic switch-off policies when $c_P = 1$, $c_T = 1$, $c_S = 0$ and $\gamma \in \{0.5, 1\}$ (left). Dynamic switch-on policy resulting from the lookahead analysis (right).

4.3 Proactive Switch-On of Servers

A similar lookahead analysis can be performed for a system in state $(u_1, 0)$, where server 2 has been switched off. As u increases, at some point it may be beneficial to switch server 2 back on, as the next job most likely ends up there. Perhaps the most elementary lookahead action to consider in this case is the action that switches server 2 on and routes the next job there unconditionally. This is a simple decision, and, e.g., server 1 may empty meanwhile and we keep then both servers running idle until the next job arrives. Carrying out a similar analysis and solving for the critical energy cost rate, we get

$$c_P^* = \frac{\lambda\left(e^{-2\lambda d} - (2-\rho)\right)}{(\gamma - (1-\gamma)\lambda d)\left(e^{-2\lambda d} + \rho\right) + 2\lambda d} c_S + \frac{(1+\lambda d)\left(\lambda d\left(e^{-2\lambda d} + \rho\right) + e^{-2\lambda u_1} - 1 + 2\lambda u_1\right) - \lambda d\,(1-\rho) + 1 - e^{-2\lambda d}}{2(1-\rho)\left((\gamma - (1-\gamma)\lambda d)\left(e^{-2\lambda d} + \rho\right) + 2\lambda d\right)} c_T.$$

Numerical Example. Figure 1(b) shows the keep running and proactive switching on decisions for with $\mathbb{E}[X] = 1$, $\epsilon = 1$, $c_S = 0$, $d = 1$, $c_T = 1$ and $u_1 = 5$.

5 Value Function for an Energy-Aware Single Server Queue

In this section, we consider a generic DelayedOff M/G/1-FCFS queue with arrival rate λ, generally distributed service times X, deterministic switch-off timer τ, and generally distributed setup times D. We assume a stable system, i.e., $\rho = \lambda \mathbb{E}[X] < 1$.

Our purpose is to derive the relative value function $v(u,t) - v(0,0)$, where u refers to the current virtual backlog and t the current value of the switch-off timer, $0 \le t \le \tau$. Thus, either u, t, or both are zero. The reference state is the renewal point when the server becomes idle and the switch-off timer starts to count towards τ.

We start by giving the mean number of jobs $\mathbb{E}[N]$, the mean power consumption $\mathbb{E}[P]$, and the mean switch-off rate λ_S for this DelayedOff M/G/1-FCFS queue, which are derived, e.g., in [8]:

$$\mathbb{E}[N] = \rho + \frac{\lambda^2 \mathbb{E}\left[X^2\right]}{2(1-\rho)} + \frac{\lambda(2\,\mathbb{E}[D] + \lambda\,\mathbb{E}\left[D^2\right])}{2\left(\lambda\,\mathbb{E}[D] + e^{\lambda\tau}\right)}, \tag{5}$$

$$\mathbb{E}[P] = \frac{\left(\lambda\,\mathbb{E}[D] + \rho e^{\lambda\tau}\right) + \gamma(1-\rho)\left(e^{\lambda\tau} - 1\right)}{\lambda\,\mathbb{E}[D] + e^{\lambda\tau}}, \tag{6}$$

$$\lambda_S = \frac{\lambda(1-\rho)}{\lambda\,\mathbb{E}[D] + e^{\lambda\tau}}. \tag{7}$$

Similarly as for the whole system in (1), the mean cost rate for the single server queue consists of three terms,

$$r = r_T + r_P + r_S = \mathbb{E}[N] \cdot c_T + \mathbb{E}[P] \cdot c_P + \lambda_S \cdot c_S. \tag{8}$$

The value function $v(u,t)$ is also a composite function consisting of three corresponding terms,

$$v(u,t) = v_T(u,t) + v_P(u,t) + v_S(u,t). \tag{9}$$

Proposition 1. *For a DelayedOff M/G/1-FCFS queue, the components of the relative value function $v(u,t) - v(0,0)$ are as follows:*

$$v_T(u,t) - v_T(0,0) = \frac{1}{2(1-\rho)}\left(\lambda u^2 - \frac{(2\,\mathbb{E}[D] + \lambda\,\mathbb{E}\left[D^2\right])\left(\lambda u + 1 - e^{\lambda t}\right)}{\lambda\,\mathbb{E}[D] + e^{\lambda\tau}}\right) c_T,$$

$$v_P(u,t) - v_P(0,0) = \frac{\left((1-\gamma)e^{\lambda\tau} + \gamma\right)\lambda u - (\gamma - \lambda\,\mathbb{E}[D]\,(1-\gamma))\left(e^{\lambda t} - 1\right)}{\lambda\left(\lambda\,\mathbb{E}[D] + e^{\lambda\tau}\right)} c_P,$$

$$v_S(u,t) - v_S(0,0) = -\frac{\lambda u + 1 - e^{\lambda t}}{\lambda\,\mathbb{E}[D] + e^{\lambda\tau}} c_S.$$

Proof. 1° Let us start with the response time related costs. By (5) and (8), we get

$$r_T = \mathbb{E}[N] \cdot c_T = \left(\rho + \frac{\lambda^2 \mathbb{E}\left[X^2\right]}{2(1-\rho)} + \frac{\lambda(2\,\mathbb{E}[D] + \lambda\,\mathbb{E}\left[D^2\right])}{2\left(\lambda\,\mathbb{E}[D] + e^{\lambda\tau}\right)}\right) c_T. \tag{10}$$

1.1° Assume first $u > 0$ and $t = 0$ so that the server is busy or in set up. Let B_u denote the length of the resulting "busy period", i.e., the time needed to decrease the virtual backlog from u to 0. In addition, let N_u denote the total number of jobs that arrived during that time, and $\mathbb{E}\left[T_1 + \ldots + T_{N_u}\right]$ the sum of their expected response times. By considering a separate M/G/1 queue with arrival rate λ where the service time of the first customer of each busy period equals u but for the other customers the service time follows the distribution of X, we get easily (see, e.g., [27])

$$\mathbb{E}\left[B_u\right] = \frac{u}{1-\rho}, \tag{11}$$

$$\mathbb{E}\left[T_1 + \ldots + T_{N_u}\right] = \frac{1}{2(1-\rho)}\left(\lambda u^2 + 2\rho u + \frac{\lambda^2 \mathbb{E}\left[X^2\right] u}{2(1-\rho)}\right).$$

Now, for the value function at state $(u, 0)$, from (2), we have

$$v_T(u,0) = \mathbb{E}\left[T_1 + \ldots + T_{N_u}\right] c_T - \mathbb{E}\left[B_u\right] r_T + v_T(0,0),$$

which implies, by (10) and the previous expressions, that

$$v_T(u,0) - v_T(0,0) = \frac{1}{2(1-\rho)}\left(\lambda u^2 - \frac{(2\,\mathbb{E}\left[D\right] + \lambda\,\mathbb{E}\left[D^2\right])\lambda u}{\lambda\,\mathbb{E}\left[D\right] + e^{\lambda\tau}}\right) c_T.$$

1.2° Assume now that $u = 0$ and $0 \le t \le \tau$ so that the server is idle or off. For the value function at state $(0, 0)$, we clearly have

$$\begin{aligned} v_T(0,0) &= \mathbb{E}\left[\int_0^t \lambda e^{-\lambda s}\left(-s r_T + X c_T + v_T(X,0)\right) ds\right] + e^{-\lambda t}(-t r_T + v_T(0,t)) \\ &= -\frac{1-e^{-\lambda t}}{\lambda} r_T + (1-e^{-\lambda t})(\mathbb{E}\left[X\right] c_T + \mathbb{E}\left[v_T(X,0)\right]) + e^{-\lambda t} v_T(0,t). \end{aligned}$$

By (10) and the result of 1.1°, we get, after some manipulations,

$$v_T(0,t) - v_T(0,0) = \frac{(2\,\mathbb{E}\left[D\right] + \lambda\,\mathbb{E}\left[D^2\right])\left(e^{\lambda t} - 1\right)}{2(1-\rho)(\lambda\,\mathbb{E}\left[D\right] + e^{\lambda\tau})} c_T.$$

2° Let us now consider the energy related costs. By (6) and (8), we get

$$r_P = \mathbb{E}\left[P\right] \cdot c_P = \frac{\left(\lambda\,\mathbb{E}\left[D\right] + \rho e^{\lambda\tau}\right) + \gamma(1-\rho)\left(e^{\lambda\tau} - 1\right)}{\lambda\,\mathbb{E}\left[D\right] + e^{\lambda\tau}} c_P. \tag{12}$$

2.1° Assume again first that $u > 0$ and $t = 0$. Let B_u denote the same "busy period" as in 1.1° so that (11) holds. For the value function at state $(u, 0)$, we have

$$v_P(u,0) = \mathbb{E}\left[B_u\right](c_P - r_P) + v_P(0,0),$$

which implies, by (12) and the previous expression, that

$$v_P(u,0) - v_P(0,0) = \frac{\left((1-\gamma)e^{\lambda\tau} + \gamma\right) u}{\lambda\,\mathbb{E}\left[D\right] + e^{\lambda\tau}} c_P.$$

2.2° Assume now that $u = 0$ and $0 \le t \le \tau$. For the value function at state $(0, 0)$, we clearly have

$$v_P(0,0) = \mathbb{E}\left[\int_0^t \lambda e^{-\lambda s}\left(s(\gamma c_P - r_P) + v_P(X,0)\right) ds\right] + e^{-\lambda t}(t(\gamma c_P - r_P) + v_P(0,t))$$
$$= \frac{1-e^{-\lambda t}}{\lambda}(\gamma c_P - r_P) + (1-e^{-\lambda t})\mathbb{E}\left[v_P(X,0)\right] + e^{-\lambda t} v_P(0,t).$$

By (12) and the result of 2.1°, we get, after some manipulations,

$$v_P(0,t) - v_P(0,0) = -\frac{(\gamma - \lambda\,\mathbb{E}\left[D\right](1-\gamma))\left(e^{\lambda t}-1\right)}{\lambda\left(\lambda\,\mathbb{E}\left[D\right] + e^{\lambda\tau}\right)} c_P.$$

3° Consider finally the switching costs. By (7) and (8), we get

$$r_S = \lambda_S \cdot c_S = \frac{\lambda(1-\rho)}{\lambda\,\mathbb{E}\left[D\right] + e^{\lambda\tau}} c_S. \tag{13}$$

3.1° As before, assume first that $u > 0$ and $t = 0$. Let B_u denote the same "busy period" as in 1.1° and 2.1° so that (11) holds. For the value function at state $(u, 0)$, we have

$$v_S(u,0) = -\mathbb{E}\left[B_u\right] r_S + v_S(0,0),$$

which implies, by (13) and the previous expression, that

$$v_S(u,0) - v_S(0,0) = -\frac{\lambda u}{\lambda\,\mathbb{E}\left[D\right] + e^{\lambda\tau}} c_S.$$

3.2° Assume now that $u = 0$ and $0 \le t \le \tau$. For the value function at state $(0, 0)$, we clearly have

$$v_S(0,0) = \mathbb{E}\left[\int_0^t \lambda e^{-\lambda s}\left(-s r_S + v_S(X,0)\right) ds\right] + e^{-\lambda t}(-t r_S + v_S(0,t))$$
$$= -\frac{1-e^{-\lambda t}}{\lambda} r_S + (1-e^{-\lambda t})\mathbb{E}\left[v_S(X,0)\right] + e^{-\lambda t} v_S(0,t).$$

By (13) and the result of 3.1°, we get, after some manipulations,

$$v_S(0,t) - v_S(0,0) = \frac{e^{\lambda t}-1}{\lambda\,\mathbb{E}\left[D\right] + e^{\lambda\tau}} c_S,$$

which completes the proof. □

6 Experiments

In this section, we present the results of a series of simulation experiments designed to evaluate the gains of combining dispatching control (Sect. 4.1) with the lookahead policies for switching off and turning on servers (Sects. 4.2 and

4.3, respectively). We compare this combined control approach with performing dispatching control only, to quantify the value of the lookahead policies. In addition to investigating the potential gains, we also explore the relative benefits of the two lookahead approaches.

The first experiment (Table 1), serving as a baseline for the remaining experiments, was performed under the following parameter settings: $d = 10$, X exponentially distributed with rate 1, and Λ was chosen such that the resulting loads $\rho = \Lambda/(2\mu)$ in Table 1 were achieved. The cost parameters were $e = 10$, $\gamma = 0.6$, $c_S = 100$, $c_T = 10$, and $c_P = 1$. The average cost rate with dispatching control and both lookahead policies is denoted by r_{LA}, while the average cost rate with dispatching control only is denoted by r_D. Simulations were run for 1000000 simulated time units. At the lowest load ($\rho = 0.2$), adding the lookahead resulted in a higher average cost rate. At first glance, this is counter intuitive, as additional control possibilities should decrease the cost. The issue here is that the lookahead for server turnoffs is too aggressive in keeping the server on – the dispatching control and this lookahead are designed separately and at low loads they appear to actually counteract each other. This effect was seen in varying degrees in all of the experiments. The best gain is 30.6% at $\rho = 0.5$.

Table 1. Average cost rates for the first experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	14.50	22.83	34.46	46.24	55.89	65.98	84.99	140.47
r_{LA}	18.47	22.78	28.31	35.41	45.11	58.78	82.07	137.05

The second experiment (Table 2) was the same as Experiment 1, but c_T was reduced to 1. Here, the problematic behavior at lower loads seen in the first experiment is more pronounced. This is due to energy costs being the dominant part of the average cost rate. The fact that the lookahead policy often keeps the server on is even more disadvantageous, as leaving the server on can only negatively impact the average energy cost rate.

Table 2. Average cost rates for the second experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	8.48	10.39	14.58	19.00	21.80	23.66	26.11	32.13
r_{LA}	8.50	15.71	17.34	19.22	21.19	23.19	25.81	31.95

The third experiment (Table 3) was the same as the first, but c_T was increased to 20. The key observation for this experiment is that the most significant gains are seen at moderate loads (a maximum gain of 41.5% at $\rho = 0.5$). This can be explained by the fact that at moderate loads, when the server that is always on

Table 3. Average cost rates for the third experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	21.13	35.96	56.04	75.61	92.20	111.80	151.48	258.47
r_{LA}	23.02	30.48	40.35	53.43	71.72	98.90	143.63	261.10

is the only server operating, long queue lengths develop so that the other server is required. However, the server that can be switched off is then idle at a high frequency. Thus, it appears that both lookahead policies would be of value. The reality is that the lookahead to turn the server off was the only mechanism that was used – the lookahead to turn the server on was used at most once in each run. This was true of all experiments in this section.

The fourth experiment (Table 4) was the same as the first, but c_T was increased to 1000. Here, the gain of including lookahead is amplified, as the average cost rate is almost completely determined by the average holding cost rate. The maximum gain is 71.1% at $\rho = 0.4$.

Table 4. Average cost rates for the fourth experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	663.03	1290.46	2088.64	2854.21	3559.25	4611.84	6624.41	11992.76
r_{LA}	469.35	787.80	1220.39	1828.96	2683.30	4006.41	6268.40	11946.07

The fifth experiment (Table 5) was the same as the first, but X was chosen to follow a hyperexponential distribution with two phases with means 0.01 and 100 (the overall mean was 1). Here, the maximum gain is 15.7% at $\rho = 0.5$. The presence of very large jobs (high variance of service times) appears to mitigate the gains. The mechanism for this is not obvious, but one possibility is that as large jobs can be sent to both servers, the fact that we are using the Random Routing policy as our base policy for dispatching is problematic – size-aware routing may be a better choice.

Table 5. Average cost rates for the fifth experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	38.12	84.52	185.51	372.18	710.54	1483.53	2899.18	6835.29
r_{LA}	34.34	76.79	171.93	321.70	699.95	1343.16	2797.43	6647.91

The sixth experiment (Table 6) was the same as the first, but X was chosen to be constant. Here, the maximum gain is 15.2% at $\rho = 0.5$. The reduced variance leads to less opportunity for improvement, potentially due to the decreased

Table 6. Average cost rates for the sixth experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	12.94	18.94	31.01	47.30	59.55	70.98	86.71	118.52
r_{LA}	12.94	23.63	30.84	41.06	54.01	68.65	86.03	119.54

Table 7. Average cost rates for the seventh experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	14.32	21.17	29.03	36.74	44.46	54.36	71.29	119.26
r_{LA}	14.31	22.12	26.75	32.31	39.29	49.34	66.74	118.88

variability of the workload at the server that is always on – there are no large fluctuations that require the additional control. Combining the observations from the sixth and seventh experiments, we see that the opportunities for improvement diminish as the service time variance approaches very small or very large values.

The seventh experiment (Table 7) was the same as the first, but d was reduced to 1. The maximum improvement is 13.7% at $\rho = 0.5$. The fact that the maximum improvement has decreased is not surprising, as the short setup times mean that the penalty paid for poor turnoff decisions is not as severe.

The eighth experiment (Table 8) was the same as the first, but d was increased to 100. The maximum gain is 22.2%, at $\rho = 0.5$. The gains are generally lower as the routing control tends to keep the server that can be switched off busy at all times (thus avoiding the long setup times), so there are less opportunities for the lookahead to be used.

Table 8. Average cost rates for the eighth experiment

ρ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_D	14.50	23.53	49.18	111.34	169.28	212.47	249.52	325.44
r_{LA}	19.81	27.39	44.28	91.15	168.31	212.01	249.41	324.69

In summary, the simulation results suggest that:

1. The lookahead for server turnoffs is the mechanism for reducing the cost.
2. The most gain from dynamic switch-off control is made at "moderate" values of load, service time variability, and setup times. Outside of these values, dispatching control alone appears to be sufficient.
3. When energy costs are dominant, dispatching control also appears to be sufficient.

Note that dispatching decisions indirectly control also the energy consumption due to the assumed default configuration where Server 1 was NeverOff and Server 2 InstantOff, explaining the latter two observations.

7 Conclusions

We considered the joint problem of combining dispatching control and server sleep state control, computing near optimal policies using one step of policy iteration for dispatching and lookahead techniques for server sleep state control. In addition to providing these policies explicitly, we identified when this joint control approach is most effective. Some issues for future work:

1. Is it possible to quantify the gap between state-dependent and state-independent sleep state control? This would give insight into the value of state information in making these control decisions. The work in [21] suggests that this value goes to zero in a many server asymptotic regime, but the answer to this question for finite systems is of interest. Answering this question would involve characterizing (near) optimal state-independent sleep state control.
2. How does the approach scale? One important related question is determining which servers are always on.
3. If the servers are not FCFS can similar gains be expected?
4. For high variance service time distributions, it may be useful to consider a different initial policy for the dispatching control problem. One possibility is a SITA-like policy [3].

References

1. Barroso, L., Hölzle, U.: The case for energy-proportional computing. IEEE Comput. **40**(12), 33–37 (2007)
2. Chen, Y., Das, A., Qin, W., Sivasubramaniam, A., Wang, Q., Gautam, N.: Managing server energy and operational costs in hosting centers. In: Proceedings of ACM Sigmetrics 2005, pp. 303–314 (2005)
3. Crovella, M., Harchol-Balter, M., Murta, C.: Task assignment in a distributed system: Improving performance by unbalancing load. In: Proceedings of ACM Sigmetrics 1998, pp. 268–269 (1998)
4. Gandhi, A., Doroudi, S., Harchol-Balter, M., Scheller-Wolf, A.: Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward. In: Proceedings of ACM Sigmetrics 2013, pp. 153–166 (2013)
5. Gandhi, A., Gupta, V., Harchol-Balter, M., Kozuch, M.: Optimality analysis of energy-performance trade-off for server farm management. Perform. Eval. **67**, 1155–1171 (2010)
6. Gandhi, A., Harchol-Balter, M., Adan, I.: Server farms with setup costs. Perform. Eval. **67**, 1123–1138 (2010)
7. Gandhi, A., Harchol-Balter, M., Raghunathan, R., Kozuch, M.: AutoScale: Dynamic, robust capacity management for multi-tier data centers. ACM Trans. Comput. Syst. **30**, 1–26 (2012)
8. Gebrehiwot, M., Aalto, S., Lassila, P.: Optimal sleep-state control of energy-aware M/G/1 queues. In: Proceedings of Value Tools 2014, pp. 82–89 (2014)
9. Gebrehiwot, M., Aalto, S., Lassila, P.: Energy-performance trade-off for processor sharing queues with setup delay. Oper. Res. Lett. **44**, 101–106 (2016)

10. Gebrehiwot, M.E., Aalto, S., Lassila, P.: Energy-aware server with SRPT scheduling: analysis and optimization. In: Agha, G., Van Houdt, B. (eds.) QEST 2016. LNCS, vol. 9826, pp. 107–122. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-43425-4_7
11. Harchol-Balter, M.: Performance Modeling and Design of Computer Systems: Queueing Theory in Action. Cambridge University Press, New York (2013)
12. Hyytiä, E., Penttinen, A., Aalto, S.: Size- and state-aware dispatching problem with queue-specific job sizes. Eur. J. Oper. Res. **217**, 357–370 (2012)
13. Hyytiä, E.: Lookahead actions in dispatching to parallel queues. Perform. Eval. **70**, 859–872 (2013)
14. Hyytiä, E., Righter, R., Aalto, S.: Task assignment in a heterogeneous server farm with switching delays and general energy-aware cost structure. Perform. Eval. **75–76**, 17–35 (2014)
15. Hyytiä, E., Righter, R., Aalto, S.: Energy-aware job assignment in server farms with setup delays under LCFS and PS. In: Proceedings of ITC 26 (2014)
16. Maccio, V., Down, D.: On optimal policies for energy-aware servers. In: Proceedings of IEEE MASCOTS 2013, pp. 31–39 (2013)
17. Maccio, V., Down, D.: On optimal control for energy-aware queueing systems. In: Proceedings of ITC 27, pp. 98–106 (2015)
18. Maccio, V., Down, D.: Exact analysis of energy-aware multiserver queueing systems with setup times. In: Proceedings of IEEE MASCOTS 2016, pp. 11–20 (2016)
19. Mitrani, I.: Service center trade-offs between customer impatience and power consumption. Perform. Eval. **68**, 1222–1231 (2011)
20. Mitrani, I.: Managing performance and power consumption in a server farm. Annals Oper. Res. **202**, 121–134 (2013)
21. Mukherjee, D., Dhara, S., Borst, S., Leeuwaarden, J.: Optimal service elasticity in large-scale distributed systems. In: Proceedings of ACM Sigmetrics 2017 (2017)
22. Penttinen, A., Hyytiä, E., Aalto, S.: Energy-aware dispatching in parallel queues with on-off energy consumption. In: Proceedings of IEEE IPCCC 2011 (2011)
23. Phung-Duc, T.: Multiserver queues with finite capacity and setup time. In: Gribaudo, M., Manini, D., Remke, A. (eds.) ASMTA 2015. LNCS, vol. 9081, pp. 173–187. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18579-8_13
24. Phung-Duc, T.: Exact solutions for M/M/c/Setup queues. Telecommun. Syst. **64**, 309–324 (2017)
25. Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York (1994)
26. Slegers, J., Thomas, N., Mitrani, I.: Dynamic server allocation for power and performance. In: Kounev, S., Gorton, I., Sachs, K. (eds.) SIPEW 2008. LNCS, vol. 5119, pp. 247–261. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69814-2_16
27. Welch, P.: On a generalized M/G/1 queuing process in which the first customer of each busy period receives exceptional service. Oper. Res. **12**, 736–752 (1964)
28. Whittle, P.: Optimal Control: Basics and beyond. Wiley, Chichester (1996)

On the Value of Service Demand Estimation for Auto-scaling

André Bauer(✉), Johannes Grohmann, Nikolas Herbst, and Samuel Kounev

University of Würzburg, Würzburg, Germany
{andre.bauer,johannes.grohmann,nikolas.herbst,
samuel.kounev}@uni-wuerzburg.de
https://descartes.tools

Abstract. In the context of performance models, service demands are key model parameters capturing the average time individual requests of different workload classes are actively processed. In a system under load, due to measurement interference, service demands normally cannot be measured directly, however, a number of estimation approaches exist based on high-level performance metrics. In this paper, we show that service demands provide significant benefits for implementing modern auto-scalers. Auto-scaling describes the process of dynamically adjusting the number of allocated virtual resources (e.g., virtual machines) in a data center according to the incoming workload. We demonstrate that even a simple auto-scaler that leverages information about service demands significantly outperforms auto-scalers solely based on CPU utilization measurements. This is shown by testing two approaches in three different scenarios. Our results show that the service demand-based auto-scaler outperforms the CPU utilization-based one in all scenarios. Our results encourage further research on the application of service demand estimates for resource management in data centers.

Keywords: Service demand estimation · Auto-scaling
Online estimation · Elastic cloud computing

1 Introduction

The cloud computing paradigm has a high impact in the ICT domain as it allows on-demand access to data center resources (e.g., networks, servers, storage and applications). In order to guarantee a reliably operating service, mission-critical applications usually run with a fixed amount of resources. This has some drawbacks: On the one hand, if the application is not fully used, the resource consumption becomes inefficient; on the other hand, if an unexpected event occurs, the performance and availability is not ensured.

The domain of auto-scaling is concerned with automatically and precisely allocating the required amount of resources (e.g., number of VMs) for a given

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 142–156, 2018.
https://doi.org/10.1007/978-3-319-74947-1_10

time period and changing this allocation as the demand changes. Existing auto-scaling mechanisms either assume as input an estimate on the processing speed of the resources or rely on measured resource utilization averages (r.t., Sect. 5).

In this work, we compare two auto-scaling approaches with identical underlying decision logic under changing workloads. The first approach uses the CPU utilization to scale the system, while the second one uses online service demand estimation. Both approaches scale three different application types: (i) a scalable application limited in performance mainly by hardware contention, (ii) a second application that has software bottlenecks leading to software contention, and (iii) a third application that exhibits both hardware and software contention. Given these three application types and the two different approaches to realize auto-scaling, we pose ourselves the following research questions:

RQ1: How can CPU utilization-based and service demand-based auto-scaling mechanisms be compared in a fair manner?

RQ2: In which scenarios does service demand-based auto-scaling outperform a CPU utilization-based mechanism?

RQ3: What are benefits of using service demand estimates instead of utilization measurements for automatic scaling decisions?

We structure the three main contributions of this paper as follows: We address RQ1 by discussing the competing auto-scaling approaches in Sect. 2. Then, the experiment setup is introduced. Afterwards, RQ2 and RQ3 are answered by comparing and quantifying the two approaches in three different scenarios. In Sect. 5, we summarize related work before concluding the paper.

2 Service Demand Estimation for Auto-scaling

In this section, the foundations of the service demand estimation are explained. Afterwards, the competing auto-scaling approaches are introduced.

2.1 Service Demand Estimation

Service demands are a key parameter of stochastic performance models. A service demand is the average time a unit of work (e.g., request or transaction) spends obtaining service from a resource (e.g., CPU or hard disk) in a system, over all visits at the resource, excluding any waiting times [1,2]. Different requests can be grouped into different *workload classes*. Service demands are normally considered on a per workload class basis.

In most realistic systems, the direct measurement of service demands is not feasible during operation [3] due to instrumentation overheads and possibly measurement interference. Willneker et al. [4] show that statistical estimation approaches can provide a comparable accuracy to direct measurements. Thus, we focus on statistical approaches for estimating the service demands.

The advantage of statistical estimation approaches compared to direct measurement techniques is their general applicability and low overheads. Estimation

approaches typically rely only on coarse-grained measurements from the system (e.g., CPU utilization and end-to-end average response times) that can be obtained easily with monitoring tools without the need for fine-grained code instrumentation. These measurements are routinely collected for many applications (e.g., in data centers) and therefore estimation approaches are also applicable on systems serving production workloads.

Over the years, a number of approaches to service demand estimation have been proposed based on different statistical estimation techniques (e.g., linear regression [5,6] or Kalman filters [7,8]) and combined with laws from queueing theory. We refer to Spinner et al. [3] for an overview as well as a classification and experimental evaluation of the different approaches for service demand estimation.

LibReDE: For estimating the service demands in an online setting, our approach uses the Library for Resource Demand Estimation (LibReDE) [9]. LibReDE[1] is a library of ready-to-use implementations of state-of-the-art approaches to service demand estimation that can be used for online and offline analysis. It supports several different approaches to service demand estimation.

For the sake of simplicity, here we restrict ourselves to using only one approach, based on the Service Demand Law [6]:

The service demand law [2] states that the the service demand $D_{i,c}$ of c at resource i can be obtained by dividing the utilization $U_{i,c}$ of a resource i due to workload class c by the system throughput $X_{0,c}$ of workload class c:

$$D_{i,c} = \frac{U_{i,c}}{X_{0,c}}. \tag{1}$$

However, usually $U_{i,c}$ is not measurable, since we can only measure the total utilization of resource i caused by all workload classes running on i. Therefore, we have to partition the utilization among the different workload classes. Menascé et al. [2] and Lazowska et al. [1] solve this by collecting additional per-class data, while Brosig et al. [6] estimate it based on the response times of the different classes. Therefore, we only need to measure the average CPU utilization and the throughput of each workload class in order to estimate the service demands.

2.2 Competing Methods

In order to investigate the value of service demand estimation for auto-scaling, we compare the same threshold-based auto-scaling algorithm, as implemented for example on Amazon Web Services (AWS) EC2, but feed it with two different input parameters: (i) the measured average CPU utilization and (ii) the average system utilization based on queueing theory (see Eq. 1). The main advantage of using the CPU utilization is that it is easy to measure, however, the thresholds for the scaling have to be tuned for each application individually and measurable load levels are limited at 100%. In contrast, the average system utilization

[1] LibReDE: https://descartes.tools/librede/.

based on queueing theory has no limitation and the thresholds can be defined independent of the application. However, the determination of the average system utilization requires application level metrics like response times and trough put per request class. The thresholds can be either learned leveraging machine learning approaches or as in our case determined based on experience.

Taking AWS as example, per default they provide metrics such as CPU utilization, disk IO (disk read operations or disk write operation), and network IO for the auto-scaling decision. The reasons for choosing CPU utilization as scaling indicator are threefold: (i) in many cases the bottleneck resource may not known at configuration time, (ii) our software contention scenario is limited by the number of unlocked files and not by an IO rate, and finally, (iii) when using IO metrics, a deep knowledge of the IO characteristics of the machine is required, which appears unfeasible for cloud deployments.

The auto-scaling mechanism communicates with the cloud every minute and gathers VM specific information, such as the amount of running VMs and the average CPU utilization, and application specific information, such as request arrival rates. Algorithm 1 illustrates the simplified decision logic that forms the basis for both approaches. Important parameters for decision making are *up_threshold* and *down_threshold*. Input parameters are the current average system utilization $\overline{\rho}$ and the number of currently running VMs *vms*. In contrast to the CPU utilization-based approach where $\overline{\rho}$ is equal to the measured CPU load, the service demand-based approach uses the average system utilization based on the service demand law from queueing theory that offers a good trade-off between estimation time and accuracy [10]. The system utilization ρ is derived from the the arrival rate multiplied with the estimated service demand respectively the highest service demand if multi-class systems are scaled [11]. The service demand estimate is updated online every 10 min, as service demand estimated are expected to not change significantly in a short period of time.

In the second line, Algorithm 1 checks if the average system utilization $\overline{\rho}$ per VM is greater than a predefined threshold. While this condition is true, the new average system utilization per VM is calculated after iteratively increasing the number of VMs. Otherwise, if the average system utilization per VM is less than a predefined threshold, the number of VMs is decreased iteratively until $\overline{\rho}$ is greater than the threshold. Finally, the algorithm returns the number of VMs that are required (amount > 0) or that can be released (amount < 0).

3 Experiment Description

In order to obtain a authentic workload with a time-varying behavior, we use the Retailrocket[2] trace. This trace represents HTTP requests to servers of an anonymous real-world e-commerce website during June 2015. For our experiments, we sample the arrival rates every 15 min, i.e., each day consists of 96 data points. Furthermore, we crop out two days and speed-up the trace by the factor 15 so that each data point represents one minute.

[2] Retailrocket Source: https://www.kaggle.com/retailrocket/ecommerce-dataset.

ALGORITHM 1. Decision logic.

```
Compute-Optimal-VMs (double ρ̄, int vms)
    amount = 0;
    if ρ̄ > up_threshold then // is ρ̄ > a predefined threshold
        while ρ̄ > up_threshold do
            amount++;
            ρ̄ = ρ̄ * (vms / (vms + amount)); // calculates the new average utilization
    else if ρ̄ <= down_threshold then // is ρ̄ ≤ a predefined threshold
        while ρ̄ <= down_threshold do
            amount−−;
            ρ̄ = ρ̄ * (vms / (vms + amount)); // calculates the new average utilization
            if ρ̄ > up_threshold then
                amount++; // undo
    return amount;
```

The auto-scaling mechanisms are configured to monitor and auto-scale three different applications: (i) an application with its performance limited by hardware contention (CPU), (ii) an application exhibits software contention, and (iii) an application that exhibits both hardware and software contention. The first application is a CPU-intensive Java Enterprise application that is a re-implementation of the LU worklet from SPEC's Server Efficiency Rating Tool SERT™2. The application calculates the LU Decomposition [12] of a random generated $n \times n$ matrix, where n is the GET parameter of each HTTP request.

The second application is also a Java Enterprise application. This application is used by a content-delivery service provider with limited capacities. Here, the incoming HTTP requests try to read a file out of a limited pool of randomly generated files. While a file is being read by an HTTP request, it is locked and cannot be accessed by other requests, i.e., an incoming HTTP-request either successfully reads a file or waits until a file is unlocked.

The third scenario contains the constraints of both scenarios. At first the application calculates the LU decomposition. Afterwards, it tries to read a file out of the limited pool.

All applications are deployed on WildFly application servers in our private cloud infrastructure. Here, we use an Apache CloudStack[3] cloud that manages virtualized Xen-Server hosts. This cloud environment is running in a cluster of 11 homogeneous HP servers. Eight of them are managed by CloudStack. The last three servers are not part of the cloud and are used for hosting the software for the cloud management as well as the benchmark framework: (i) the load-balancer (Citrix Netscaler[4]) and the cloud management for CloudStack, (ii) the auto-scaling mechanisms, and (iii) the load driver and the experiment controller. The specification of each physical machine can be found in Table 1.

[3] Apache CloudStack: https://cloudstack.apache.org/.
[4] Citrix Netscaler: https://www.citrix.de/products/netscaler-adc/.

Elasticity Benchmarking Framework: In order to evaluate the two approaches, we use the BUNGEE Cloud Elasticity Benchmark controller [13]. First, the controller constructs for the SuT (System under Test) a discrete mapping function that determines for each load intensity the associated minimum amount of resources required to meet the SLOs. Based on this mapping and a predefined workload profile, the SuT is stressed while BUNGEE monitors the supplied VMs. After the experiment, the elasticity and user-oriented metrics based on the collected monitoring data are calculated.

Table 1. Specification of the servers.

Criteria	Server	Worker VMs
Model	HP DL160 Gen9	–
Operating system	Xen-Server	Centos 6.5
CPU	8 cores	2 vcores
Memory	32 GB	4 GB

4 Results

This section presents the experiments and the respective findings. First, the specific metrics used to evaluate and compare the two approaches are defined. Then, the hardware contention scenario is discussed. In Sect. 4.3, the software contention scenario is examined. Afterwards, the mixed scenario is discussed. Finally, the results are discussed with their associated threats to validity.

4.1 Quantifying the Auto-scaler Performance

In order to evaluate the two competing approaches, on the one hand, we use user-oriented metrics such as SLO (service level objective) violations in combination with the average response time. On the other hand, we use system-oriented elasticity metrics endorsed by the Research Group of the Standard Performance Evaluation Corporation (SPEC) [14]. In particular, we use the provisioning accuracy and the wrong provision time share. Using only single metrics, it is hard to gain insight into the performance differences between the two approaches. Hence, we use multiple metrics and summarize the gain of using each approach using an aggregate metric called elasticity speedup. Each elasticity metric is described in the remainder of this section. For the following equations, we define: (i) T as the experiment duration and time $t \in [0, T]$, (ii) s_t as the resource supply at time t, and (iii) d_t as the demanded resource units at time t. The demanded resource units d_t is the minimal amount of VMs required to meet the SLOs under the load intensity at time t. Δt denotes the time interval between the last and the current change either in demand d or supply s. The curve of demanded resource units d over time T is derived by BUNGEE, see Sect. 3. The resource supply s_t is the monitored number of running VMs at time t.

Provisioning accuracy θ_U and θ_O: The provisioning accuracy describes the relative amount of resources that are under-provisioned, respectively, over-provisioned during the measurement interval. In order words, the under-provisioning accuracy θ_U is the amount of missing resources normalized by the current demanded resource units that are required to meet the SLOs normalized by the experiment time. Similarly, the over-provisioning accuracy θ_O is the amount of resources that the auto-scaler supplies in excess. The range of this metric is the interval $[0, \infty)$, where 0 is the best value and indicates that the supply curve follows lays on demand curve during the entire measurement interval.

$$\theta_U[\%] := \frac{100}{T} \cdot \sum_{t=1}^{T} \frac{max(d_t - s_t, 0)}{d_t} \Delta t \tag{2}$$

$$\theta_O[\%] := \frac{100}{T} \cdot \sum_{t=1}^{T} \frac{max(s_t - d_t, 0)}{d_t} \Delta t \tag{3}$$

Wrong provisioning time share τ_U and τ_O: The wrong provisioning time share captures the time in which the system is an under-provisioned, respectively over-provisioned, state during the experiment interval, i.e., the under-provisioning time share τ_U is the time relative to the measurement duration, in which the system is under-provisioned. Similarly, the over-provisioning time share τ_O is the time relative to the measurement duration in which the system is over-provisioned. The range of this metric is the interval $[0, 100]$. The best value 0 is achieved, when the system has during the measurement no over- or under-provisioning.

$$\tau_U[\%] := \frac{100}{T} \cdot \sum_{t=1}^{T} max(sgn(d_t - s_t), 0) \Delta t \tag{4}$$

$$\tau_O[\%] := \frac{100}{T} \cdot \sum_{t=1}^{T} max(sgn(s_t - d_t), 0) \Delta t \tag{5}$$

Elastic Speedup ϵ: This comparing method calculates for each approach its gain based on its scaling behavior compared to the no auto-scaling scenario. This approach allows to compare our two approach by taking only the system-oriented metrics into account. In other words, the elasticity metrics $x = (\theta_U, \theta_O, \tau_U, \tau_O)$ of each approach a is compared to the metrics of the no-auto scaling scenario n. To this end, the geometrical mean of the ratio between each metric pair is calculated. If the value is greater than 1, the proposed method is better than having no auto-scaler and the value reflects the gain. If the values is less than 1, the approach is worse than having no auto-scaler. Mathematically, the elastic speedup ϵ for an auto-scaler a based on the no auto-scaling scenario n can be formulated as:

$$\epsilon_n := \left(\frac{\theta_{U,n}}{\theta_{U,a}} \cdot \frac{\theta_{O,n}}{\theta_{O,a}} \cdot \frac{\tau_{U,n}}{\tau_{U,a}} \cdot \frac{\tau_{O,n}}{\tau_{O,a}} \right)^{\frac{1}{4}} \tag{6}$$

4.2 Hardware Contention Scenario

The results for the hardware contention scenario are depicted in Fig. 1. This diagram is divided into three parts: The first part shows the scaling behavior, the second one the average system utilization for each approach, and the last one the estimated service demand. In the first part, the black line describes the curve of demanded resource units (determined by BUNGEE, see Sect. 3), the red line the scaling behavior for the CPU utilization-based approach, and the blue line the scaling behavior for the service demand-based approach. Here, both approaches tend to over-provision the system during the decreasing tail of each day. However, the auto-scaler based on CPU utilization has more instances over-provisioned during this period compared to the service demand-based approach. Furthermore, the service demand-based auto-scaler can handle the increasing load during each day more efficiently than the CPU utilization-based one.

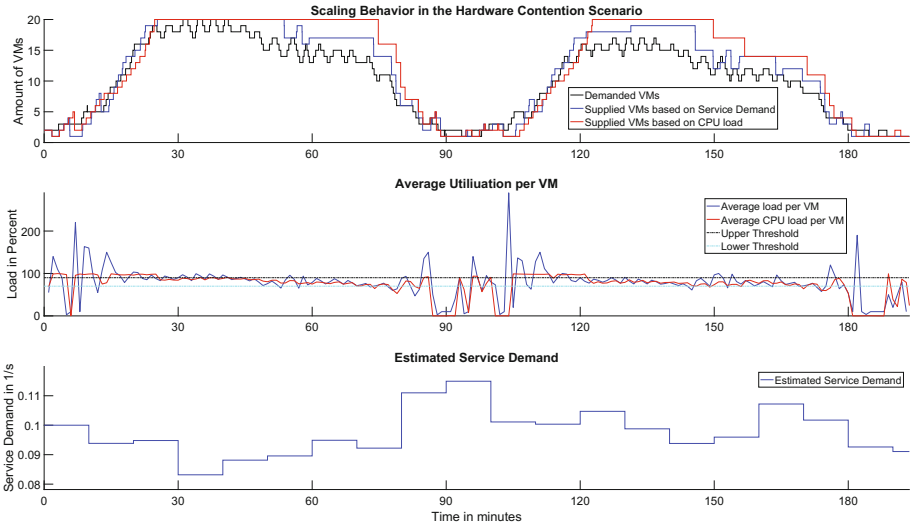


Fig. 1. Scaling behavior in the hardware contention scenario. (Color figure online)

These observations can be explained by looking at the middle part of the figure where the blue line shows the average system utilization and the red line the average CPU utilization. The black dashed line represents the threshold for up-scaling (90%) and the cyan dashed line the threshold for down-scaling (70%). While the CPU utilization is limited by 100%, the system utilization can have values higher than this limit, for instance, at minute 115 where the system is in under-provisioned state for both approaches, the CPU utilization is 100% and the system utilization is 160%. The service demand-based approach assigns 3 VMs to handle this utilization and the CPU utilization-based one only 1 VM.

The estimated service demand (see Sect. 2.2) for calculating the system utilization, depicted in the bottom part of the figure, has the average value of $0.099 \pm 0.016 \frac{1}{s}$.

Table 2. Results for the hardware contention scenario.

Metric	No scaling	Service demand-based	CPU utilization-based
θ_U (accuracy$_U$)	**3.20**%	7.54%	7.46%
θ_O (accuracy$_O$)	203.28%	**14.60**%	23.57%
τ_U (timeshare$_U$)	22.89%	**19.10**%	22.20%
τ_O (timeshare$_O$)	67.38%	**62.56**%	62.65%
ϵ (Elastic Speedup)	1.00	**1.66**	1.42
ψ (SLO violations)	45.72%	**8.40**%	12.67%
Avg. #VMs	15.00	8.58	7.93
Avg. response time	2.62 s	**0.70** s	0.94 s
Med. response time	2.86 s	**0.22** s	0.24 s

To enable quantitative comparison, we calculated the elasticity metrics (see Sect. 4.1) as well as some user-oriented metrics listed in Table 2. Here, each row shows a metric and each column an auto-scaler. The best values are printed in bold. As a baseline scenario ("no auto-scaling"), we run 15 VMs (75% of the available VMs) throughout the experiment duration. When comparing only the individual elasticity metrics, the service demand-based approach exhibits the best results for 3 out of 4 metrics. Thus, it also achieves the highest elastic speedup (1.66). The CPU utilization-based approach also has an elastic speedup greater than 1, i.e., both approaches are more efficient than the baseline (no auto-scaling) scenario. Furthermore, both approaches use less VMs, achieve significantly higher SLO conformance, and have a lower response time than the no auto-scaling scenario. However, the service demand-based auto-scaler has the lowest amount of SLO violations (8.40%) and the lowest response time (0.70 s).

4.3 Software Contention Scenario

Similar to Sect. 4.2, the results for the software contention scenario are depicted in Fig. 2. The scaling behavior of both approaches is shown in the upper diagram, where the black line describes the curve of demanded resource units, the red line the scaling behavior for the CPU utilization-based approach, and the blue line the scaling behavior for the service demand-based approach. In contrast to the hardware contention scenario, this scenario marginally stresses the CPU. Although the thresholds for up-scaling (30%) and down-scaling (5%) are adjusted, the CPU utilization-based approach is not able to meet the required VMs during the two days. The supply curve of the service demand-based approach is close to the demand curve. There are only few intervals in which the system is in an under-provisioned state for a short time. In analogy to the hardware contention scenario, the system utilization is more suitable to describe the required amount of VMs. In both scenarios the service demand-based approach uses the same thresholds in contrast to the CPU utilization-based one.

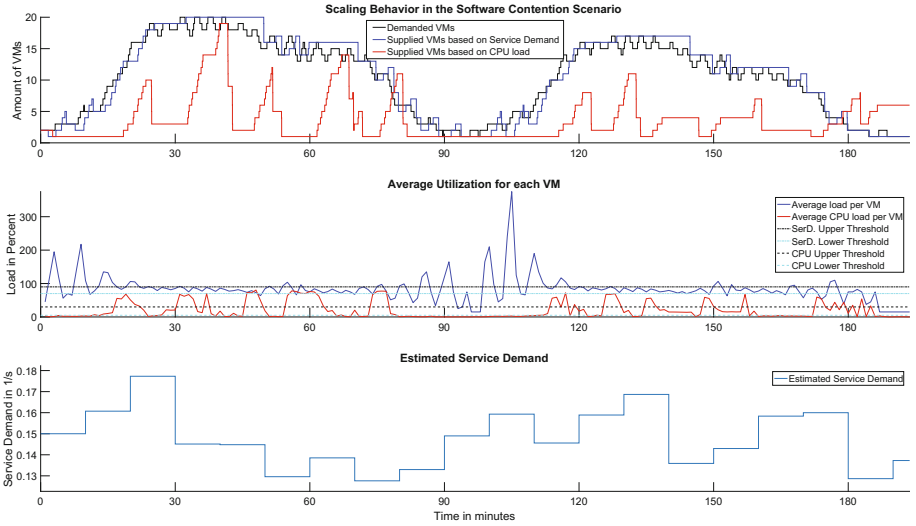


Fig. 2. Scaling behavior in the software contention scenario. (Color figure online)

Table 3. Results for the software contention scenario.

Metric	No scaling	Service demand-based	CPU utilization-based
θ_U (accuracy$_U$)	**3.20**%	7.64%	60.45%
θ_O (accuracy$_O$)	203.28%	**8.36**%	24.67%
τ_U (timeshare$_U$)	22.89%	**27.02**%	86.83%
τ_O (timeshare$_O$)	67.38%	43.37%	**9.29**%
ϵ (Elastic Speedup)	1.00	**1.91**	0.96
ψ (SLO violations)	8.64%	**6.27**%	97.25%
Avg. #VMs	15.00	8.21	6.69
Avg. response time	1.07 s	1.04 s	1.97 s
Med. response time	**0.98** s	**0.98** s	2.00 s

The estimated service demand (see Sect. 2.2) for calculating the system utilization, depicted in the last row of the figure, has the average value of $0.153 \pm 0.025 \frac{1}{s}$.

In order to compare the two approaches, we calculate the elasticity metrics, collect user information, and compare the scaling behavior with the baseline no (auto-scaling scenario) in which 15 VMs are permanently running during the experiment. The results are summarized in Table 3. Here, each row represents a metric and each column an auto-scaler. The best values are highlighted in bold. The service demand-based auto-scaler exhibits the best elastic speedup, the lowest amount of SLO violations, and the lowest response time. In contrast, the CPU utilization-based approach has an elastic speedup lower than 1, i.e., the performance judged by the elasticity metrics is worse than the no auto-scaling scenario. This can also be seen by the high amount of SLO violations (97.25%).

4.4 Mixed Contention Scenario

The results for the mixed contention scenario are depicted in Fig. 3. Here, the scaling behavior of the service demand-based approach (blue curve), the scaling behavior of the CPU utilization-based approach (red curve), and the demanded resource units (black curve) are shown in the upper part of the figure. As this application has both software and hardware contention, we determined and calibrated the upper-threshold (55%) and lower-threshold (40%) for the CPU utilization-based approach. While this approach has problems to meet the required VMs at pitch of the first day, the remaining days are covered better than in the software contention scenario. Furthermore, there is less over-provisioning than in the hardware contention scenario. Similar to the previous scenarios, the service demand-based approach has almost no under-provisioning and tends to over-provision slightly.

The estimated service demand (see Sect. 2.2) for calculating the system utilization, depicted in the bottom row of the figure, has the average value of $0.238 \pm 0.043 \frac{1}{s}$.

The observed scaling behavior is quantified by the metrics shown in Table 4. Here, each row represents a metric and each column an auto-scaler. While, the service demand-based approach has the best under-provision timeshare metric, it also has the highest elastic speedup (1.71) and the lowest SLO violations (4.77%). Also the CPU utilization-based approach has a value higher than 1 and thus, both approaches are more efficient than the no auto-scaling scenario, in which 15 VMs are running throughout the experiment.

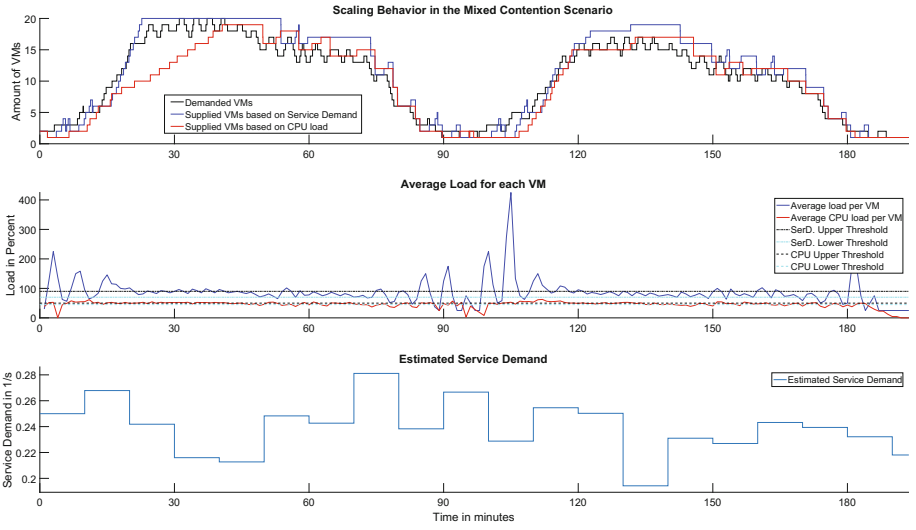


Fig. 3. Scaling behavior in the mixed contention scenario. (Color figure online)

Table 4. Results for the mixed contention scenario.

Metric	No scaling	Service demand-based	CPU utilization-based
θ_U (accuracy$_U$)	**3.20**%	7.13%	14.61%
θ_O (accuracy$_O$)	203.28%	14.12%	**7.05**%
τ_U (timeshare$_U$)	22.89%	**19.28**%	42.86%
τ_O (timeshare$_O$)	67.38%	59.90%	**35.03**%
ϵ (Elastic Speedup)	1.00	**1.71**	1.60
ψ (SLO violations)	6.40%	4.77%	11.96%
Avg. #VMs	15.00	8.96	9.51
Avg. response time	1.95 s	**1.94** s	1.95 s
Med. response time	**1.95** s	**1.95** s	**1.95** s

4.5 Summary

When comparing the different scenarios, the service demand-based auto-scaler behaves in a similar manner in each scenario and per day. In contrast, the CPU utilization-based approach behaves differently depending on the scenario. Furthermore, the service demand-based auto-scaler achieves the best values of the elasticity and user-oriented metrics. While the CPU utilization-based approach has problems in the software contention scenario, it is still more efficient than using no auto-scaling in the hardware contention and mixed contention scenarios. Note that the elasticity metrics of the no auto-scaling are identical per definition throughout all experiments.

Considering the applicability of both approaches, the CPU utilization-based auto-scaler is easy to setup and needs no further instrumentation. The CPU utilization can be gathered through standard tools or services such as SNMP (Simple Network Management Protocol). In contrast, the service demand-based approach requires a more complex, yet still not intrusive, instrumentation. Either the service demand has to be determined ahead of time assuming it is static or the service demand has to be estimated online as done in this paper. For this, the resource estimator needs structural application knowledge and information that may require a basic instrumentation of the application to monitor high-level metrics such as like request completion rates and response times.

To avoid the performance variability of public cloud infrastructures due to overbooking of resources and background-traffic, which also renders CPU utilization measures both unstable and unreliable [15], we ran the experiments in our private cloud environment under controlled conditions. Based on our experience with experiments in public clouds, CPU utilization based auto-scaling is supposed to perform worse than under controlled conditions while having a minor impact on the service demand-based approach.

We chose two similar days of the Retailrocket trace and conduct long experiments in order to validate the measurements internally respectively to have the second day as repetition of the first one. As the scaling behavior is influenced by the defined thresholds, we cannot prove that we have chosen the optimal ones.

5 Related Work

We studied two surveys on existing auto-scalers from Jennings and Stadler [16], and Lorido-Botran et al. [17]. Besides a broad overview of existing auto-scalers, the survey in [17] proposes a classification of auto-scalers into five groups: (i) threshold-based rules [18,19], (ii) queueing theory [20,21], (iii) control theory [22,23], (iv) reinforcement learning [24,25], and (v) time series analysis [26,27]. While analyzing the auto-scalers in these survey, we can conclude that service demands - on a higher abstraction level the server's processing speed - is estimated indirectly and in an application-specific way for threshold-based rule approaches, assumed to be provided or calibrated as input for queueing and control theory auto-scaler, or learned over a training phase for reinforcement learning approaches. As examples for approaches leveraging time-series analysis, the auto-scaler called CloudScale, which is designed by Shen et al. [28], predicts the demand for resources with fast Fourier transformation (FFT) algorithms. A similar approach, called AGILE, is proposed by Nguyen et al. [29] who leverages wavelets instead. However, the demand for resources in the context of both of the above papers is understood as the CPU utilization. They sample the CPU utilization as time series and predict the future CPU load, instead of estimating the current processing speed of servers in terms of resource consumption per request.

To the best of our knowledge, there is only one auto-scaler proposed by Spinner et al. [30] that uses online service demand estimation to scale-out one web-server by adding virtual CPU cores. In contrast to this work, here we focus on horizontal scaling by adding virtual machine instances to a load-balancer. However, the results of Spinner et al. [30] support our message, as the results demonstrate an increased controller stability for an service-demand based approach compared a classical one based on CPU utilization threshold.

6 Conclusion and Discussion

In this paper, we compare two different approaches for auto-scaling: (i) based on measurements of the CPU utilization and (ii) based on service demand estimation as input values for an identical decision logic. To answer RQ1 "How can CPU utilization-based and service demand-based auto-scaling mechanisms be compared in a fair manner?", the two approaches scale three different types of applications: (i) a scalable application limited in performance mainly by hardware contention, (ii) a second application that has software bottlenecks, and (iii) a third application that exhibits both hardware and software contention. We use an established set of elasticity metrics to evaluate and compare the two auto-scaling approaches on a level playing field. We summarize our research findings as follows: The service demand-based approach is independent of the scenario, i.e., the service demand estimation does not rely on knowing the bottleneck resource and can be configured independently of the application. Furthermore, it achieves the best values of the various metrics in all scenarios and exhibits a similar scaling behaviour. These findings answer both RQ2

"In which scenarios does service demand-based auto-scaling outperform a CPU utilization-based mechanism?" and RQ3 "What are benefits of using service demand estimates instead of utilization measurements for automatic scaling decisions?".

We are confident that our results can encourage further research activity in the application of service demand estimation from the performance modeling domain for resource management in cloud data centers. For future work, we plan to extend the set scope of our analysis by conducting experiments in other cloud environments and investigating further types of applications. Additionally, a workload containing multiple workload classes will be considered. Finally, more research on different approaches to service demand estimation is planned, since the quality of the service demand estimates has direct influence on the auto-scaler decisions.

Acknowledgements. This work was co-funded by the German Research Foundation (DFG) under grant No. (KO 3445/11-1) and by Google Inc. (Faculty Research Award).

Arif Merchant, Google Inc., contributed with helpful ideas and feedback.

References

1. Lazowska, E.D., Zahorjan, J., Graham, G.S., Sevcik, K.C.: Quantitative System Performance: Computer System Analysis Using Queueing Network Models. Prentice-Hall, Inc., Upper Saddle River (1984)
2. Menascé, D.A., Dowdy, L.W., Almeida, V.A.F.: Performance by Design: Computer Capacity Planning by Example. Prentice Hall PTR, Upper Saddle River (2004)
3. Spinner, S., Casale, G., Brosig, F., Kounev, S.: Evaluating approaches to resource demand estimation. Perform. Eval. **92**, 51–71 (2015)
4. Willnecker, F., Dlugi, M., Brunnert, A., Spinner, S., Kounev, S., Gottesheim, W., Krcmar, H.: Comparing the accuracy of resource demand measurement and estimation techniques. In: Beltrán, M., Knottenbelt, W., Bradley, J. (eds.) EPEW 2015. LNCS, vol. 9272, pp. 115–129. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23267-6_8
5. Rolia, J., Vetland, V.: Parameter estimation for performance models of distributed application systems. In: CASCON 1995, p. 54. IBM Press (1995)
6. Brosig, F., Kounev, S., Krogmann, K.: Automated extraction of palladio component models from running enterprise java applications. In: VALUETOOLS 2009, pp. 1–10 (2009)
7. Wang, W., et al.: Application-level CPU consumption estimation: towards performance isolation of multi-tenancy web applications. In: IEEE CLOUD 2012, pp. 439–446, June 2012
8. Zheng, T., Woodside, C., Litoiu, M.: Performance model estimation and tracking using optimal filters. IEEE TSE **34**(3), 391–406 (2008)
9. Spinner, S., Casale, G., Zhu, X., Kounev, S.: Librede: a library for resource demand estimation. In: ACM/SPEC ICPE 2014, pp. 227–228. ACM, New York (2014)
10. Grohmann, J., Herbst, N., Spinner, S., Kounev, S.: Self-tuning resource demand estimation. In: Proceedings of the 14th IEEE International Conference on Autonomic Computing (ICAC 2017), July 2017

11. Bolch, G., et al.: Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications. John Wiley & Sons, New York (2006)
12. Bunch, J.R., Hopcroft, J.E.: Triangular factorization and inversion by fast matrix multiplication. Math. Comput. **28**(125), 231–236 (1974)
13. Herbst, N., Kounev, S., Weber, A., Groenda, H.: BUNGEE: an elasticity benchmark for self-adaptive IaaS cloud environments. In: SEAMS 2015, pp. 46–56. IEEE Press (2015)
14. Herbst, N., et al.: Ready for Rain? A View from SPEC Research on the Future of Cloud Metrics. CoRR abs/1604.03470 (2016)
15. Iosup, A., Yigitbasi, N., Epema, D.: On the performance variability of production cloud services. In: CCGrid 2011, pp. 104–113 (2011)
16. Jennings, B., Stadler, R.: Resource management in clouds: survey and research challenges. J. Netw. Syst. Manag. **23**(3), 567–619 (2015)
17. Lorido-Botran, T., Miguel-Alonso, J., Lozano, J.A.: A review of auto-scaling techniques for elastic applications in cloud environments. J. Grid Comput. **12**(4), 559–592 (2014)
18. Han, R., Guo, L., et al.: Lightweight resource scaling for cloud applications. In: IEEE/ACM CCGrid 2012, pp. 644–651. IEEE (2012)
19. Maurer, M., Brandic, I., Sakellariou, R.: Enacting SLAs in clouds using rules. In: Jeannot, E., Namyst, R., Roman, J. (eds.) Euro-Par 2011, Part I. LNCS, vol. 6852, pp. 455–466. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23400-2_42
20. Urgaonkar, B., et al.: Agile dynamic provisioning of multi-tier internet applications. ACM TAAS **3**(1), 1 (2008)
21. Zhang, Q., Cherkasova, L., Smirni, E.: A regression-based analytic model for dynamic resource provisioning of multi-tier applications. In: IEEE ICAC 2007, p. 27. IEEE (2007)
22. Kalyvianaki, E., Charalambous, T., Hand, S.: Self-adaptive and self-configured CPU resource provisioning for virtualized servers using Kalman filters. In: ACM ICAC 2009, pp. 117–126. ACM (2009)
23. Ali-Eldin, A., Tordsson, J., Elmroth, E.: An adaptive hybrid elasticity controller for cloud infrastructures. In: IEEE NOMS 2012, pp. 204–212. IEEE (2012)
24. Tesauro, G., Jong, N.K., Das, R., Bennani, M.N.: A hybrid reinforcement learning approach to autonomic resource allocation. In: IEEE ICAC 2006, pp. 65–73. IEEE (2006)
25. Rao, J., et al.: VCONF: a reinforcement learning approach to virtual machines auto-configuration. In: ACM ICAC 2009, pp. 137–146. ACM (2009)
26. Iqbal, W., Dailey, M.N., Carrera, D., Janecek, P.: Adaptive resource provisioning for read intensive multi-tier applications in the cloud. Futur. Gener. Comput. Syst. **27**(6), 871–879 (2011)
27. Chen, G., et al.: Energy-aware server provisioning and load dispatching for connection-intensive internet services. In: NSDI, vol. 8, pp. 337–350 (2008)
28. Shen, Z., Subbiah, S., Gu, X., Wilkes, J.: Cloudscale: elastic resource scaling for multi-tenant cloud systems. In: ACM Symposium on Cloud Computing. ACM (2011)
29. Nguyen, H., et al.: Agile: elastic distributed resource scaling for infrastructure-as-a-service. In: ICAC, vol. 13, pp. 69–82 (2013)
30. Spinner, S., et al.: Runtime vertical scaling of virtualized applications via online model estimation. In: IEEE SASO 2014, pp. 157–166. IEEE (2014)

Evaluating a Single-Server Queue with Asynchronous Speed Scaling

Alexander Rumyantsev[1,2(✉)], Polina Zueva[2], Ksenia Kalinina[1,2], and Alexander Golovin[1]

[1] Karelian Research Centre of RAS, Institute of Applied Mathematical Research, 11 Pushkinskaya Street, Petrozavodsk, Russia
ar0@krc.karelia.ru

[2] Petrozavodsk State University, 33 Lenina Pr., Petrozavodsk, Russia

Abstract. We introduce an approach to energy efficiency policies evaluation in various application fields, based on widely available technical and software tools. The approach is based on a simple client-server-type application run on a single linux-operated laptop, equipped with standard frequency scaling tools. We validate the approach by analyzing two energy efficiency policies: the celebrated hysteretic control and the randomized switching (introduced recently) in a single-server queue. Explicit analytical results are obtained by means of Matrix-Analytic method, and a simulation model based on discrete event stochastic simulation (a particular case of the generalized Kiefer–Wolfowitz stochastic recursion introduced recently) allows to obtain performance and energy estimates, when analytical results are inappropriate. The results of real-world experiments are introduced, and applicability of the approach is discussed.

Keywords: Energy efficiency · Hysteretic control
Randomized switching · Internet of Things · Matrix-Analytic method
Single-server queue · Frequency scaling · Linux

1 Introduction

Due to an increasing demand for energy saving, researchers and developers are interested in energy-aware computer and communication networks, systems of high-performance, distributed and fog computing. However, basically energy consumption should be considered along with performance costs. Various possibilities to control energy efficiency of the system have been analyzed in recent research. The widely-used state-dependent switching policy allows to reduce power consumption [1]. In [12], a single-server system with two sojourn time dependent service rates is considered. The optimal sleep-state control of a single energy-aware server is designed in [6]. To analyze energy-efficiency of server clusters the performance and energy consumption of two load balancing policies (Round-Robin and Join the Shortest Queue) are compared in [5]. The main results of applying traffic-oblivious policy for optimality analysis of

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 157–172, 2018.
https://doi.org/10.1007/978-3-319-74947-1_11

energy-performance trade-off for a server farm are presented in [4]. Basically the optimization problems are solved, which are either unconstrained with a single (e.g. product-form) criterion, or a constrained optimization (e.g. with limited QoS degradation).

For some classes of policies analytical results can be obtained. E.g. for a single-server queue with state-dependent service rate based on the amount of work right after customer arrivals, the steady-state workload distribution at arbitrary epochs was obtained for the two-threshold policy [1]. In [14] for a single-server queue with two service rates and randomized switching discipline, the analytical results were obtained by means of Matrix-Analytic method.

At that, when it is difficult (or even impossible) to construct an analytical model, the control policy can be evaluated with the help of simulation model (see, e.g. [5,6,10]). One of the well-known ways to construct such a model is by using Lindley-type recursion [1,12] and modified Kiefer–Wolfowitz-type stochastic recursion [13].

It is even more reasonable to evaluate the control policy with the help of technical model. It is useful to design this model e.g. for energy efficiency evaluation based on measuring the power consumption of a laptop (the battery level) or the CPU frequency (see, e.g. [7]), Apache web server performance evaluation [2], or linux-based high-performance server energy efficiency study [3]. It is important to be able to compare the numerical results that can we get using simulation model to the values that can be calculated with the help of technical model, like it is done in the recent work [11], where authors predict the power consumption of workloads for different load levels in virtualized environments and evaluate their approach comparing predicted results against measurements of power consumption for various configurations on a target server.

The purpose of this paper is to introduce a common unified framework for performing experiments with various energy efficiency policies. Motivated by an idea to combine all the types of models mentioned below (analytical, simulation and technical) for control policies performance evaluation, we introduce an approach that does not require specific technical and software tools. It is based on a simple client-server-type application, a Linux-operated laptop and standard frequency scaling tools. We illustrate our approach by considering the two energy efficiency policies: the widely used hysteretic control and the randomized switching (described in details below). For the latter policy we use the explicit analytical results presented in [14]. In case when these results are inappropriate, our approach provides a simulation model (based on discrete event stochastic simulation) that allows to get the estimates for energy and performance characteristics.

To validate this framework, we consider a single-server queueing system fed with a Poisson input (with input rate $\lambda > 0$) of customers, each having an (exponentially distributed) iid amount of work to be processed. The server can process work at two speeds, $r_L < r_H$ units of work per unit time (referred as *low* and *high* below, respectively). Motivated by practical applications, we assume that the server has four distinct values of energy consumption $e_{0,L} \leqslant e_{0,H} < e_L < e_H$, related to being idle (at low/high speed), working at low and

high speed, respectively. In the sequel we assume, that the server is capable of switching the speed asynchronously, only at customer arrival/departure epochs, implementing one of the following energy efficiency policies:

- two level hysteretic control based on the queue size at switching time instant (referred below as *hysteretic control, HC*): if the queue size at arrival (departure) epoch exceeds (falls below) some fixed level $k_H \geqslant 0$ ($k_L + 1 \geqslant 0$), the server switches to the speed r_H (r_L, respectively);
- blind randomized switching introduced in [14] (referred below as *randomized switching, RS*): switching at arrival (departure) epoch to r_H (r_L) is governed by an independent Bernoulli trial with success probability p_H (p_L, respectively).

Note that in particular cases the HC policy represents a system with single threshold control (if $H = L$), as well as the system working at a single fixed speed (if $H = L = 0$, or $H = L = \infty$). At that, the RS policy does not induce queue size control, which allows to implement the policy in various practical applications, including small IoT nodes (in which the battery restrictions and simplified logic significantly complicate tracking of the queue size), and web services under high load conditions (when queue size tracking significantly reduces the performance of the system).

Below we consider the described system in stationary mode, and discuss the stability conditions for HC and RS policies separately. Note that whereas the theoretical results are valid for the whole stability region of the system, in practical experiments we primarily discuss the most interesting case when the system at speed r_L becomes unstable (and cannot cope with the given load).

This paper is organized as follows. In Sect. 2 the analytical models related to HC and RS policies of a single-server queue are introduced, and the performance measures are derived explicitly. In Sect. 3 the appropriate simulation model for stochastic simulation is presented, that allows to cover more general cases, where analytical results are not available. In Sect. 4 we introduce the technical model that implements the HC and RS policies, which is based on the client-server-type system on a linux-operated laptop. We discuss the results of experiments in Sect. 5. The conclusions and discussion of the framework, as well as limitations of the approach are given in Sect. 6.

2 Stochastic Models

In this section we discuss the available analytical results which, in turn, allow to validate the introduced framework. First we give some common notation used in the sequel.

We consider a single-server queue with a renewal input of customers arriving at epochs $t_i, i = 1, 2, \ldots$ into a FCFS-type unbounded queue, with iid interarrival times $T_i := t_{i+1} - t_i, i \geqslant 1$. Customer i requires to attain S_i units of service, which, however, is unknown before the service starting epoch. For simplicity we assume T, S to be exponentially distributed (where the r.v. without subindex is

a generic r.v. of a sequence). Recall that $\lambda = 1/\mathsf{E}T$ is the input rate, whereas w.l.o.g. we assume $\mathsf{E}S = 1$. Then the server load equals $\rho_L := \lambda/r_L$ ($\rho_H := \lambda/r_H$) for a system working at higher (lower) speed with a degenerate switching policy. We also denote by $V(t) \geqslant 0$ the number of customers in the system at time $t-$, and by $R(t) \in \{r_L, r_H\}$ we denote the frequency at $t-$. The continuous-time Markov process

$$\{X(t) := (V(t), R(t)), \quad t \geqslant 0\} \tag{1}$$

is the so-called Quasi-Birth-Death process (for details on this process see [8]), where the *level* component $V(t)$ is increased/decreased by at most one at a time, and there are two possible values for *phase* component $R(t)$ at each level. The infinitesimal generator Q of the process (with instantaneous transition rates) has the following block-tridiagonal form

$$Q = \begin{pmatrix} A^{0,0} & A^{0,1} & 0 & 0 & \dots \\ A^{1,0} & A^{1,1} & A_0 & 0 & \dots \\ 0 & A_2 & A_1 & A_0 & \ddots \\ 0 & 0 & A_2 & A_1 & \ddots \\ 0 & 0 & \ddots & \ddots & \ddots \end{pmatrix}, \tag{2}$$

where the (not necessarily square) matrices $A^{k,l}, k, l = 0, 1$ correspond to boundary states, and square matrices A_i of order 2 correspond to non-boundary states, $i = 0, 1, 2$.

The matrix Q defines a stable process if the celebrated Neuts ergodicity condition holds [8,15]:

$$\alpha A_0 \mathbf{1} < \alpha A_2 \mathbf{1}, \tag{3}$$

where the vector α (describing the phase distribution at high levels [8]) is the solution of the following system

$$\begin{cases} \alpha(A_0 + A_1 + A_2) = 0, \\ \alpha \mathbf{1} = 1, \end{cases} \tag{4}$$

and $\mathbf{1}$ is the vector of ones. Given the stability condition (3) holds, the stationary probability vector π (such that $\pi Q = 0$ and $\pi \mathbf{1} = 1$) partitioned as $\pi = (\pi_0, \pi_1, \dots)$ may be found as the following matrix-geometric solution:

$$\pi_i = \pi_1 R^{i-1}, i \geqslant 2, \tag{5}$$

where the matrix R is the minimal nonnegative solution of the matrix polynomial equation

$$R^2 A_2 + R A_1 + A_0 = \mathbf{0}, \tag{6}$$

the matrix $\mathbf{0}$ is a square matrix of order 2, and vectors π_0, π_1 are the solution of boundary conditions system (see e.g. [8])

$$(\pi_0, \pi_1) \begin{pmatrix} A^{0,0} & A^{0,1} \\ A^{1,0} & A^{1,1} + R A_2 \end{pmatrix} = 0, \tag{7}$$

$$\pi_0 \mathbf{1} + \pi_1 (I - R)^{-1} \mathbf{1} = 1. \tag{8}$$

Then it is easy to define the following key performance measures of the system: the mean stationary number of customers in the system [8]

$$\mathcal{V} = \pi_1 (I - R)^{-2} \mathbf{1}, \tag{9}$$

and the average stationary energy consumption

$$\mathcal{E} = \pi_0 \begin{pmatrix} e_{0,L} \\ e_{0,H} \end{pmatrix} + \pi_1 (I - R)^{-1} \begin{pmatrix} e_L \\ e_H \end{pmatrix}. \tag{10}$$

The server aims to minimize the average energy consumption by switching the speed according to the control policy.

It is well known, that the performance measures for system working at a constant speed r_j may be found as follows:

$$\mathcal{V}_j = \rho_j (1 - \rho_j)^{-1}, \quad \mathcal{E}_j = \rho_j e_j + (1 - \rho_j) e_0, \quad j = \{L, H\}. \tag{11}$$

If possible, one has to optimize the values k_H, k_L (p_H, p_L) to obtain the energy consumption less than $\mathcal{E}_H$, provided a controlled decrease of quality of service (mean stationary number of customers) above $\mathcal{V}_H$.

2.1 Hysteretic Control

In case of HC policy, the matrices in Q are defined as follows:

$$A_0 = \begin{pmatrix} 0 & 0 \\ 0 & \lambda \end{pmatrix}, \quad A_1 = \begin{pmatrix} 0 & 0 \\ 0 & -\lambda - r_H \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & 0 \\ 0 & r_H \end{pmatrix}, \tag{12}$$

which means, that the system is working at speed r_H when the number of customers exceeds k_H. The matrix $A^{0,0}$ is itself a block-tridiagonal square matrix of order $2(k_H + 1)$ with a special structure

$$A^{0,0} = \left(A^0_{i,j}\right), \quad i = 0, \ldots, k_H, j \in \{\max(i-1, 0), i, \min(i+1, k_H)\}, \tag{13}$$

filled with zeroes outside the blocks.

The square 2×2 matrices $A^0_{i,j}$ are defined as follows:

$$A^0_{i,i+1} = \begin{pmatrix} \lambda 1_{i<k_H} & \lambda 1_{i=k_H} \\ 0 & \lambda 1_{i \geqslant k_L+1} \end{pmatrix}, \quad A^0_{i+1,i} = \begin{pmatrix} r_L & 0 \\ r_H 1_{i=k_L} & r_H 1_{i>k_L} \end{pmatrix}, \tag{14}$$

$$A^0_{i,i} = \begin{pmatrix} -\lambda - r_L 1_{i>0} & 0 \\ 0 & -(\lambda + r_H) 1_{i \geqslant k_L+1} \end{pmatrix}, \quad i = 0, \ldots, k_H, \tag{15}$$

where 1_A equals 1 if A holds. It remains to note, that the matrix $A^{0,1}$ is a $2(k_H + 1) \times 2$ matrix, and $A^{1,0}$ is a $2 \times 2(k_H + 1)$ matrix, and $A^{1,1}$ is a square matrix of order 2 with the following structure

$$A^{1,0} = (0 \;\; A_2), \quad A^{1,1} = A_1, \quad A^{0,1} = (0 \;\; A^0_{k_H, k_{H+1}})', \tag{16}$$

where $(\cdot)'$ is the transpose operation. Note that the matrix $A^{0,1}$ defines the switching to speed r_H, whereas $A^0_{k_L+1,k_L}$ defines switching to speed r_L.

Note that stability condition (3) reduces to stability at the higher speed, that is $\rho_H < 1$. Moreover, it is easy to check, that the solution of (6) is as follows

$$R = \begin{pmatrix} 0 & 0 \\ 0 & \rho_2 \end{pmatrix}. \tag{17}$$

Provided stability holds, the stationary probabilities can be obtained. Moreover, the special structure of matrix $A^{0,0}$ allows to transform the system (8) and solve it iteratively to derive $\pi_t, t \geqslant 0$ by the following algorithm (which may be obtained by sequential substitution). (Note that for ease of interpretation, we enumerate the components of $2k_H+2$-component vector π_0 as $\pi^j_{0,i}$, $i = 0, \ldots, k_H$ and $j \in \{L, H\}$.)

Step 1. $\pi^H_{0,i} = \left(1 - \rho_2^{i-k_L}\right)\left(1 - \rho_2^{k_H-k_L+1}\right)^{-1}, \quad k_L < i \leqslant k_H.$

Step 2. $\pi^L_{0,i} = \pi^H_{0,k_L+1}(\rho_1^{i-k_H} - \rho_1)/(\rho_2 - \rho_1\rho_2), \quad k_L < i \leqslant k_H.$

Step 3. $\pi^L_{0,i} = \rho_1^{i-k_L-1}\left(\pi^L_{0,k_L+1} + \rho_1(1-\rho_2)\left[\rho_2 - \rho_2^{k_H-k_L+2}\right]^{-1}\right), \; i \leqslant k_L.$

Step 4. From the balance equation, obtain

$$\pi^H_{0,k_H+1} = \left[1 + \sum_{i=0}^{k_H} \pi^L_{0,i} + \sum_{i=k_L+1}^{k_H} \pi^H_{0,i} + \rho_2(1-\rho_2)^{-1}\right]^{-1}.$$

Step 5. Recalculate $\pi^j_{0,i} := \pi^j_{0,i}\pi^H_{0,k_H+1}$ for $j \in \{L, H\}$ and $i \neq k_H + 1$.

Step 6. Obtain $\pi^H_t = \rho_2^t \pi^H_{0,k_H+1}, \quad t \geqslant 1$, and recall $\pi^L_t = 0$.

For notational convenience, below we fill the vector π_0 with zeroes, if the corresponding position was not defined in the algorithm. Then we obtain the following performance measures of interest (denoting them by superscript HC)

$$\mathcal{V}^{HC} = \sum_{i=1}^{k_H} i(\pi^L_{0,i} + \pi^H_{0,i}) + \pi^H_{0,k_H+1} \sum_{i \geqslant 0} (i + k_H)\rho_2{}^i \tag{18}$$

$$\mathcal{E}^{HC} = e_0\pi^L_{0,0} + \sum_{i=1}^{k_H}(e_L\pi^L_{0,i} + e_H\pi^H_{0,i}) + e_H\pi^H_{0,k_H+1}\frac{\rho_2}{1-\rho_2}. \tag{19}$$

Note that the Eqs. (9) and (10) need to be appropriately modified due to special structure of matrix $A^{0,0}$ and vector π_0. Namely, the modified Eq. (9) uses the fact, that only the component $\pi^L_{0,0}$ corresponds to the idle system, whereas $\pi^j_{0,i}$ is the stationary probability of having i customers in the system and processing them at the speed r_j, $j = \{L, H\}$, which gives (18). At that, the values $e_{0,L}$ and $e_{0,H}$ from (10) are the vectors of corresponding dimension, with $e_{0,L} = (e_0, e_L, \ldots, e_L)$ and $e_{0,H} = (e_H, \ldots, e_H)$.

2.2 Randomized Switching

In this section we discuss the RS policy and, due to space limitation, we briefly recall the results presented in [14], where the necessary details can be found. The matrices constituting the generator Q are defined as follows:

$$A_0 = \begin{pmatrix} (1-p_H)\lambda & p_H\lambda \\ 0 & \lambda \end{pmatrix}, \tag{20}$$

$$A_1 = \begin{pmatrix} -\lambda - r_L & 0 \\ 0 & -\lambda - r_H \end{pmatrix}, \tag{21}$$

$$A_2 = \begin{pmatrix} r_L & 0 \\ p_L r_H & (1-p_L) r_H \end{pmatrix}, \tag{22}$$

$$A^{0,0} = -\lambda I, \quad A^{0,1} = A_0, \tag{23}$$

$$A^{1,1} = A_1, \quad A^{1,0} = A_2. \tag{24}$$

Thus, the stability condition is

$$\lambda p_H(\lambda - r_H) + r_H p_L(\lambda - r_L) < 0. \tag{25}$$

Intuitively, the condition (25) indicates a negative drift of the service process of the system under heavy load, with respect to the mode switching intensity.

Despite the fact, that A_0 and A_2 are in general full rank matrices, the matrix R can be found explicitly by the following algorithm (see [14]).

1. Define a determinantal polynomial $\det(A(\xi)) := \det(A_0 + \xi A_1 + \xi^2 A_2)$.
2. Using trigonometric solution, obtain the greatest root ξ_3 of the third degree polynomial $\det(A(\xi))/(\xi - 1) = a_3\xi^3 + a_2\xi^2 + a_1\xi + a_0$, with roots known to be real.
3. Find $b_0 = -a_0/(a_3\xi_3), b_1 = a_2/a_3 + \xi_3$.
4. Find R as follows:

$$R = [b_0 A_2 - A_0][A_1 - b_1 A_2]^{-1}.$$

Using the boundary conditions (8), it is easy to obtain π_1 as follows [14]:

$$\begin{cases} \pi_1 \left(\lambda^{-1} A_2 - R^{-1}\right) A_0 \mathbf{1} & = 0, \\ \pi_1 \left(\lambda^{-1} A_2 + (I - R)^{-1}\right) \mathbf{1} & = 1. \end{cases} \tag{26}$$

Then the value π_0 is obtained as follows [14]:

$$\pi_0 = \lambda^{-1} \pi_1 A_2. \tag{27}$$

Then the performance measures $\mathcal{V}^{RS}, \mathcal{E}^{RS}$ may be obtained from (9) and (10).

3 Simulation Model

To obtain the simulation model, we use the components of a stochastic recursion at arrival/departure time epochs, introduced recently in [18], which is a generalization of the celebrated Kiefer–Wolfowitz recursion. First, we define the key time epochs T_i, which are the arrival and departure epochs. At each such time epoch T_i we define the system state as a tuple $(M_i; I_i^A; I_i^D; B_i(j), j \in M_i; R_i)$, that is, the consecutive numbers of customers present in the system, indicators that the time epoch is an arrival (departure), the remaining work of each customer in the system, and the service speed, respectively.

The epoch T_{i+1} is recursively defined as follows

$$T_{i+1} = \min\left\{t_{A(T_i)+1}, T_i + B_i(j)/R_i\right\}, \quad j = \min M_i, \tag{28}$$

where the counting process $A(t) = k$, $t_k \leqslant t < t_{k+1}$, $k \geqslant 1$ is the number of arrivals up to time t; and $T_i + B_i(j)/R_i$, $j = \min M_i$ is the potential departure time of the task being served (if any) at instant T_i. Recurrent relations $B_i(j), R_i$ are defined below, and (if necessary), we add a superscript to indicate the control policy.

The set of numbers of customers present in the system at T_{i+1} is changed at key epochs as follows

$$M_{i+1} = M_i \cup \{A(T_{i+1}) : I_{i+1}^A = 1\} \backslash \{\min M_i : I_{i+1}^D = 1\}, \tag{29}$$

with obvious conventions $\min \varnothing = \varnothing$. The remaining work at T_{i+1} is defined as follows

$$B_{i+1}(j) = \begin{cases} B_i(j) - (T_{i+1} - T_i)R_i, & j = \min M_i \cap M_{i+1}, \\ S_j, & j \in M_{i+1} \setminus \min M_i, \end{cases} \tag{30}$$

where $B_{i+1}(j)$ is nothing, if $i+1$ is the departure epoch of task j. For the HC policy, the speed R_{i+1}^{HC} at time T_{i+1} is given by recurrent relation

$$R_{i+1}^{HC} = I_{i+1}^D(r_L 1_{|M_i|=k_L+1} + R_i^{HC} 1_{|M_i|\neq k_L+1}) + I_{i+1}^A(r_H 1_{|M_i|=k_H} + R_i^{HC} 1_{|M_i|\neq k_H}).$$

For the RS policy, the speed R_{i+1}^{RS} is given as follows

$$R_{i+1}^{RS} = I_{i+1}^D(r_L \beta_{p_L} + R_i^{RS}(1-\beta_{p_L})) + I_{i+1}^A(r_H \beta_{p_H} + R_i^{RS}(1-\beta_{p_H})),$$

where β_x is an independent Bernoulli trial with success probability x. It remains to define the recursion base. Since at $T_1 = 0$ an arrival of the first customer occurs, then

$$M_1 = \{1\},\ B_1 = \{S_1\},\ R_1 = r_L.$$

Now the performance of the model may be obtained as

$$\overline{\mathcal{V}} = \frac{1}{T}\sum_{i:T_i \leqslant T} (T_{i+1} - T_i)|M_i|, \quad \overline{\mathcal{E}} = \frac{1}{T}\sum_{i:T_i \leqslant T} (T_{i+1} - T_i)(e_{0,R_i} 1_{|M_i|=0} + e_{R_i} 1_{|M_i|>0}).$$

For simulation purpose, the recurrent relations were implemented in R language [17].

4 Technical Model

In this section we present a technical model, that implements the control policies discussed in Sect. 2. We use the simulation model described in Sect. 3 together with analytical results given in Sect. 2 to validate the model. We note, that both the technical model and the simulation model allow to evaluate the performance of control policy when the analytical results are not available. For technical clarification, in the sequel we will use the term *frequency* as a synonym of the *speed*, and *client* as a synonym of *customer* used in previous sections.

We give an overview of the technical model. A prepared Linux-operated battery-powered laptop is used to evaluate the client-server web application (web service) that provides a simple, but CPU-intensive workload (generating and/or compressing a pseudo random sequence). The client initiates the service by sending a request to the server, with a random file (generated on the server side) of exponentially-distributed random size, that is to be compressed. The iid. interarrival times have exponential distribution. The Apache-based server has specific settings that allow to serve only one client at a time. Thus, other clients have to wait in a queue (indeed, it is the so-called TCP backlog, where the waiting clients are the TCP connections in unacknowledged state). Based on the queue size (in case of HC policy), or on an independent Bernoulli trial with a given success probability (in case of RS policy), the server switches one of the frequencies, performing CPU frequency scaling by means of the `cpufreq` subsystem [9]. The server is able to monitor the queue size at arrival and departure epochs. At each such a key time epoch, the server records the battery level (mAh/mWh), queue size (number of unacknowledged TCP connections) and current time (with nanosecond precision). The asynchronous switching mode is performed by means of file synchronization with the `incron` subsystem.

4.1 General Linux Settings

The Linux operation system is enhanced with a frequency scaling `cpufreq` subsystem [9,16] operated by various user- and system-level tools. This tools allow to set various frequency scaling policies using the so-called governors and a corresponding daemon process. The dynamic governors allow to adopt the frequency based on the value of the current load to save the power and/or increase performance. Moreover, these governors allow to customize the policy of frequency scaling. There are several in-kernel governors available for use with the `cpufreq` subsystem:

performance statically switches to the highest frequency available;
powersave statically switches to the lowest available frequency;
userspace allows the user (with appropriate privileges) to switch the frequency manually;
ondemand dynamically changes the frequency based on processor utilization;
conservative dynamically adjusts frequencies based on processor utilization, with a gradual frequency increase.

To use the `userspace` governor, it is necessary to switch to the `acpi-cpufreq` driver, which requires to disable the `intel_pstate` in the kernel. This may be done e.g. by adding the optional kernel parameter into the GRUB bootloader commandline between the `splash` and `quiet` parameters, e.g.

```
GRUB_CMDLINE_LINUX_DEFAULT= "resume=/dev/sda1 splash=silent \
intel_pstate=disable quiet showopts"
```

It is necessary to update the bootloader afterwards, e.g. by `update-bootloader` command. Note that the `userspace` governor is able to interact with many daemons (such as `cpudyn`, `cpufreqd`, `powernowd`) to control, set or change the CPU frequencies. We selected the widely available across Linux distribution `cpupower` tools package to perform the frequency scaling.

The following additional software packages should be installed (we do not give the exact names, since they are distribution specific): the Apache webserver with `apache-mod_php5`, the `cpupower` tools, the `inotify-tools` package with `incron` daemon, and `ss` socket investigation utility. Specific settings for these tools will be discussed below.

To perform experiments with high load, it may be necessary to increase the system `backlog` size (e.g. to 10000) which may be done by including the following lines in`/etc/sysctl.conf` file:

```
net.ipv4.tcp_max_syn_backlog=10000
net.core.netdev_max_backlog=10000
```

Note however, that it may be required to perform other distribution specific settings that limit the backlog size.

It is recommended also either to use a singlemode regime, or to disable the X Window system, e.g. by executing the `init 3` command in the `root` shell. Note that the `incron` daemon should be started manually at init level 3.

4.2 Apache Configuration

To perform as a single-server queue, the Apache webserver needs to be specifically set up. First, it is necessary to enable the `mpm-prefork` Apache module (e.g. by the `a2enmod` command) and disable the `worker` module. Then, it may be required to disable the KeepAlive TCP connection setting, so as to instantly close the connection after service completion epoch, and correct the `backlog` value. The location of Apache configuration is distribution specific (e.g. `/etc/apache2/`), and the following line needs to be added into configuration (e.g. in the file `server-tuning.conf`):

```
KeepAlive Off
ListenBackLog 10000
```

Additinally, the following parameters should be added in the configuration of `prefork` module (in a conditional section `<IfModule prefork.c>... </IfModule>`:

```
ServerLimit 1
MaxClients 1
MaxRequestWorkers 1
MaxConnectionsPerChild 0
```

These parameters make Apache serve the requests as a single-server FIFO queue.

4.3 Incron Configuration

To use the asynchronous switching at arrival/departure times, the `incron` subsystem is used. The `incron` daemon allows to monitor the file system events and trigger the shell script execution. Each time a client generates a request (a server completes the service), it creates an empty file, and, as a result, the appropriate shell script executes the control policy. To create the triggers, the following lines should be added into `incrontab` of the `root` user e.g. by `incrontab -e` command

```
<directory_location> IN_CREATE /bin/bash <shell_script_location>
```

The templates `<directory_location` and `shell_script_location` should be changed to the location of a directory where the client (server) writes the empty file at request initiation (service completion) event, and to the appropriate script executing the control policy at arrival (departure). Note that the `incron` uses the `inode` rather than the directory name to monitor the events, and it is necessary to recreate the triggers in case of directory deletion/creation. Recall also that `incron` should be started manually at init level 3.

4.4 Frequency Management

To perform the frequency scaling, we use the `cpupower` command with appropriate parameter `<desired_frequency>`, i.e.

```
cpupower frequency-set -f <desired_frequency>
```

This command is executed in the shell script triggered by `incron` at arrival/departure instants.

4.5 Measurements and Trace

Due to the asynchronous management mode, the most convenient way of tracing the system state is to perform instant measurements in the script triggered at arrival/departure instants by `incron` daemon. The appropriate system parameters, including the queue size, battery level, frequency and current time (with nanosecond precision) is performed at the management instants.

Queue Size. To implement the control policy, the shell script triggered by `incron` is to be aware of the queue size (for HC policy). The queue size is measured as the number of unacknowledged TCP connections persisting in backlog. The appropriate tool for measurements is the `ss -lti` command, the output of which is afterwards parsed as follows

```
q=$(ss -lti '( sport= :http)' | grep unacked)
q=$(expr "$q" : '.*unacked:\([0-9]*\)')
```

Frequency. The current CPU frequency may be obtained by `cpupower` command as follows

```
cpu-power frequency-info -f | tail -n 1
```

Note that the output may be distribution specific and may require additional effort to extract the number e.g. by means of

Energy Consumption. A laptop is in general equipped with a battery, and the information on the remaining capacity may be extracted directly from the system device. Note that the location of the data, as well as the units of measurement, are distribution- and laptop-specific. E.g. the command for battery level extraction is as follows:

```
cat /sys/class/power_supply/BAT0/charge_now
```

Time. The time measurements with nanosecond precision are performed by the following standard shell routine

```
date +%s%N
```

4.6 Client Application

A simple client application is the shell script, that receives a specifically structured input file with two-column table defining the exponentially-distributed interarrival times and the sizes of random files to generate (compress). The HTTP request to the Apache server at localhost is performed by a `curl` command, by the following command

```
curl http://localhost/index.php?count=$var &
```

where `$var` is the random request size (amount of work). Note that the request is done in asynchronous mode, so as to allow to perform another request without waiting for the current request completion. Afterwards, a `sleep` command is initiated, to wait for the interarrival time before sending the next request. Note that the same application creates an empty file in a specific location to trigger the control policy.

4.7 Server Load

For the experiment we need to emulate a web service that provides CPU-intensive application. As a model we assume the random file generation (compression) service. The following command

```
dd if=/dev/urandom iflag=fullblock bs=<random_size> count=1 \
| gzip > /dev/null
```

allows to compress a pseudo randomly generated file of size `<random_size>`. Note that it is not necessary to compress the file, since generating a pseudorandom sequence itself is a CPU-frequency-dependent operation. However, a higher load may be performed when the `gzip` command is used. Moreover, due to the change of sequence generation algorithm for the `/dev/urandom` device in Linux kernel $\geqslant 4.8$, the usage of `gzip` may be necessary. However, in the numerical experiment we used a laptop with an older kernel, which allowed to directly read the `dev/urandom` device from a PHP5 application as follows:

```
$fp=@fopen('/dev/urandom','rb');
if($fp !== FALSE){
    $result=@fread($fp,$_GET['count']);
    @fclose($fp);}
```

Note, that to avoid access restrictions, it is recommended to use PHP7 with `random_bytes` command.

4.8 Calibration of the Model

The technical model calibration should include measuring the time of random file generation, e.g. by `time` command, measurements of energy consumption at idle, maximum and minimum available frequencies. Some additional measurements may be required to estimates the latency of client, server, and monitoring applications.

5 Numerical Experiments

An HP EliteBook 2760p laptop with SSD drive was used for experiments, equipped with OpenSUSE Leap linux with kernel `4.1.36-44-default x86_64`. The following calibration results were obtained:

- Minimum frequency (used for experiments): 1.2 GHz;
- Maximum frequency (used for experiments): 2.6 GHz;
- Energy consumption: 165, 330, 550 mAh;
- Service speed: generating a 100MB random sequence in 14.4 (6.67) s at minimum (maximum) frequency.

Additional validation experiments were performed by Dell Vostro laptop equipped with Ubuntu linux with kernel `4.8`. However, the results of experiments agree with the ones performed by HP, and we do not discuss them below.

First, we performed a generation and validation of the framework for the HC policy with $k_L = 5, k_H = 10$, $r_L = 0.46, r_H = 1, \lambda = 0.6$ clients per second (where the service speeds are adopted from calibration results for notational convenience). We used 10000 client requests to validate the framework. The average number of customers in the system obtained is relatively close, which is also illustrated by the proximity of the (theoretical, simulated and measured) discrete distributions of mean number of customers (client requests) in the system, see Fig. 1. Note that various input conditions were evaluated, which in general give the same qualitative results, and thus we give the figures for illustration purpose only.

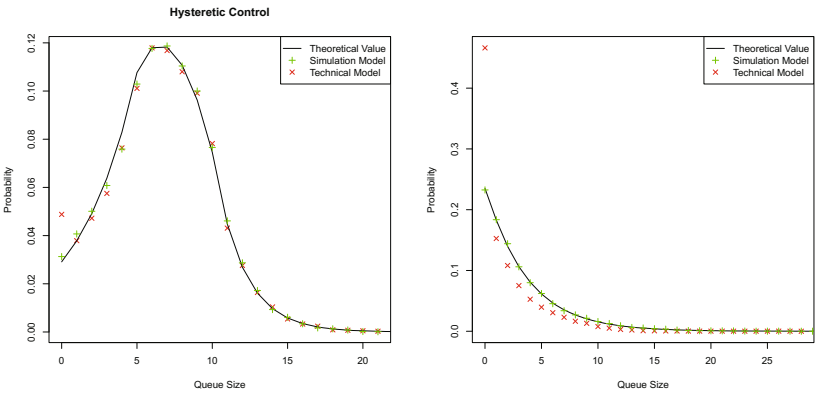


Fig. 1. Control policy evaluation: discrete distribution of the number of clients in the system for analytical, simulation and numerical results. Left: HC policy, $k_L = 5, k_H = 10$, $r_L = 0.46, r_H = 1, \lambda = 0.6$, 10000 clients. Right: RS policy, $p_H = 0.7, p_L = 0.4, \lambda = 6, r_L = 4.62, r_H = 10$, 50000 clients.

Next, we evaluated the RS policy with the following settings: $p_H = 0.7, p_L = 0.4, \lambda = 6, r_L = 4.62, r_H = 10$ clients per second, with 50000 client requests. The obtained proximity is shown on Fig. 1. Note however, that the distribution given by technical model is relatively shifted to lower values. This effect may be considerably reduced by allowing the mean service time to be higher, however, resulting in a longer simulation runs.

Note that in general the energy consumption obtained by experiments agreed with the one provided by simulation model and analytical results. However, we recommend to perform additional tuning by running several calibration experiments with random data and performing parameter fitting.

6 Conclusion and Discussion

We presented a framework that allows to validate the three components of energy efficiency/performance evaluation of various control policies. The results of experiments illustrated the applicability of the approach. However, we stress that it may be desirable to use specific devices for various fields of application, e.g. a Raspberry Pi platform may be used as a technical model of the Internet-of-Things device. Also extensive parameter tuning may be desirable, including usage of special devices for energy consumption management (since the accuracy of battery capacity level may be not enough), and reducing the system noise (e.g. switching to singlemode etc.). Note also, that the calibration phase is kernel dependent.

The experiments presented in the paper were designed to validate that the analytical, simulation and technical models agree, and thus were performed in the region, where analytical results were obtained. Namely, the exponentially distributed service times, and Poisson arrivals were used to generate the sequences both for simulation, and for technical model. However, these assumptions seem to be restrictive and hardly comply with the real workload data. Nevertheless, the good agreement of the three models allows to extend the region of model evaluation (by means of simulation), which might give some new insights. Note also, that the structure of the framework allows to modify the configuration in order to get a multiserver system model, as well as consider confidence intervals for the key measures of interest. We leave this possibilities for future research.

Acknowledgements. Authors thank the Editor and anonymous referees for reviewing the paper and providing some helpful comments. Authors thank Dr. Rostislav Razumchik for some very useful comments. The research is supported by RF President's Grant No. MK-1641.2017.1.

References

1. Bekker, R.: Queues with state-dependent rates. Ph.D. thesis, Technische Universiteit Eindhoven, Eindhoven (2005)
2. Do, T.V., Krieger, U.R., Chakka, R.: Performance modeling of an Apache Web server with a dynamic pool of service processes. Telecommun. Syst. **39**(2), 117–129 (2008). http://link.springer.com/10.1007/s11235-008-9116-y
3. Gandhi, A., Harchol-Balter, M., Das, R., Kephart, J.O., Lefurgy, C.: Power capping via forced idleness. In: Proceedings of Workshop on Energy Efficient Design, pp. 1–6 (2009). http://repository.cmu.edu/compsci/868/
4. Gandhi, A., Harchol-Balter, M., Das, R., Lefurgy, C.: Optimal power allocation in server farms. In: Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems, SIGMETRICS 2009, pp. 157–168. ACM, New York (2009). https://doi.org/10.1145/1555349.1555368
5. Gebrehiwot, M.E., Aalto, S., Lassila, P.: Energy efficient load balancing in web server clusters. In: 2017 29th International Teletraffic Congress (ITC 29), vol. 3, pp. 13–18, September 2017

6. Gebrehiwot, M.E., Aalto, S.A., Lassila, P.: Optimal sleep-state control of energy-aware m/g/1 queues. In: Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2014, pp. 82–89. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels (2014). https://doi.org/10.4108/icst.Valuetools.2014.258149
7. Hanappe, P.: Fine-grained CPU throttling to reduce the energy footprint of volunteer computing. Technical report, Sony Computer Science Laboratory Paris (2012). http://low-energy-boinc.cslparis.fr/info/images/f/fd/Hanappe-12a.pdf
8. He, Q.M.: Fundamentals of Matrix-Analytic Methods. Springer, New York (2014)
9. Hopper, J.: Reduce linux power consumption, part 1: The cpufreq subsystem (2009). https://www.ibm.com/developerworks/library/l-cpufreq-1/
10. Kecskemeti, G., Hajji, W., Tso, F.P.: Modelling low power compute clusters for cloud simulation. In: 2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 39–45, March 2017
11. von Kistowski, J., Schreck, M., Kounev, S.: Predicting power consumption in virtualized environments. In: Fiems, D., Paolieri, M., Platis, A.N. (eds.) EPEW 2016. LNCS, vol. 9951, pp. 79–93. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46433-6_6
12. Morozov, E., Rumyantsev, A.: A state-dependent control for green computing. In: Abdelrahman, O.H., Gelenbe, E., Gorbil, G., Lent, R. (eds.) Information Sciences and Systems 2015. LNEE, vol. 363, pp. 57–67. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-22635-4_5
13. Morozov, E., Rumyantsev, A., Kalinina, K.: Inequalities for workload process in queues with NBU/NWU input. In: Gruca, A., Czachórski, T., Harezlak, K., Kozielski, S., Piotrowska, A. (eds.) ICMMI 2017. AISC, vol. 659, pp. 535–544. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-67792-7_52
14. Murthy, G.R., Rumyantsev, A.: On an exact solution of the rate matrix of quasi-birth-death process with small number of phases. In: Proceedings: 31st European Conference on Modelling and Simulation ECMS 2017, Budapest, Hungary, pp. 713–719, 23–26 May 2017. https://doi.org/10.7148/2017-0713, oCLC: 993291446
15. Neuts, M.F.: Matrix-Geometric Solutions in Stochastic Models. Johns Hopkins University Press, Baltimore (1981)
16. Pallipadi, V.: Enhanced intel speedstep technology and demand-based switching on linux (2010). https://software.intel.com/en-us/articles/enhanced-intel-speedstepr-technology-and-demand-based-switching-on-linux/
17. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2017). https://www.R-project.org/
18. Rumyantsev, A., Kalinina, K., Morozova, T.: Stochastic modelling of the supercomputer with threshold-based service rate control. In: Distributed Computer and Communication Networks: Control, Computation, Communications: Proceedings of the 20 International Scientific Conference, Moscow, pp. 286–290, 25–29 September 2017. (in Russian)

Active Queue Management Based on Congestion Policing (CP-AQM)

Michael Menth(✉) and Sebastian Veith

Chair of Communication Networks, University of Tuebingen,
Sand 13, 72076 Tuebingen, Germany
{menth,sebastian.veith}@uni-tuebingen.de

Abstract. Buffers in switches or routers are used to achieve sufficiently high utilization of transmission resources but permanently filled buffers add excessive queuing delay to communication. Active queue management (AQM) accommodates infrequent traffic bursts but avoids a standing queue. Currently, many new AQM mechanisms are discussed in the IETF. In this work, we propose a new AQM mechanism based on the idea of congestion policing. We evaluate its performance for various networking scenarios and transport protocols, and illustrate the impact of its parameters.

Keywords: Bufferbloat · Active queue management
Congestion policing

1 Introduction

Bufferbloat [14] is the phenomenon that end-to-end delay in the Internet can be very high due to large buffers in switching nodes on the paths. The delay occurs if large buffers get filled as this increases the queueing delay of packets. This observation has fueled the discussion of new active queue management (AQM) mechanisms for the Internet. Random Early Detection (RED) has been proposed as AQM for the Internet [6] already in 1998, but so far there is only little deployment which is also due to its rather difficult configuration. Therefore, the IETF has started a new working group on "Active Queue Management and Packet Scheduling" (AQM) [18] where novel AQM algorithms are investigated and standardized.

Congestion policing (CP) is the idea that traffic gets policed if a certain amount of congestion is exceeded. The idea was first introduced in [10] to improve the fairness among competing users on a remote link under congestion conditions. Packets of a user get dropped when he causes more CE-marked packets than

The authors acknowledge the funding by the Deutsche Forschungsgemeinschaft (DFG) under grant ME2727/2-1. The authors alone are responsible for the content of the paper.

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 173–187, 2018.
https://doi.org/10.1007/978-3-319-74947-1_12

his congestion allowance in form of a configured rate plus some tolerance. A quantitative evaluation was missing. In this work, we take a step back and apply this principle to the traffic entering a local queue: if more congestion occurs than a predefined amount, some packets are dropped. We propose an appropriate definition of congestion and design a congestion policer. The congestion policer is based on a token bucket and controls all traffic entering the queue. This yields the CP-AQM mechanism. We propose suitable token bucket parameters and investigate CP-AQM by means of simulation in different networking scenarios, for different transport protocols, and for different traffic types on a 10 Mb/s link. We recommend congestion parameters for which low average queue lengths and high link utilization are observed in all considered experiments.

In Sect. 2, we revisit related work and in Sect. 3 we present the design of CP-AQM. In Sect. 4 we explain our simulation setup and methodology and present comprehensive results that give insight into the performance of CP-AQM and the impact of its parameters. Section 5 summarizes the results and gives an outlook on future work.

2 Related Work

Buffers are deployed in almost all devices of the Internet as they are needed for efficient packet multiplexing. Some of them are oversized which may cause persistent excessive delay if a standing queue occurs. This phenomenon is called bufferbloat [14]. It is due to the fact that TCP cannot recognize an increasing queue unless there is packet loss or significant delay. While some authors are rather doubtful about the prevalence of bufferbloat and its impact [2,17], bufferbloat has been demonstrated in cellular networks [21]. The authors of [9] pointed out many sources contributing to Internet latency and countermeasures.

AQM mechanisms generally reduce queue lengths and fight against bufferbloat. Random Early Detection (RED) [13] was among the first AQM mechanisms for the Internet and has been proposed as a standard [6]. The survey in [1] nicely categorizes a multitude of other AQMs, many of them using the same principle as RED. The "Adaptive Virtual Queue (AVQ) is a rate-based AQM that attempts to maintain input (arrival) rate at a desired utilization" [1]. "Stabilized Virtual Buffer (SVB) uses both packet arrival rate and queue-length as part of its congestion indicator and attempts to keep the the packet arrival rate and the queue-length around their individual target values" [1]. These mechanisms are most similar to the proposed CP-AQM scheme, but differ in a significant detail: their virtual queue monitors traffic instead of congestion.

Since low delay has become more important in the recent years and since RED has not been vastly deployed, the IETF working group AQM [18] has been established. Its objective is to produce new recommendations for AQM in the Internet [4]. Three novel AQM mechanisms have been presented in that course: CoDel, PIE, and GSP. CoDel [27,28] stands for "Controlled Delay". It is mostly considered in combination with stochastic fair queuing (SFQ) to isolate flows against each other [16]. However, this combination is basically applicable

to any AQM mechanism as buffer management and scheduling are orthogonal to each other [5]. One advantage of CoDel over RED is that it copes with variable capacity links which is relevant in wireless networks. The authors of [23] provide a comparison between CoDel and RED. Interactions between CoDel and LEDBAT, a transmission protocol for background traffic, have been studied in [15]. The authors of [32] have presented a software-defined implementation in an FPGA for RED and CoDel to support 10 Gb/s. PIE [29,30] stands for "Proportional Integral controller Enhanced". PIE was designed to yield more efficient implementations than CoDel as it does not require timestamps. It has already been tested for DOCSIS systems [12,36,37]. A comparison between PIE and CoDel for DOCSIS is provided in [26]. Another comparison between PIE, CoDel, and ARED is presented in [22]. GSP is short for "Global Synchronization Protection for Packet Queues" [25]. The basic operation of GSP relies on fixed bandwidth. There is an adaptation of GSP for scenarios with higher loads and for queues with variable capacity. WQM [31] is a novel queue management system designed for IEEE 802.11n networks to fight against bufferbloat resulting from variable server rates.

CP-AQM leverages CP. CP limits the maximum congestion a user or traffic aggregate can cause by packet drops that are enforced by a policer. The idea of CP was first introduced in [10] and later in [20] and [8]. As it was originally intended for distributed systems, information about congestion caused by a flow and observed by the receiver is returned back to the sender that inserts information about this observed congestion into the IP header. This information is known as re-feedback [10] or congestion exposure (ConEx). An IETF working group [19] was established to standardize this protocol as experimental standard. ConEx information is intended to perform congestion management through CP [11]. Use cases are data centers [7] or backhaul networks for mobile access networks [24]. A modified version of ConEx-based CP has been presented in [3]. So far, there is no local application of the CP principle and there is no quantitative evaluation of CP. This work suggests such a local application and investigates its performance by means of packet-based simulation.

3 Design of CP-AQM

In this section we introduce CP-AQM. We first give an appropriate definition of congestion, then explain the design and operation of CP-AQM, and eventually derive configuration parameters.

3.1 Congestion Function

Congestion is a rather informal term denoting some form of overload on a link. However, we need a quantitative definition to measure it. Briscoe quantified it as a rate of lost and CE-marked packets on a link [8]. As packet loss occurs only under extreme load, and CE-marking of packets depends on the configuration of the marking algorithm, this definition is not appropriate for our purpose.

Instead we provide the following congestion function that depends on the current queue occupancy x in bytes:

$$c(x) = \begin{cases} 0 & x < T_c \\ 1 + \frac{x - T_c}{Q_{max} - T_c} \cdot (c_{max} - 1) & x \geq T_c. \end{cases} \quad (1)$$

Thereby Q_{max} is the capacity of the queue in bytes and T_c is the congestion threshold: queue sizes below that threshold are considered uncongested and queue sizes equal or larger are considered congested. Congestion starts with $c(T_c) = 1$, the severity of congestion linearly increases with the queue length, and reaches c_{max} for a full queue. Therefore, c_{max} should be larger than 1.

3.2 Congestion Policer

The congestion policer drops packets if the congestion on the link exceeds a configured congestion allowance or if the packet does not fit into the queue. The congestion allowance consists of token bucket parameters: rate R_{CA} (bit/s) and bucket size B_{CA} (bytes). That means, the policer has a bucket of size B_{CA} that is continuously refilled with tokens at a rate of R_{CA}. Let B be the size of a packet on the link including all overheads. If a packet arrives at the queue and the queue currently holds x bytes, then the packet contributes $B \cdot c(x)$ congestion (bytes). If the fill state of the bucket is at least $B \cdot c(x)$, the packet is accepted for sending and the fill state of the queue is reduced by $B \cdot c(x)$; otherwise, the packet is dropped by the policer without changing the bucket fill state. If the packet must be dropped because the queue is full, the fill state is not reduced.

There is an important difference between CP-AQM and conventional token bucket (or virtual queue) based policers. Conventional token bucket policers meter traffic and drop packets if the traffic stream exceeds a configured rate by some configured tolerance. CP-AQM meters congestion and drops packets if the congestion stream exceeds the configured congestion allowance. Congestion exists only if the fill state of the queue is sufficiently high, but then the congestion rate of a traffic stream may exceed its traffic rate.

3.3 Parametrization

Obvious configuration parameters of CP-AQM are the congestion threshold T_c, the maximum congestion c_{max}, and the congestion allowance parameters rate R_{CA} and bucket size B_{CA}. Let C be the link bandwidth.

If the queue is full and traffic is sent at link speed C, then a maximum congestion rate of $C \cdot c_{max}$ can be generated. Thus, for the policer to be effective, the congestion allowance rate R_{CA} must be smaller than that value. To enable the policer to avoid permanent queue occupancies of size T_c or larger, the congestion allowance rate R_{CA} should be at most C. If the bucket becomes empty, the queue is above the congestion threshold T_c. If the sender sends with at least link bandwidth C in this situation and its congestion allowance rate R_{CA} is set to a

value smaller than C, then the policer drops at least a fraction of $\frac{C-R_{CA}}{C}$ packets. To make the loss rate in that situation only dependent on the arrival rate and the fill state of the queue, but independent of R_{CA}, we choose $R_{CA} = C$.

The policer should allow the traffic to fill the entire queue, but only for short time. In particular, filling the queue with a single burst should be possible without causing the policer to drop any packets. The resulting amount of congestion can be approximated by $(Q_{max} - T_c) \cdot \frac{1+c_{max}}{2}$, which is a reasonable lower bound on the bucket size. This value is maximized for $T_c = 0$ and $c_{max} = 2$ in our experiments, which leads to $\frac{3}{2} \cdot Q_{max}$. If the physical queue has room for $Q_{max} = 45$ KB and IP packets are 1.5 KB large, a bucket size of $B_{CA} = 45 \text{ KB} \cdot \frac{1.507 \text{ KB/pkt}}{1.5 \text{ KB/pkt}} \cdot 1.5 = 67.815$ KB is needed after taking the PPP overhead in our simulation into account which is also respected for the congestion calculation. In the special case of $T_c = 0$ the bucket needs to be even one packet larger because then all packets cause congestion, even in the presence of an empty queue. We use this rule to configure B_{CA} for all experiments because additional runs showed that this bucket size is sufficient to produce high utilization and that larger bucket sizes cannot increase the utilization significantly. Thus, from a certain bucket size on, CP-AQM is rather insensitive to the congestion allowance bucket size B_{CA}. After all, the congestion threshold T_c and the maximum congestion c_{max} remain as configuration parameters for CP-AQM.

4 Queueing Behavior with CP-AQM

We investigate the impact of the configuration parameters of CP-AQM on average queue length and utilization on a 10 Mb/s link. To that end, we consider various networking scenarios, transport protocols, and traffic types. We first describe the simulation methodology and experiment setup. We illustrate the queuing behavior of both TCP New Reno and TCP Cubic connections in various networking conditions using tail-drop buffer management as baseline. Then, we show how CP-AQM performs under various conditions for non-reactive traffic generating persistent severe congestion. Eventually, we investigate the impact of CP-AQM's configuration parameters for different networking scenarios and give recommendations.

4.1 Simulation Methodology and Setup

We used the INET framework [33] of OMNeT++ [34] for simulations. As we do not trust INET's TCP implementation, we use the Network Simulation Cradle [35] based on which INET allows to integrate Linux networking stacks including TCP New Reno and Cubic.

We briefly describe the simulation setup. Users are connected to a server via a private, fast access link and a shared, slow bottleneck link. The access links have capacity $C_a = 1$ Gb/s and one-way propagation delay $D_a = 0.1$ ms while the shared bottleneck link has capacity of $C_b = 10$ Mb/s and a one-way propagation delay D_b. The access links do not cause any packet loss. The queue length on the bottleneck link is $Q_{max} = 45$ KB which corresponds to 30 maximum-size IP packets.

For non-reactive traffic with a given average rate, we consider periodic traffic and traffic with exponential inter-arrival times. TCP performance significantly depends on round-trip time (RTT) and packet loss. Especially the latter strongly depends on the number of flows on the bottleneck link. Therefore, we look at different networking scenarios with $n \in \{1, 16\}$ TCP flows and $D_b =\in \{5, 50\}$ ms. These values cause round trip times of at least 10.1 ms or 100.1 ms propagation delay plus transmission and queueing delay which may be significant.

Our simulation features a bottleneck link with $C_b = 10\,\text{Mb/s}$ over which PPP frames are transmitted. Thus, packets come with 7 bytes overhead for PPP header, 20 bytes overhead for IP header, and 8 or 20 bytes overhead for UDP or TCP header. The maximum transfer unit for an IP packet is 1500 bytes. UDP traffic is constant bit rate with 1472 bytes UDP payload per packet.

Each data point in the figure is an average gained from at least 100 simulation runs, each of them pertains to a simulation time of at least 100 s with a preceding 10 s warmup phase. Flows were randomly started within the first 5 s of the simulation. In the case of a single flow, we conducted at least 1000 simulation runs with a duration of at least 1000 s.

4.2 Tail-Drop Buffer Management

We illustrate the impact of tail-drop buffer management on the queueing behavior to provide a baseline for CP-AQM. We perform experiment series for different traffic types. Figures 1(a) and (b) show the average queue length and the utilization of the bottleneck link for different one-way delays.

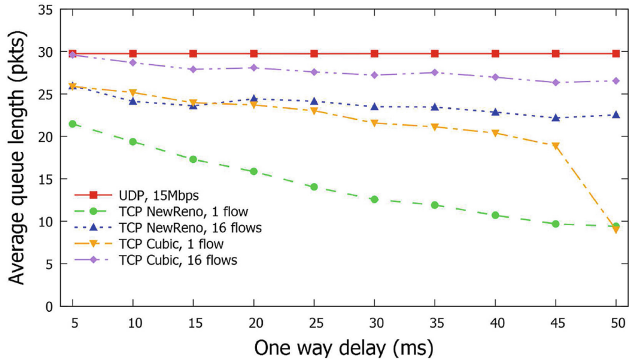
We first consider constant-bit rate UDP traffic with 15 Mb/s on PPP layer. As the offered traffic is significantly larger than the available bandwidth, the link utilization is 100% and the queue is always fully occupied regardless of the one-way delay.

A single TCP New Reno connection also achieves full utilization, but only up to a one-way delay of about $D_b = 20$ ms while for $D_b = 50$ ms the link utilization degrades to 92.7%. TCP Cubic is more aggressive and fills the pipe up to $D_b = 25$ ms and reaches a utilization of about 98% even for $D_b = 50$ ms. With increasing one-way delay, the average queue length decreases from 20 packets to 10 packets for a single TCP New Reno connection. TCP Cubic leads to larger average queue length than TCP New Reno as its congestion control algorithm increases its sending rate more quickly after packet loss.

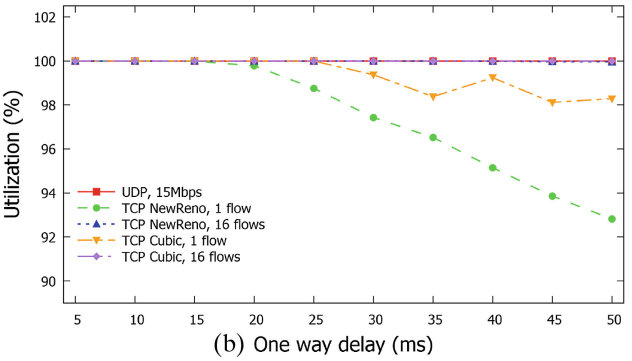
More TCP connections lead to more congestion. As a result, the link capacity can be fully used by 16 TCP flows even for a $D_b = 50$ ms, regardless of the TCP variant. The average queue length increases to values between 26 and 30 packets for TCP Cubic and 22.5 and 25.5 packets for TCP New Reno. Also packet loss becomes significant under these conditions and varies between 4% and 12% for TCP Cubic and between 2.5% and 9% for TCP New Reno (not shown in the figures).

The fact that TCP can lead to a full queue over long time, which is the case for average queue lengths above 15 packets, can be considered as bufferbloat. In particular for $D_b = 5$ ms, TCP Cubic keeps the buffer almost constantly full

which is not necessary for efficient packet multiplexing, but adds delay which is especially annoying when additional real-time traffic is also carried over the bottleneck link.



(a) Average queue length on the bottleneck link.



(b) One way delay (ms)

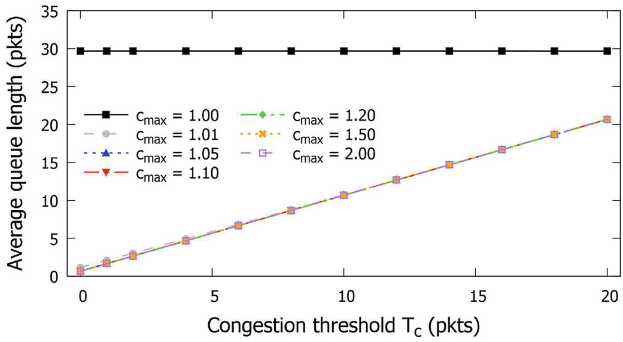
Fig. 1. Impact of networking parameters on the performance of TCP traffic with tail-drop buffer management.

4.3 Non-Responsive Traffic with CP-AQM

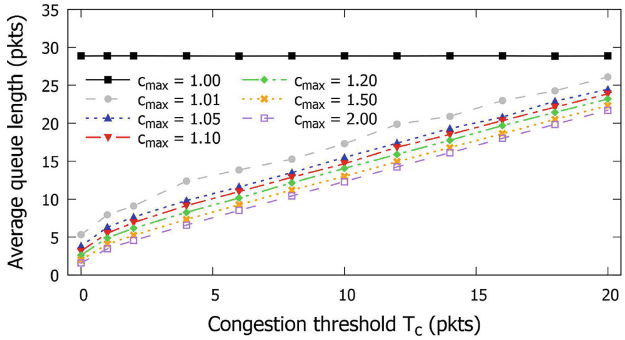
We study how configuration parameters of CP-AQM influence the queueing behavior of non-responsive traffic that causes significant overload. The link utilization was 100% for all experiment series and all investigated configuration parameters.

Figure 2(a) illustrates the average queue length for 15 Mb/s constant bit rate UDP traffic on PPP layer with constant packet inter-arrival times. For $c_{max} > 1$ the average queue length increases linearly with the congestion threshold T_c but the exact value of c_{max} has no impact. We analyze the system behavior. Since the traffic rate exceeds the link bandwidth, the queue initially increases so that the congestion function yields values larger than 1. As a result, the bucket is drained faster than it is refilled so that it eventually holds too few tokens to

accept a packet. From this point on, less traffic than link bandwidth can be accepted due to $c(x) > 1$ for $x > T_c$ so that the physical queue shrinks to T_c for which $c(T_c) = 1$ holds. At this stage, a traffic rate of exactly C_b can be accepted so that excess traffic is dropped. If the queue size falls below T_c due to a packet drop, the fill state of the bucket slightly increases, which allows a packet train of consecutive packets to be accepted so that the queue exceeds T_c by a very few packets. Thus, the queue length oscillates with a very low amplitude around T_c.



(a) Constant packet inter-arrival times.



(b) Exponential packet inter-arrival times.

Fig. 2. Impact of CP-AQM configuration parameters on the average queue length on the bottleneck link for non-responsive UDP traffic with a rate of 15 Mb/s.

Figure 2(b) shows that average queue lengths for non-responsive traffic with exponential packet inter-arrival times are larger than for periodic traffic. This is due to the fact that the bucket can refill significantly during inter-arrival times that are larger than average, which allows the queue to grow larger afterwards. We provide a simple model for this phenomenon. We assume that the queue size falls below T_c for t_{below} time which increases the bucket size by $C_b \cdot t_{below}$ tokens. If the queue size above T_c is on average Q^{avg}_{above}, the duration t_{above} of the queue size above T_c can be calculated by

$$t_{above} = \frac{C_b \cdot t_{below}}{C_b \cdot \frac{Q^{avg}_{above_c} - T_c}{Q_{max} - T_c} \cdot (c_{max} - 1)}. \quad (2)$$

Example values $t_{below} = 2.4$ ms, $c_{max} = 1.2$, $T_c = 10$ and $Q^{avg}_{above} = 15$ yield a bucket increase by 3000 bytes (2 packets) after which the queue size can stay $t_{above} = 48$ ms at $Q^{avg}_{above} = 15$ packets on average. Thus, a very short time (2.4 ms) of the queue size below T_c can allow the queue to stay for long time (48 ms) above T_c. If we assume an average queue length of 9 packets during t_{below}, this leads to an overall average queue length of 14.7 packets.

The observed deviations increase with smaller maximum congestion c_{max}. Thus, with exponential packet inter-arrival times the queue length oscillates more strongly around T_c, and c_{max} influences the size of the amplitudes. Experiments with 12 Mb/s lead to larger average queue sizes because this makes larger packet inter-arrival times more likely. Conversely, 20 Mb/s lead to smaller average queue sizes.

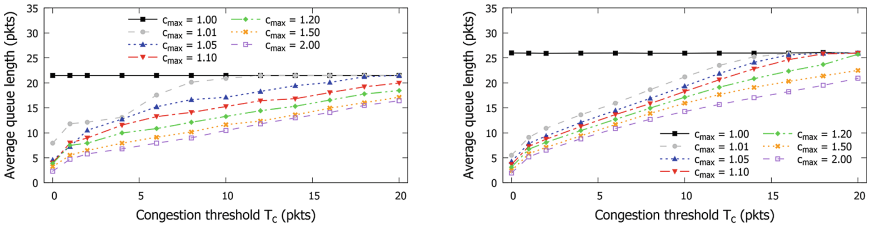
For a maximum congestion of $c_{max} = 1$ we observe in both figures a straight line on the level of the average queue length obtained for tail-drop. This parameter value does not cause the policer to drop any packets so that the queuing behavior is independent of the congestion threshold T_c and equal to the one for drop-tail. This parameter value effects that the congestion contributed by a packet is exactly its size on the channel (PPP frame size in our simulation). As the congestion allowance rate R_{CA} equals the link bandwidth C_b, the number of tokens missing in the token bucket of the policer can be at most the current queue length (minus the congestion threshold T_c, plus the PPP header overhead of the stored IP packets, to be accurate). As we have chosen the token bucket size sufficiently larger than the queue size, the token bucket cannot run empty so that the policer cannot drop packets. In the following, we use the curve for $c_{max} = 1$ as reference for tail-drop.

4.4 TCP Traffic with CP-AQM

An AQM should be configured such that it performs well for all relevant traffic patterns. As we observed in the preceding section a significant impact of the number of TCP flows, the one-way delay, and the TCP version on average queue length, we investigate the impact of CP-AQM's configuration parameters on queuing behavior for 8 combinations consisting of $n \in \{1, 16\}$ flows, $D_b \in \{5, 50\}$ ms, and TCP version $\in$ {New Reno, Cubic}.

Experiments with TCP New Reno. Figure 3(a) shows the average queue length on the bottleneck link for $n = 1$ TCP New Reno connection and $D_b = 5$ ms. It clearly increases with an increasing congestion threshold T_c and with a decreasing maximum congestion c_{max}. We briefly discuss these findings.

The fact that the average queue length increases with the congestion threshold is rather intuitive. A packet contributes to congestion only if the queue occupation is at least the congestion threshold T_c at its arrival. Only then the



(a) $n = 1$ flow and $D_b = 5$ ms.

(b) $n = 16$ flows and $D_b = 5$ ms.

(c) $n = 1$ flow and $D_b = 50$ ms.

(d) $n = 16$ flows and $D_b = 50$ ms.

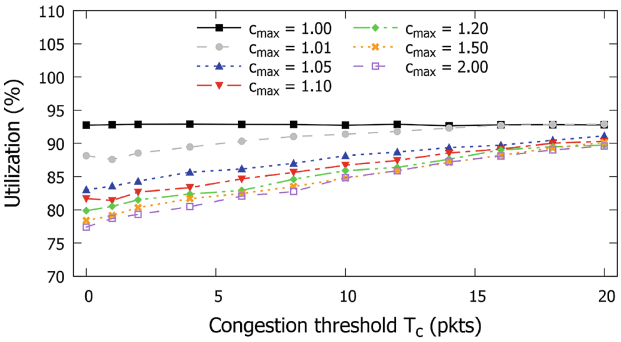
Fig. 3. Average queue lengths on the bottleneck link for TCP New Reno.

packet can be policed; otherwise it does not contribute any congestion. Thus, for larger values of T_c, the queue can grow larger without consuming tokens from the bucket, which effects policer drops only at larger queue sizes. The average queue length is shorter for larger maximum congestions c_{max} than for small ones. This is because large values of c_{max} generate a similar congestion rate already at lower queue sizes compared to small values of c_{max} so that CP starts dropping at lower queue sizes. The link utilization reaches mostly 100% except for $T_c \in \{0, 1\}$ packets and large c_{max} values where utilization is between 86% and 99% (not shown by figures). Thus, CP-AQM can limit the average queue length for $D_b = 5$ ms without sacrificing hardly any utilization.

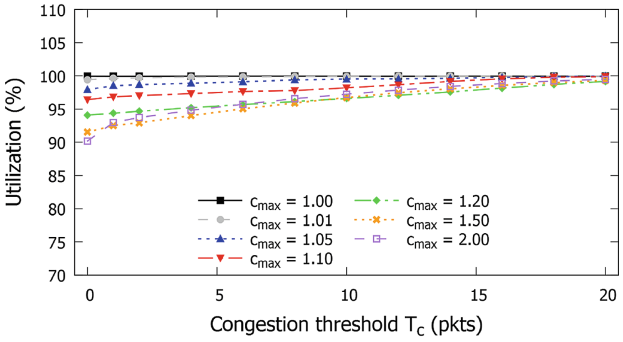
Figures 3(b)–(d) show the results for more TCP New Reno connections, or longer one-way delays, or both. They are qualitatively the same, but differ in detail. For short one-way delays $D_b = 5$ ms or for only a single $n = 1$ TCP flow, the contention for queue space is low enough so that a low congestion threshold $T_c \leq 5$ can enforce small average queue lengths. However, for $D_b = 50$ ms and $n = 16$ TCP New Reno connections, a sufficiently large maximum congestion of $c_{max} \geq 1.05$ is needed in addition to keep the average queue length low.

The link utilization is exactly or close to 100% for almost all experiments with a $D_b = 5$ ms. This is different for $D_b = 50$ ms. Figures 4(a) and (b) provide the link utilization for $D_b = 50$ ms and $n \in \{1, 16\}$ TCP Reno flows. For $n = 1$ flow, the link utilization was only 92.5% with tail-drop ($c_{max} = 1$), but the values for CP-AQM fall below that level (80%–90%), in particular for small congestion thresholds T_c and large maximum congestion c_{max}. Under these challenging conditions low average queue lengths come at the expense of reduced

link utilization. For $n = 16$ flows, significantly reduced link utilization can be avoided by choosing the maximum congestion as $c_{max} \leq 1.05$.



(a) $n = 1$ TCP New Reno flow.

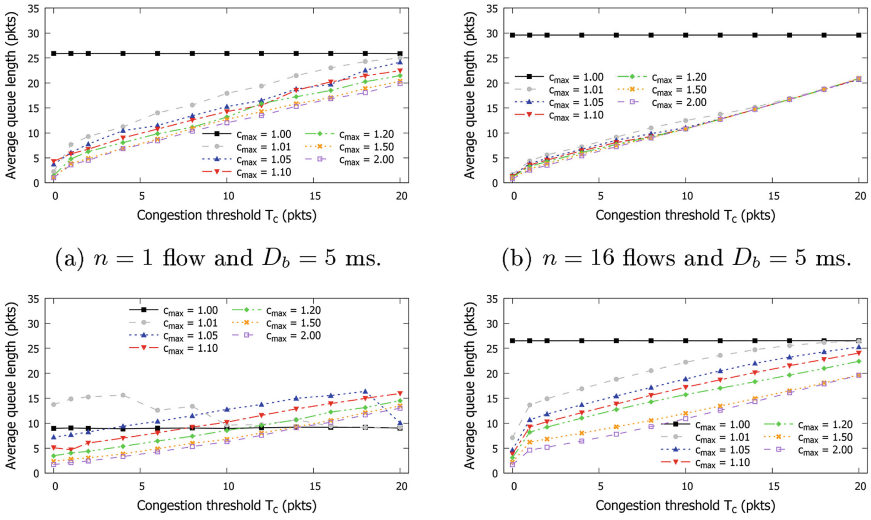


(b) $n = 16$ TCP New Reno flows.

Fig. 4. Link utilization on the bottleneck link for a one-way delay of $D_b = 50$ ms.

Experiments with TCP Cubic. Figures 5(a)–(d) show that the more aggressive TCP Cubic variant leads to very similar results regarding average queue length as TCP New Reno. For most configurations of T_c and c_{max}, the average queue length is slightly larger than for TCP New Reno. For $D_b = 50$ ms and $n = 1$ TCP Cubic flow, CP-AQM may even lead to longer average queue lengths for large T_c and small c_{max} than for tail-drop ($c_{max} = 1$). This looks counterintuitive at first sight, but may be due to implementation specifics of TCP Cubic and the fact that tail-drop regularly fills the entire queue before packet loss occurs.

We do not show figures for the link utilization with TCP Cubic and $D_b = 50$ ms, but report results. For $n = 1$ flow, the link utilization is about 98% like for tail-drop. For small congestion thresholds of $T_c \leq 5$ packets and large maximum congestions of $c_{max} \geq 1.2$, the utilization may range between 94% and 98%.



(a) $n = 1$ flow and $D_b = 5$ ms.

(b) $n = 16$ flows and $D_b = 5$ ms.

(c) $n = 1$ flow and $D_b = 50$ ms.

(d) $n = 16$ flows and $D_b = 50$ ms.

Fig. 5. Average queue lengths on the bottleneck link for TCP Cubic.

4.5 Recommendations for Configuration

We recommend to set the congestion allowance parameters R_{CA} and B_{CA} like proposed in Sect. 3.

Looking at all investigated networking scenarios, we propose to set $T_c = 5$ and $c_{max} = 1.2$ because that limits the average queue length to at most 12 packets. The price to pay is a reduced link utilization of 95% for $D_b = 50$ ms and $n = 16$ TCP New Reno connections and even lower for $n = 1$ TCP New Reno flow. When choosing a lower maximum congestion of $c_{max} = 1.05$, New Reno achieves almost full link utilization for $D_b = 50$ ms and $n = 16$ flows, but average queue lengths may reach 15 packets in that case. A larger maximum congestion of $c_{max} = 1.5$ may limit the average queue length to 9 packets with some more decrease in utilization for $D_b = 50$ ms (82% for $n = 1$ flows TCP New Reno, 94% for $n = 16$ flows TCP New Reno, 96% for $n = 1$ flow and TCP Cubic, 99% for $n = 16$ flows TCP Cubic), which is moderate for TCP Cubic. Thus, there is a tradeoff between low average queue length and high utilization where a decision needs to be taken depending on preference.

5 Conclusion

We have presented CP-AQM as a new AQM mechanism based on the idea of congestion policing (CP). Two configuration parameters define congestion based on the state of the queue and two more configuration parameters define the behavior of the congestion policer which is based on a token bucket. We derived

appropriate token bucket parameters and performed experiments to find suitable values for the remaining two parameters that define the congestion function.

To that end, we simulated average queue length and utilization in the presence of tail-drop and CP-AQM for various configurations, networking scenarios and transport protocols on a 10 Mb/s link. CP-AQM keeps the average queue length very short in case of persistent overload through non-responsive traffic. With reasonable configuration it achieves also short average queue lengths for TCP traffic (New Reno and Cubic) and 100% utilization if multiple flows are transmitted. The reduction is significant: CP-AQM ($T_c = 5$ packets) leads to an average queue length of only 7 packets for 16 TCP Cubic flows and a one-way delay of 5 ms instead of 30 packets for tail-drop. If only a single flow is transmitted, CP-AQM causes slightly decreased link utilization for large one-way delays in the range of 50 ms. The most intriguing feature of CP-AQM is that it keeps queue lengths very short for persistent non-responsive traffic while allowing mostly full utilization for TCP traffic. Moreover, our proposed configuration of CP-AQM assures that the entire buffer can be utilized by a single burst.

While this first study of CP-AQM is promising, it is unclear whether CP-AQM can be extended to cope with varying bandwidth to make it applicable also for wireless networks. CP-AQM should be compared to other AQM mechanisms such as RED [13], CoDel [28], PIE [30], and GSP [25]. A comparison is needed also for bandwidths other than 10 Mb/s, for more complex traffic patterns, and for other objectives like keeping the queue length very short even at the expense of significantly reduced utilization.

References

1. Adams, R.: Active queue management: a survey. IEEE Commun. Surv. Tutor. **15**(3), 1425–1476 (2013)
2. Allman, M.: Comments on Bufferbloat. ACM SIGCOMM Comput. Commun. Rev. **43**(1), 30–37 (2013)
3. Baillargeon, S., Johansson, I.: ConEx lite for mobile networks. In: Capacity Sharing Workshop (CSWS) (2014)
4. Baker, F., Fairhurst, G.: RFC7567: IETF Recommendations Regarding Active Queue Management, July 2015
5. Baker, F., Pan, R.: RFC7806: On Queuing, Marking, and Dropping, April 2016
6. Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., Zhang, L.: RFC2309: Recommendations on queue management and congestion avoidance in the internet, April 1998
7. Briscoe, B., Sridharan, M.: Network Performance Isolation in Data Centres using Congestion Policing, February 2014. http://tools.ietf.org/html/draft-briscoe-conex-data-centre
8. Briscoe, B.: Re-feedback: Freedom with accountability for causing congestion in a connectionless internetwork. Ph.D. thesis, Department of Computer Science, University College London (2009)

9. Briscoe, B., Brunstrom, A., Petlund, A., Hayes, D., Ros, D., Tsang, I.J., Gjessing, S., Fairhurst, G., Griwodz, C., Welzl, M.: Reducing internet latency: a survey of techniques and their merits. IEEE Commun. Surv. Tutor. **18**(3), 2149–2196 (2016)
10. Briscoe, B., Jacquet, A., di Cairano-Gilfedder, C., Salvatori, A., Soppera, A., Koyabe, M.: Policing congestion response in an internetwork using re-feedback. In: ACM SIGCOMM, Portland, OR, August 2005
11. Briscoe Ed., B., Woundy Ed., R., Cooper Ed., A.: RFC6789: Congestion Exposure (ConEx) Concepts and Use Cases, December 2012
12. Cloonan, T., Allen, J., Cotter, T., Widrevitz, B., Howe, J.: Minimizing bufferbloat and optimizing packet stream performance in DOCSIS 3.0 CMs and CMTSs. In: Cable-Tec EXPO 13, Atlanta, GA, October 2013
13. Floyd, S., Jacobson, V.: Random early detection gateways for congestion avoidance. IEEE/ACM Trans. Netw. **1**(4), 397–413 (1993)
14. Gettys, J., Nichols, K.: Bufferbloat: dark buffers in the internet. ACM Queue **9**(11) (2011)
15. Gong, Y., Rossi, D., Testa, C., Valenti, S., Taht, M.D.: Fighting the bufferbloat: on the coexistence of AQM and low priority congestion control. In: IEEE INFOCOM Workshop on Traffic Measurement and Analysis (2013)
16. Hoeiland-Joergensen, T., McKenney, P., Taht, D., Gettys, J., Dumazet, E.: RFC8290: The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm, January 2018. https://tools.ietf.org/html/rfc8290
17. Hohlfeld, O., Pujol, E., Ciucu, F., Feldmann, A., Barford, P.: BufferBloat: how relevant? A QoE perspective on buffer sizing. Technical report 2012-11, TU Berlin, Faculty of Electrical Engineering and Computer Science, November 2012
18. IETF Working Group on Active Queue Management and Packet Scheduling (AQM): Description of the Working Group (2013). http://tools.ietf.org/wg/aqm/charters
19. IETF Working Group on Congestion Exposure (CONEX): Description of the Working Group (2010). http://tools.ietf.org/wg/conex/charters
20. Jacquet, A., Briscoe, B., Moncaster, T.: Policing freedom to use the internet resource pool. In: Re-Architecting the Internet (ReArch), Madrid, Spain, December 2008
21. Jiang, H., Liu, Z., Wang, Y., Lee, K., Rhee, I.: Understanding bufferbloat in cellular networks. In: Workshop on Cellular Networks: Operations, Challenges, and Future Design (CellNet), August 2012
22. Khademi, N., Ros, D., Welzl, M.: The new AQM kids on the block: an experimental evaluation of CoDel and PI. In: IEEE Global Internet Symposium (2014)
23. Kuhn, N., Lochin, E., Mehani, O.: Revisiting old friends: is CoDel really achieving what RED cannot? In: Capacity Sharing Workshop (CSWS) (2014)
24. Kutscher, D., Mir, F., Winter, R., Krishnan, S., Zhang, Y., Bernados, C.J.: Mobile Communication Congestion Exposure Scenario, October 2015. http://tools.ietf.org/html/draft-ietf-conex-mobile
25. Lautenschlaeger, W.: Global Synchronization Protection for Packet Queues, May 2016. http://tools.ietf.org/html/draft-lauten-aqm-gsp
26. Martin, J., Hong, G., Westall, J.: Managing fairness and application performance with active queue management in DOCSIS-based cable. In: Capacity Sharing Workshop (CSWS) (2014)
27. Nichols, K., Jacobson, V., McGregor, Ed., A., Iyengar, Ed., J.: RFC8289: Controlled Delay Active Queue Management, January 2018. https://tools.ietf.org/html/rfc8289

28. Nichols, K., Jacobson, V.: Controlling queue delay. ACM Queue **10**(5) (2012)
29. Pan, R., Natarajan, P., Baker, F., White, G.: RFC8033: Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem, February 2017. https://tools.ietf.org/html/rfc8033
30. Pan, R., Natarajan, P., Piglione, C., Prabhu, M.S., Subramanian, V., Baker, F., VerSteeg, B.: PIE: a lightweight control scheme to address the bufferbloat problem. In: IEEE Workshop on High Performance Switching and Routing (HPSR) (2013)
31. Showail, A., Jamshaid, K., Shihada, B.: WQM: an aggregation-aware queue management scheme for IEEE 802.11n based networks. In: Capacity Sharing Workshop (CSWS) (2014)
32. Sivaraman, A., Winstein, K., Subramanian, S., Balakrishnan, H.: No silver bullet: extending SDN to the data plane. In: ACM Workshop on Hot Topics in Networks (HotNets), November 2013
33. Varga, A.: INET-2.2 released, August 2013. http://inet.omnetpp.org/
34. Varga, A.: OMNeT++ 4.3.1 released, September 2013. http://www.omnetpp.org/
35. Wand, S.: Network Simulation Cradle (2012). http://research.wand.net.nz/software/nsc.php
36. White, G., Pan, R.: RFC8034: Active Queue Management (AQM) Based on Proportional Integral Controller Enhanced (PIE) for Data-Over-Cable Service Interface Specifications (DOCSIS) Cable Modems, February 2017. https://tools.ietf.org/html/rfc8034
37. White, G.: Active Queue Management in Docsis 3.X Cable Modems. Technical report, Cable Television Laboratories, Inc., May 2014

Deficit Round Robin with Limited Deficit Savings (DRR-LDS) for Fairness Among TCP Users

Michael Menth(✉), Marcel Mehl, and Sebastian Veith

Chair of Communication Networks, University of Tuebingen,
Sand 13, 72076 Tuebingen, Germany
{menth,sebastian.veith}@uni-tuebingen.de,
marcel.mehl@student.uni-tuebingen.de

Abstract. Deficit Round Robin (DRR) is a simple and computationally efficient approximation of the Weighted Fair Queueing (WFQ) scheduling discipline. Its intention is to share resources among several queues, e.g., flows or users, according to given weights. However, when users hold different numbers of TCP connections with saturated sources, the throughput among these users may differ significantly.

In this work, we quantify the difference in throughput for heavy and light users with saturated TCP flows for equal weights and for two different buffer management strategies. The difference is large if low queueing delay for packets is enforced through shallow buffers on the bottleneck link. To address this shortcoming, we propose limited deficit savings (LDS), a modification of the DRR scheduler, which can be combined with different buffer management schemes. We show that LDS reduces unequal throughput for heavy and light users with saturated TCP flows. Moreover, we illustrate that LDS clearly decreases download times for data chunks of moderate size in the presence of high background load.

Keywords: Congestion management · Scheduling · Fairness
Buffer management

1 Introduction

Deficit Round Robin (DRR) [28] is a computationally efficient approximation of the Weighted Fair Queuing (WFQ) scheduler. It serves several packet queues and allocates to them the capacity of a single server, e.g., the bandwidth of a communication link, according to configurable weights. In particular, DRR respects packet sizes so that queues cannot obtain larger capacity shares by sending larger packets.

The authors acknowledge the funding by the Deutsche Forschungsgemeinschaft (DFG) under grant ME2727/2-1. The authors alone are responsible for the content of the paper.

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 188–201, 2018.
https://doi.org/10.1007/978-3-319-74947-1_13

While fair allocation of transmission bandwidth has been the focus for many years, low delay has recently become more important. Drivers for low-delay transmission in the Internet and communication networks in general are real-time applications like voice over IP, video conferencing, financial applications, and gaming. Several specialized working groups in the Internet Engineering Task Force (IETF) work on mechanisms to reduce delay and congestion in the Internet: ConEx [16], RMCAT [17], and AQM [15]. The FP7 project RITE pursues that objective [1] and the Internet Society organized two workshops in this area [18,19]. An overview to reduce Internet latency is given in [5].

If low packet delay is desired with WFQ or DRR, some packets are dropped if the buffer holds too many of them. Such a drop decision is part of the buffer management schemes. We basically consider two strategies: drop-on-enqueue and drop-on-dequeue. As TCP is still the predominant transport protocol in the Internet, we investigate the transmission of saturated TCP sources. Heavy users may have more TCP connections than light users. We show that both considered buffer management methods do not lead to equal bandwidth allocation among heavy and light users if low delay is enforced.

A major reason for that phenomenon is the fact that in the DRR algorithm a queue does not benefit in the resource allocation process while it is empty. This problem increases with tighter delay requirements. To mitigate that problem, we propose limited deficit savings (LDS) as an extension to DRR so that queues can collect credits during short periods of inactivity between the last packet sent and the next packet arrived. We evaluated this mechanism under various conditions. To that end, we implemented variants of DRR in the INET simulation framework of OMNeT++ [30] and performed multiple experiments.

The paper is structured as follows. Section 2 explains the DRR algorithm, reviews existing work about DRR and WFQ, active queue management (AQM), and distinguishes our approach from other activities. Section 3 introduces the LDS extensions for DRR. Section 4 explains the experimental setup and discusses simulation results. Section 5 summarizes this work and draws conclusions.

2 Related Work

In this section, we briefly introduce WFQ and some of its variants and explain the DRR algorithm. We introduce the notion of bufferbloat, point out several AQM methods, and distinguish these efforts from our approach.

2.1 Weighted Fair Queueing and Variants

In 1985, John Nagle discussed the benefits of a round robin scheduling system [23] which was later called fair queueing (FQ). The approach was enhanced by Demers et al. [6] and by Zhang [36] towards a logical bit-by-bit fair scheduler which became known as Weighted Fair Queueing (WFQ). The introduction of weights allows for unequal resource allocation of capacity to different queues. In 1995, Shreedhar and Varghese proposed the Deficit Round Robin (DRR)

scheduler [28], which approximates WFQ but is computationally less demanding. Other improved approximations followed like the Worst-case Fair Weighted Fair Queueing (WF2Q) which utilizes the start time of packets additionally to the finish time to enhance accuracy, or WF2Q+ which improved accuracy and reduced complexity [4].

2.2 DRR Algorithm

We now describe DRR in more detail as it is the base for our study. With DRR, queues are associated with weights and the objective of the DRR is to assign the capacity of a server or bandwidth of a link to the active queues in the system according to these weights. A queue is active if it stores a packet, otherwise it is inactive. The queues are administered in an active list and a set of inactive queues. If a packet arrives for an inactive queue, the queue is removed from the set of inactive queues and appended to the end of the active list. The active queues are served in the following manner. Every queue is associated with a deficit counter. The deficit counter of the first queue from the active list is incremented by a quantum, which is an amount of bytes, multiplied by the weight associated with that queue. If the deficit counter is at least as large as the size of the first packet in the queue, the deficit counter is decreased by the size of that packet, and that packet will be sent next. This process continues until the deficit counter is not large enough to send the next packet or until the queue is empty. The queue is then removed from the active list. If it still holds packets, it is appended again to the end of the active list, otherwise it is added to the inactive set and the deficit counter is reset to zero. Then the process of assigning a quantum to the first queue in the active list and sending packets continues. DRR uses a shared buffer for all queues. If there is no space left in the buffer upon arrival of a new packet, a packet of the longest queue is dropped. This buffer management is called McKenney's buffer stealing algorithm [22]. In the following we refer to it as drop-on-enqueue.

2.3 Bufferbloat

Sufficiently large buffers are needed under certain conditions to achieve good bandwidth utilization on networking hardware [7,8], but if they are filled for relatively long time, packets experience excessive and unnecessary delay. This phenomenon is called bufferbloat [10], i.e., excessive packet delay due to large and unmanaged buffers. The general problem behind bufferbloat was already described in 1985 by John Nagle [23]. Quantitative evaluations showed that bufferbloat is not ubiquitous and its impact may be limited [3,14]. Bufferbloat in cellular networks has been studied in [20].

2.4 Active Queue Management

Active queue management (AQM) is a class of buffer management schemes that counteract incipient congestion by dropping or marking packets before the queue

is fully occupied. A classic scheme is Random Early Detection (RED) [9] that drops packets with a probability rising with the recent average queue occupation. With explicit congestion notification (ECN), packets are rather marked instead of dropped, but ECN is applicable only if TCP sources indicate to reduce their traffic rate in response to appropriate congestion signals from TCP receivers [27]. The PIE controller is an enhanced AQM whose packet drop probability depends on growing and shrinking average queue sizes [26]. It is already applied for DOCSIS systems [33,35], [34, Annex M]. The Controlled Delay (CoDel) AQM [24,25] is currently proposed as a countermeasure against bufferbloat and already integrated in many stacks. It is extended towards FQ-CoDel [13] by combining it with DRR and Stochastic Fair Queueing (SFQ) [22] so that low-rate flows within a traffic aggregate do not suffer excessive delay due to competing high-rate flows. Various new AQMs have been compared in [21], and [2] reviews a multitude of AQMs that have been discussed in the past. Interactions between AQMs and low-priority congestion control have been investigated in [11,12]. The authors of [29] have considered various buffer management strategies together with per-flow queueing strategies in Gigabit routers.

2.5 Our Approach

In our work we consider a scheduler with multiple queues, one for a specific aggregate that we call a user in the following. Each user may have multiple flows and the objective of the scheduler is to enforce a resource allocation to queues according to configured weights. This is typically achieved by WFQ. However, another goal is to keep packet delay low which is the objective of AQM or buffer management mechanisms. We try to achieve low delay with DRR with simple buffer management strategies and to understand their impact. FQ-CoDel is close to our approach in the sense that it keeps delays low and uses DRR, but it uses SFQ to isolate flows of a single user in different queues. Of course, CoDel could be easily modified for per-user queueing. The objective of our work is the enhancement of DRR by LDS from which other mechanisms like FQ-CoDel could also profit.

3 Limited Deficit Savings for DRR

With DRR, the deficit counters of queues are increased only if they hold at least a single packet; otherwise they are not respected for the resource allocation process. However, when low latency is enforced in the presence of multiple queues, queues are likely to hold no packet most of the time even though the system is fully utilized. As a consequence, queues with less frequent packet arrivals are disadvantaged in competing for the capacity share given by the weights. This problem arises for TCP traffic.

Our idea is to allow an empty queue to increase its deficit counter up to a deficit saving limit D_{max}. At the next packet arrival, the queue does not need to wait for a deficit increase but can be served preferentially within a set of queues that have enough deficit to send packets. We describe this DDR-LDS algorithm in the following.

3.1 State Classification of Queues

A queue is fresh if it is empty and if its deficit counter D equals D_{max}. All fresh queues are kept in the "fresh list" $\mathcal{F}$. A queue is eligible if it holds at least one packet and if its deficit counter D is at least as large as the size B of its first packet so that this packet could be sent. All eligible queues are kept in the "eligible list" $\mathcal{E}$. An empty queue is collecting if its deficit counter D is smaller than D_{max}. A non-empty queue is collecting if its deficit counter D is smaller than the size of its first packet B so that this packet cannot be sent. All collecting queues are kept in the "collecting list" $\mathcal{C}$. Figure 1 depicts how queues move from one list to another if their state changes. The states of the queues in the respective lists are indicated in the circles. The algorithms in the following describe how these state changes are triggered and when the queues are moved.

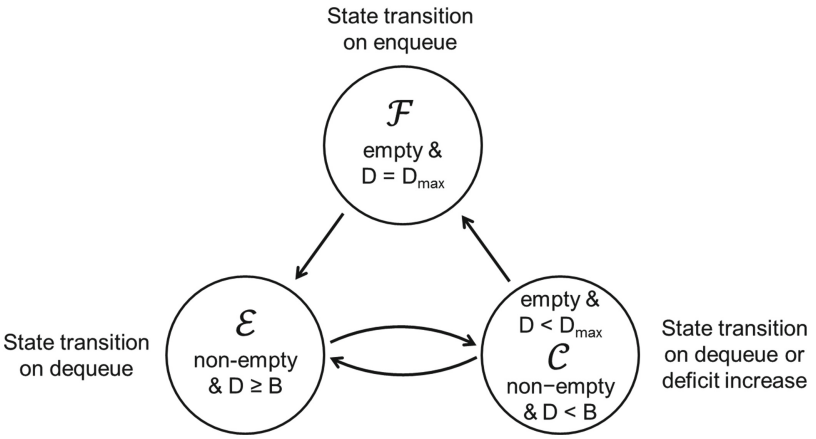


Fig. 1. A queue belongs to the fresh, eligible, or collecting list depending on its state. Queues are moved from one list to another during the execution of DRR-LDS.

3.2 Packet Enqueue

If a packet arrives, it is appended to its queue. If the queue was fresh before packet arrival, it is removed from $\mathcal{F}$ and appended to $\mathcal{E}$. If the queue was empty and in $\mathcal{C}$ before, it may need to be removed from $\mathcal{C}$ and appended to $\mathcal{E}$, depending on its new state. If the link is idle after a packet arrival, a packet may be immediately dequeued for transmission as described in the next paragraph.

3.3 Packet Dequeue

If a packet is needed for transmission and $\mathcal{E}$ is not empty, the algorithm removes the first packet of the first queue in $\mathcal{E}$. If that packet will be dropped for some reason and if the queue is empty afterwards, the queue is removed from $\mathcal{E}$ and

appended to $\mathcal{C}$ or $\mathcal{F}$ depending on the queue classification criteria. If the packet will be transmitted, the queue is removed from $\mathcal{E}$, its deficit counter is decreased by the packet size, and the queue is appended to $\mathcal{E}$ or $\mathcal{C}$, depending on its new state. This procedure repeats until a packet for transmission is found or until $\mathcal{E}$ is empty. In the latter case the deficit will be increased as described in the next paragraph.

3.4 Deficit Increase

If $\mathcal{E}$ is empty, the deficit counters of queues in $\mathcal{C}$ are increased until $\mathcal{E}$ is no longer empty or until $\mathcal{C}$ itself is empty. To that end, the deficit counter of the first queue in $\mathcal{C}$ is increased by the quantum multiplied by the queue's weight, but the deficit counter cannot exceed D_{max} if the queue is empty. Then, the queue is removed from $\mathcal{C}$ and then appended to $\mathcal{F}$, $\mathcal{E}$, or $\mathcal{C}$. This procedure repeats until $\mathcal{C}$ is empty or until $\mathcal{E}$ becomes non-empty. In the latter case, packet dequeue is resumed.

3.5 Some Observations

Queues can collect deficit only if no queue is eligible. If the buffer runs empty, all queues become fresh again. FQ-CoDel implements a similar mechanism that is described in [13]. However, with FQ-CoDel the time empty queues take to become fresh again is independent of D_{max}. With DRR-LDS the time an empty queue needs to become fresh again does depend on D_{max}.

4 Performance Evaluation

We first describe the simulation setup. Then we show that DRR cannot enforce equal bandwidth allocation in the presence of heavy and light users with TCP traffic and low latency requirements. We demonstrate that DRR-LDS reduces this unequal bandwidth allocation and leads to fast downloads of short transactions for certain buffer management schemes.

4.1 Experiment Setup

We implemented DRR and DRR-LDS in combination with two buffer management strategies based on the INET framework [30] for the discrete-event simulator OMNeT++ [31]. As we do not trust INET's TCP implementation, we use the Network Simulation Cradle [32] based on which INET allows to integrate Linux networking stacks for TCP New Reno and Cubic.

Figure 2 illustrates our simulation setup. A set of "users" is connected to one router over a fast link with a bandwidth of $C_a = 1$ Gb/s and a one-way propagation delay of $D_a = 1$ µs. This router connects to a server over a slow bottleneck link with a bandwidth of $C_b = 10$ Mb/s and a one-way propagation delay of $D_b = 5$ ms or $D_b = 50$ ms, respectively. Thus, the transmission time of a packet with an average size of 1500 bytes takes 1.2 ms. We apply DRR or DRR-LCS

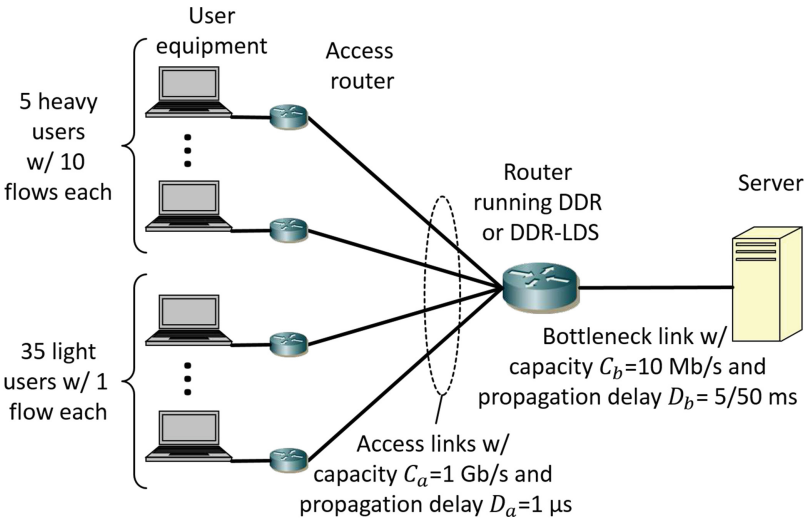


Fig. 2. Simulation setup.

for this bottleneck link and assign equal weights to all users. We configured the DRR with a quantum of 1500 bytes. The experiments use saturated TCP connections, i.e., there is always data to send. The TCP connections start randomly within the first second of a simulation run and statistic collection starts only after 5 s to avoid transient effects. If not mentioned differently, each simulation run is repeated 100 times.

4.2 Buffer Management Schemes

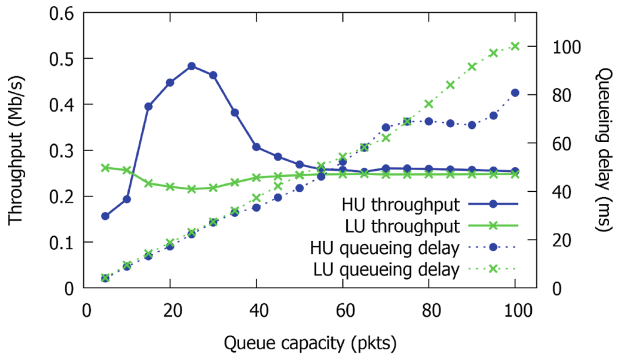
We consider two buffer management schemes for a shared buffer: *drop-on-enqueue* and *drop-on-dequeue*.

Drop-on-enqueue is the strategy originally proposed with DRR: if a packet arrives and the buffer is fully occupied, the oldest packet of the longest queue is dropped. Thereby, the packet delay can be controlled by the buffer size S_B.

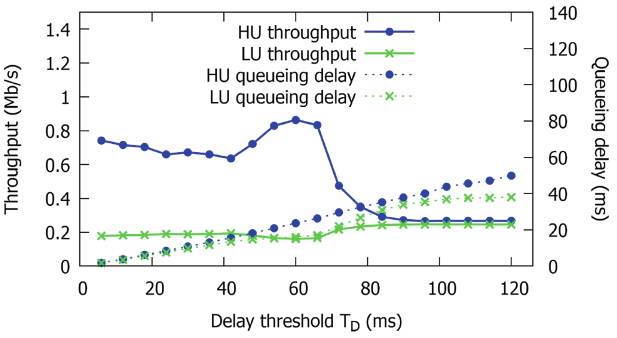
Drop-on-dequeue is inspired by new AQMs like CoDel, but we pursue a very simple approach. A packet is dropped if it is older than a configurable delay threshold T_D. This method limits the packet delay to T_D and removes packets from the queue in case of congestion. Nevertheless, the buffer can overflow under certain conditions. To avoid that, we postulated a sufficiently large buffer and assumed infinite for the sake of simplicity.

4.3 Resource Allocation with DRR

The objective of this experiment is to test whether DRR can enforce an intended resource allocation in the presence of many users so that DRR queues run empty. We consider $n = 40$ active users, 5 heavy users holding 10 TCP connections



(a) Drop-on-enqueue.



(b) Drop-on-dequeue.

Fig. 3. Throughput and packet queueing delay for heavy users (HU) and light users (LU). Results are shown for TCP New Reno and $D_b = 5\,\text{ms}$.

with the server and 35 light users holding only a single TCP connection with the server. With FIFO scheduling, we expect ratios of 10:1 for achieved transmission rates of heavy and light users assuming TCP's per-flow fairness is perfect. With DRR scheduling we expect equal transmission rates of 0.25 Mb/s for heavy and light users.

Figure 3(a) shows the throughput and packet queueing delay of heavy and light users with TCP New Reno connections for drop-on-enqueue buffer management and with $D_b = 5\,\text{ms}$ on the bottleneck link. The x-axis shows the buffer size and the y-axes the throughput and packet queueing delay of each user type. For buffer sizes between 15 and 40 packets, heavy users achieve significantly larger transmission rates than light users. This is undesired insofar as both heavy and light users have saturated sources an should share the link equally. The queueing delay for packets rises about linearly with the buffer size and its mean reveals that about $\frac{5}{6}$ of the queue is occupied on average. Thus, the queue is often filled which is due to the relatively high load. For very large queue capacity, we observe

shorter delays for heavy users than for light users. As DRR with drop-on-enqueue requires 50 packets to achieve equal bandwidth allocation for competing TCP users in our setting, it cannot enforce both fairness and low or moderate delay under heavy load.

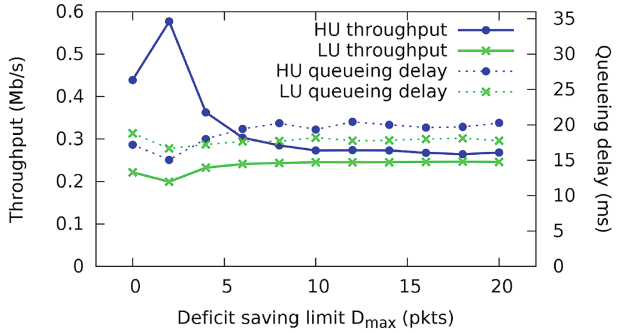
Figure 3(b) presents similar results for drop-on-dequeue. The x-axis shows the delay threshold T_D instead of the buffer size S_B; the range [6;120] ms is chosen because that corresponds to the transmission times of a packet range [5;100] which was used for drop-on-enqueue in Fig. 3(a). For drop-on-dequeue we observe even larger differences in throughput for heavy and light users, in particular for small delay thresholds. A delay threshold of $T_D = 90$ ms is needed to achieve equal bandwidth allocation for heavy and light users which results in almost 40 ms average packet delay for both user types. With drop-on-dequeue, packet queueing delay rises about linearly with the delay threshold T_D and is similar for heavy and for light users due to the absence of buffer stealing. It is about half the delay threshold T_D in our experiments. With drop-on-enqueue, the queue length cannot exceed S_B but the buffer was mostly fully occupied, which is not shown in the figures. For drop-on-dequeue the buffer size was not limited. Nevertheless, the measured average queueing delay linearly increased from zero (for $T_D = 0$ ms) to 50 ms (for $T_D = 120$ ms). Thus, the average queueing delay for drop-on-dequeue is significantly lower than the corresponding average queueing delay for drop-on-enqueue.

We performed the same experiments with $D_b = 50$ ms and obtained almost the same results. For TCP Cubic we received different results but the same conclusion: large queue capacity is needed for fair bandwidth sharing.

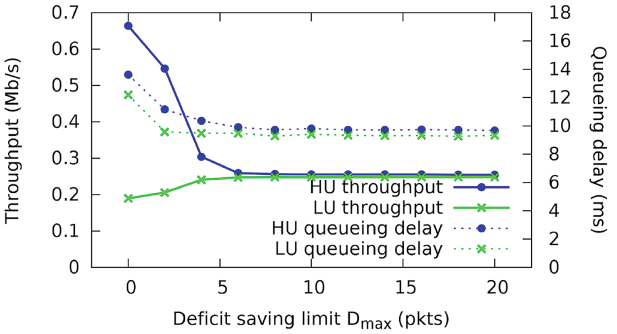
4.4 Resource Allocation with DRR-LDS

We evaluate the effect of limited deficit savings (LDS) on the throughput of heavy and light users. We first consider drop-on-enqueue for a buffer size of $S_B = 20$ packets to keep queueing delay short. The x-axis in Fig. 4(a) shows the deficit saving limit D_{max} and the y-axes show again the throughput and packet queueing delay of heavy and light users with TCP New Reno connections on a bottleneck link with a $D_b = 5$ ms. We observe that the throughput for heavy and light users significantly deviates for very small values of D_{max} but the difference vanishes for values $D_{max} \geq 10$. We explain this phenomenon by the fact that at the beginning of a congestion phase when the buffer is filled, the heavy users quickly consume their deficit. This gives priority to packets of light users when they arrive. Also during congestion periods, light users can save deficit whenever heavy users receive deficit to send further packets although they do not have packets to send. When traffic is shared about equally, the queueing delay is about 20 ms and for heavy users slightly larger than for light users.

Figure 4(b) shows similar data for drop-on-dequeue for which a delay threshold of $T_D = 24$ ms is chosen that corresponds to the transmission time of 20 packets. The difference in throughput between heavy and light users is even larger than for drop-on-enqueue. It is again decreased by an increasing value for the deficit saving limit D_{max}. A deficit saving limit of $D_{max} = 6$ is already



(a) Drop-on-enqueue for $S_B = 20$ packets.



(b) Drop-on-dequeue for $T_D = 24$ ms.

Fig. 4. Throughput and packet queueing delay for heavy users (HU) and light users (LU) *with LDS*. Results are shown for TCP New Reno and $D_b = 5$ ms.

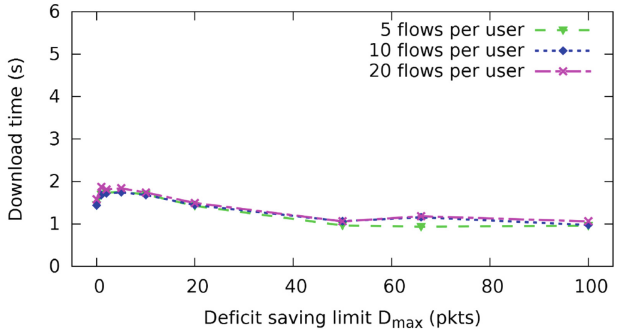
enough to achieve perfect fairness. Packet queueing delay is about 10 ms and for heavy users slightly larger than for light users.

We also performed these experiments with $D_b = 50$ ms and obtained almost the same results. For TCP Cubic we received different results. In particular, with drop-on-enqueue, light users achieved significantly larger throughput than heavy users, but light users experienced also clearly more packet delay. In contrast, drop-on-dequeue leads to a very similar queueing behavior as in Fig. 4(b) for TCP New Reno.

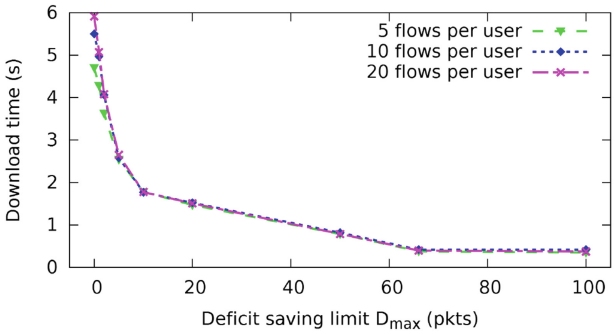
Thus, drop-on-dequeue with DRR-LDS seems an interesting approach to maximize fairness between heavy and light users with TCP flows when low latency is required.

4.5 Download Times with DDR-LDS

We consider the download time of an infrequent user sending small data bursts of 100 kB in the presence of a heavy load situation. The infrequent user holds a



(a) Drop-on-enqueue for $S_B = 20$ packets.



(b) Drop-on-dequeue for $T_D = 24$ ms.

Fig. 5. Download time of a data chunk of 100 kB for an infrequent user in the presence of heavy background traffic. The 20 other users have 5, 10, or 20 saturated TCP connections. Results are shown for TCP New Reno and $D_b = 5$ ms.

single TCP New Reno connection to download a data chunk of 100 kB for which the duration is measured. The 20 other users hold 5, 10, and 20 connections in different experiments. The TCP connections of the 20 other users are saturated, constitute the background load, and start within the first second of the experiment. The infrequent user starts its download after 5 s. Each experiment is repeated 100 times. With perfect fairness the download time for 100 kB is 1.68 s under the assumption that TCP can initially send already sufficiently fast.

We first consider the download time for drop-on-enqueue configured with a buffer size of $S_B = 20$ packets. Figure 5(a) shows the download time for the infrequent user depending on the deficit saving limit D_{max}. The download time is almost independent of the background traffic. Small deficit saving limits D_{max} lead to a slight increase in download time, but larger D_{max} clearly decreases the download time below 1.68 s. Thus, a reduction of download time through LDS is visible but rather modest. The same results are obtained for $D_b = 50$ ms and for TCP Cubic.

To study drop-on-dequeue, we use a delay threshold of $T_D = 24\,\text{ms}$. Figure 5(b) shows that without LDS ($D_{max} = 0$) the infrequent user's download time is significantly larger than for drop-on-enqueue. Larger background loads lead to longer download times without LDS, i.e., 4.8 s when the other users hold 5 connections each, 5.5 s when they hold 10 connections each, and 5.9 s when they hold 20 connections each. With very little deficit saving limit D_{max}, the difference in download time already vanishes. With even larger deficit saving limit D_{max} the download time decreases to less than 0.4 s. This is a significant speedup even compared to the fair download time of 1.68 s. Very similar results are obtained for $D_b = 50\,\text{ms}$ and for TCP Cubic.

Thus, LDS in combination with DRR can expedite transactional traffic in the presence of heavy background load, with both the drop-on-enqueue or the drop-on-dequeue buffer management strategy. For drop-on-dequeue the reduction of download time is very high.

5 Conclusion

In this work we have shown that DRR cannot achieve equal bandwidth allocation to heavy and light users that have different numbers of saturated TCP flows, in particular if low delay is enforced through shallow buffers. A reason for this phenomenon is that empty queues do not profit in DRR's bandwidth allocation process. If low delay is enforced, even queues with active TCP flows are empty for quite some time so that it is hard for them to get their full bandwidth share. Therefore, we modified the DRR algorithm such that empty queues are respected for the bandwidth allocation process in DDR and can save deficit to a limited extent. We called this mechanism limited deficit savings (LDS) and refer to the modified DDR algorithm as DDR-LDS. Our simulation results demonstrated the throughput differences between heavy and light users and revealed that they are larger for drop-on-dequeue than for drop-on-enqueue and for TCP New Reno than for TCP Cubic. We showed that LDS significantly reduces these differences. Moreover, LDS clearly decreases download times of light users for both buffer management strategies whereby the effect for drop-on-dequeue is larger than for drop-on-enqueue.

The study points at potential unfairness with DRR under low-delay constraints for TCP users. More research is needed to understand how DRR-LDS affects other traffic types, traffic mixes, and how the performance depends on networking parameters that typically influence TCP throughput. Moreover, measurements are needed to evaluate whether the investigated scenario is rather a corner case or whether the DRR-LDS scheduler is able to solve practical problems. Nevertheless, a variant of LDS can be identified in FQ-CoDel so that the presented mechanism has practical relevance and should be understood.

References

1. FP7 Project (STREP): Reducing Internet Transport Latency (RITE), November 2012–2015
2. Adams, R.: Active queue management: a survey. IEEE Commun. Surv. Tutor. **15**(3), 1425–1476 (2013)
3. Allman, M.: Comments on bufferbloat. ACM SIGCOMM Comput. Commun. Rev. **43**(1), 30–37 (2013)
4. Bennett, J.C., Zhang, H.: WF^2Q : worst-case fair weighted fair queueing. In: IEEE Infocom (1996)
5. Briscoe, B., Brunstrom, A., Petlund, A., Hayes, D., Ros, D., Tsang, I.J., Gjessing, S., Fairhurst, G., Griwodz, C., Welzl, M.: Reducing internet latency: a survey of techniques and their merits. IEEE Commun. Surv. Tutor. **18**(3), 2149–2196 (2016)
6. Demers, A., Keshav, S., Shenker, S.: Analysis and simulation of a fair queuing algorithm. In: ACM SIGCOMM (1989)
7. Dhamdhere, A., Dovrolis, C.: Open issues in router buffer sizing. ACM SIGCOMM Comput. Commun. Rev. **36**(1), 87–92 (2006)
8. Dhamdhere, A., Jiang, H., Dovrolis, C.: Buffer sizing for congested internet links. In: IEEE Infocom. Miami, FL, March 2005
9. Floyd, S., Jacobson, V.: Random early detection gateways for congestion avoidance. IEEE/ACM Trans. Netw. **1**(4), 397–413 (1993)
10. Gettys, J., Nichols, K.: Bufferbloat: dark buffers in the internet. ACM Queue **9**(11), 40 (2011)
11. Gong, Y., Rossi, D., Leonardi, E.: Modeling the interdependency of low-priority congestion control and active queue management. In: International Teletraffic Congress (ITC) (2013)
12. Gong, Y., Rossi, D., Testa, C., Valenti, S., Taht, M.D.: Fighting the bufferbloat: on the coexistence of AQM and low priority congestion control. In: IEEE INFOCOM Workshop on Traffic Measurement and Analysis (2013)
13. Hoeiland-Joergensen, T., McKenney, P., Taht, D., Gettys, J., Dumazet, E.: RFC8290: The Flow Queue CoDel Packet Scheduler andActive Queue Management Algorithm, January 2018. https://tools.ietf.org/html/rfc8290
14. Hohlfeld, O., Pujol, E., Ciucu, F., Feldmann, A., Barford, P.: BufferBloat: How Relevant? A QoE Perspective on Buffer Sizing. Technical report 2012–11, TU Berlin, Faculty of Electrical Engineering and Computer Science, November 2012
15. IETF Working Group on Active Queue Management and Packet Scheduling (AQM): Description of the Working Group (2013). http://tools.ietf.org/wg/aqm/charters
16. IETF Working Group on Congestion Exposure (CONEX): Description of the Working Group (2010). http://tools.ietf.org/wg/conex/charters
17. IETF Working Group on RTP Media Congestion Avoidance Techniques (RMCAT): Description of the Working Group (2012). http://tools.ietf.org/wg/rmcat/charters
18. Internet Society: Bandwidth Management - Internet Society Technology Roundtable Series, November 2012. http://www.internetsociety.org/sites/default/files/BWroundtable_report-1.0.pdf
19. Internet Society: Report on the Workshop on Reducing Internet Latency, December 2013. http://www.internetsociety.org/latency2013
20. Jiang, H., Liu, Z., Wang, Y., Lee, K., Rhee, I.: Understanding bufferbloat in cellular networks. In: Workshop on Cellular Networks: Operations, Challenges, and Future Design (CellNet), August 2012

21. Khademi, N., Ros, D., Welzl, M.: The New AQM Kids on the Block: Much Ado About Nothing? Technical report 434, University of Oslo, Department of Informatics (2013)
22. McKenney, P.E.: Stochastic fairness queueing. In: IEEE Infocom (1990)
23. Nagle, J.: RFC970: On Packet Switches with Infinite Storage, December 1985
24. Nichols, K., Jacobson, V., McGregor, Ed., A., Iyengar, Ed., J.: RFC8289: Controlled Delay Active Queue Management, January 2018. https://tools.ietf.org/html/rfc8289
25. Nichols, K., Jacobson, V.: Controlling queue delay. ACM Queue **10**(5), 1–15 (2012)
26. Pan, R., Natarajan, P., Piglione, C., Prabhu, M.S., Subramanian, V., Baker, F., VerSteeg, B.: PIE: a lightweight control scheme to address the bufferbloat problem. In: IEEE Workshop on High Performance Switching and Routing (HPSR) (2013)
27. Ramakrishnan, K., Floyd, S., Black, D.: RFC3168: The Addition of Explicit Congestion Notification (ECN) to IP, September 2001
28. Shreedhar, M., Varghese, G.: Efficient fair queuing using deficit round robin. IEEE/ACM Trans. Netw. **4**(3), 375–385 (1996)
29. Suter, B., Lakshman, T.V., Stiliadis, D., Choudhury, A.K.: Buffer management schemes for supporting TCP in gigabit routers with per-flow queueing. IEEE J. Sel. Areas Commun. **17**(6), 1159–1169 (1999)
30. Varga, A.: INET-2.2 released, August 2013. http://inet.omnetpp.org/
31. Varga, A.: OMNeT++ 4.3.1 released, September 2013. http://www.omnetpp.org/
32. Wand, S.: Network Simulation Cradle (2012). http://research.wand.net.nz/software/nsc.php
33. White, G., Pan, R.: RFC8034: Active Queue Management (AQM) Based on Proportional Integral Controller Enhanced (PIE) for Data-Over-Cable Service Interface Specifications (DOCSIS) Cable Modems, February 2017. https://tools.ietf.org/html/rfc8034
34. White, G.: Data-over-Cable Service Interface Specification - MAC and Upper Layer Protocols Interface Specification. Technical report CM-SP-MULPIv3.1-I01-131029, Cable Television Laboratories, Inc., October 2013
35. White, G.: Active Queue Management in Docsis 3.X Cable Modems. Technical report, Cable Television Laboratories, Inc., May 2014
36. Zhang, L.: VirtualClock: a new traffic control algorithm for packet switching networks. In: ACM SIGCOMM (1990)

Modeling the Performance of ARQ Error Control in an LTE Transmission System

Udo R. Krieger[1(✉)] and B. Krishna Kumar[2]

[1] Fakultät WIAI, Otto-Friedrich-Universität,
An der Weberei 5, 96047 Bamberg, Germany
udo.krieger@ieee.org
[2] Department of Mathematics, Anna University,
Chennai 600025, India
drbkkumar@hotmail.com

Abstract. We consider the transmission of protocol data units in an LTE eNodeB in downlink direction and focus on the radio link control (RLC) and hybrid Automatic-Repeat-Request functionality at layer 2 of the transmission system. We model the associated window flow control of RLC frames in terms of an open queueing network with two stations and describe RLC frames waiting for acceptance by the flow control window by repeated objects collected in an orbit in front of this multi-server queueing network. We describe the correlated arrivals of frames by a general Markovian arrival process and the transfer of data transport blocks along the orthogonal frequency-division multiplexing channels by state-dependent service processes and varying channel capacities. We derive a versatile finite Markovian queueing model and show that its steady-state distribution can be computed in terms of a level-independent QBD process. Then we determine the basic performance metrics of the system in terms of the latter steady-state distribution.

Keywords: LTE performance analysis · Window flow control
Hybrid ARQ · MAP · Level-independent QBD

1 Introduction

At present mobile cloud services and multimedia applications generate the majority of current traffic load in the Internet and demand a fast evolution of the underlying mobile networks from Long Term Evolution (LTE) and LTE-Advanced to dynamically evolving 5G technologies. Considering the protocol stack of LTE transport systems, many improvements established in recent years have concerned layer 1 and 2 functionalities. They are related to improved transmission techniques such as hybrid Automatic-Repeat-Request (ARQ), advanced adaptive space-time coding methods that are dependent on channel states as well as multiple-input/multiple-output (MIMO) and beamforming techniques.

The paper is devoted to the transmission of radio link control (RLC) protocol data units (PDUs) in an LTE base station called eNodeB in downlink direction.

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 202–217, 2018.
https://doi.org/10.1007/978-3-319-74947-1_14

It focuses on the RLC error control and hybrid ARQ functionality as basic control method at layer 2 (cf. [3,15]). We model the flow control scheme of radio link control frames that is governed by an ARQ selective repeat (SR) policy in terms of an open queueing network (QNW) with two multi-server stations. We describe those frames waiting for acceptance by the flow control window by retrying objects that are collected in an orbit in front of this multi-server queueing network. The correlated arrivals of RLC frames are modelled by a general Markovian arrival process (MAP) which can be mapped easily to real data (cf. [2,7]). The transport of radio data blocks along the orthogonal frequency-division multiplexing (OFDM) channels is described by state-dependent service processes with varying channel capacities. In this regard the sketched model extends our past modeling approaches that either did not map the correlated arrival process to an adequate point process or considered a simpler closed queueing network of the flow control process (cf. [12]). In this regard our open queueing network and its underlying continuous-time Markov chain (CTMC) also differ substantially from the discrete-time models of ARQ and hybrid ARQ systems developed in recent years by Zorzi et al. and others (cf. [1,14] and references therein). There the main focus was given by the performance of control algorithms in lower layers of a wireless network whereas our approach studies the intertwining between the RLC error control scheme at layer 2 of an LTE transmission system and a correlated packet arrival process, e.g. a TCP flow arising from advanced cloud and multimedia applications. It allows us to understand the impact of the layer 2 functionality on the quality-of-service (QoS) and quality-of-experience (QoE) performance of mobile cloud applications in a better way (cf. [5]).

We derive a versatile finite Markovian queueing model and show that its steady-state distribution can be computed in terms of a quasi birth-and-death (QBD) process. Then we determine the basic performance metrics of the system in terms of the latter steady-state distribution taking into account its matrix-geometric closed form.

Looking at the requirements of tactile Internet based on software-defined 5G wireless networks in the near future, we may argue that gaining insight on the performance of basic layer 2 techniques such as ARQ selective repeat (ARQ-SR) and hybrid ARQ error control of RLC frames is a fundamental step towards a self-optimizing resource assignment and an adaptive orchestration methodology. Analytic models constitute a first simple step in that direction to optimize delay-throughput and QoE performance of such challenging architectures. In this regard the presented modeling approach may provide a first analytic guideline how to reach these challenging goals.

The paper is organized as follows. First we present a brief description of the RLC error control and hybrid ARQ scheme and its modeling by a queueing network. Regarding the error control in the considered LTE transmission system, we derive a Markovian model on a finite state space and compute its steady-state vector in Sect. 3. In Sect. 4 we determine the basic performance metrics of the model. Finally we discuss some conclusions of our modeling approach.

2 Modeling ARQ Error Control of RLC Frames on Logical Channels in an LTE eNodeB

In the following we regard the protocol stack of an LTE network at layers 1 to 3 and the transmission of the corresponding protocol data units that are arising from some advanced application on top of the IP network layer, see Fig. 1 (cf. [3]). The latter are associated with an LTE bearer service, first segmented and encapsulated into PDUs of the protocol data convergence protocol (PDCP) by some compression and encryption transformations and then disassembled and converted into radio link layer control PDUs. The objects of these RLC streams are multiplexed into MAC PDUs. In our paper we focus on the transmission of RLC frames as part of MAC PDUs in an LTE base station called eNodeB in downlink direction. We study the logical protection of transported RLC frames at layer 2 of the transmission system at the LTE air interface (cf. [3,15]).

2.1 Error Control of RLC Frames in an LTE Transmission System

In an LTE eNodeB IP packets are transformed during the complex downlink processing steps such that MAC PDUs are formed from this payload (SDU) as shown in Fig. 1 (cf. [3]). The latter comprise the potentially segmented IP payload in terms of RLC frames that are transferred along a logical channel, namely a data traffic channel (DTCH). These RLC PDUs are then mapped to transport channels on the MAC layer, namely the downlink shared channels (DL-SCHs) and to one or multiple physical downlink shared channels (PDSCHs) at the physical layer in terms of transport blocks, see Figs. 1 and 2 (cf. [3]). Thereafter, the generated physical layer PDUs are transmitted along the latter channels by digital signal streams towards a mobile client.

In an LTE eNodeB hybrid ARQ schemes such as a parallel version of the simple stop-and-wait policy (ARQ-SW) combined with smart adaptive channel coding are applied at the LTE air interface to the MAC PDUs (cf. [3], [10, Chap. 3.4.4], [11]). During the preparation of those transport blocks on physical channels further coding elements such as forward error correction (FEC) code words are attached to the original PDUs at layers 2 and 1 and appropriately encoded by space-time coding techniques.

Following Zorzi's line of research [1,14], we study here the effective ARQ selective repeat policy that can be used at the RLC sublayer to retransmit RLC PDUs. It is applied in the acknowledged mode to a data flow traversing a logical channel. It complements the hybrid ARQ functionality with its advanced adaptive FEC techniques that is used at the physical and MAC layers for the associated PDUs (see Fig. 2, cf. [3]). We investigate the latter ARQ window flow control protocol at the sender side by analytical means based on an open queueing network. More precisely, we model the associated transformations from the logical RLC channels to the transport and physical channels and the associated transport process of RLC PDUs by a queueing network with three stations. This derived LTE transmission model describing the flow controlled transport of RLC frames consists of two building blocks, namely, an inner block comprising a multi-server queueing network with two substations and an outer substation with K

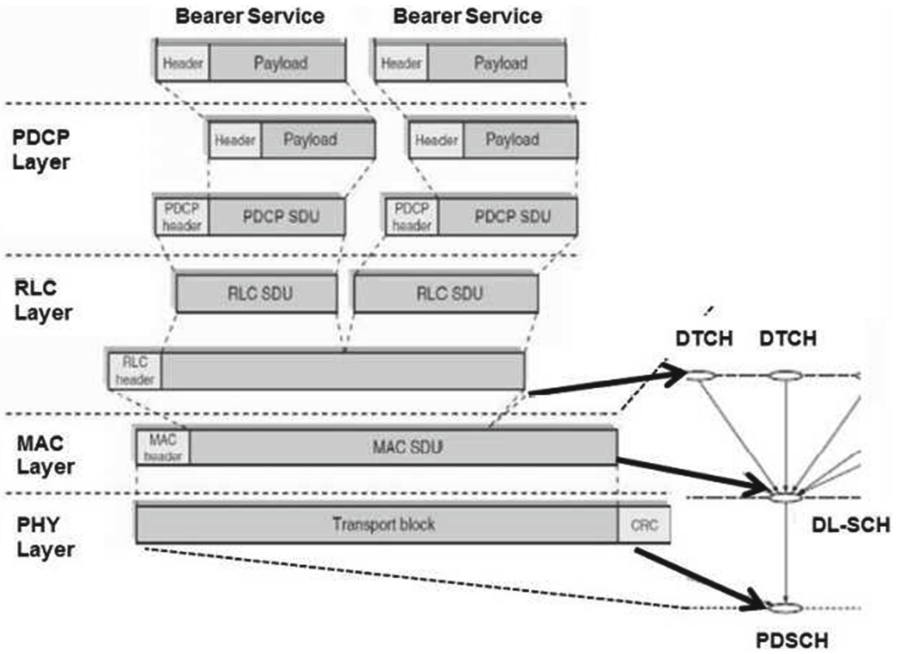


Fig. 1. Data processing in a protocol stack of an LTE eNodeB.

places and a single server with exponentially distributed service times. This is, of course, a first rough abstraction of many complicated processing steps applied to RLC PDUs by the LTE eNodeB stack that cannot be captured in full detail by any queueing model. We assume that the flow control window of the selective repeat scheme which is applied to aggregated streams of RLC frames comprises a finite number M of PDUs. They are served by J_1 parallel servers with exponentially distributed service rates μ_1 that describe a set of corresponding downlink shared transport channels (DL-SCHs) at the MAC layer. Completed frames are forwarded to the second substation consisting of a queue served by J_2 service stations. The latter describe the mapping of the RLC frames embedded into MAC frames from transport channels onto the physical downlink shared OFDM channels assigned by the radio resource control (RRC) protocol (cf. [3]). The employed hybrid ARQ scheme induces a variable FEC part on each MAC PDU which is encoded by symbol sequences corresponding to the associated transport blocks on the multiple-input multiple-output (MIMO) channels of the LTE air interface based on spatial multiplexing techniques. Thus, we model these complex structures by random service times with state-dependent rates $\mu_{2,s}$. The physical channel capacity is described in this simplistic abstraction by a modulating Markovian channel environment J_C with $S \in \mathbb{N}$ states and an irreducible generator matrix $Q_C \in \mathbb{R}^{S \times S}$, e.g. by means of a Gilbert-Elliott error model or its various extensions, in accordance with previous studies (cf. [1,11,14,16,19]).

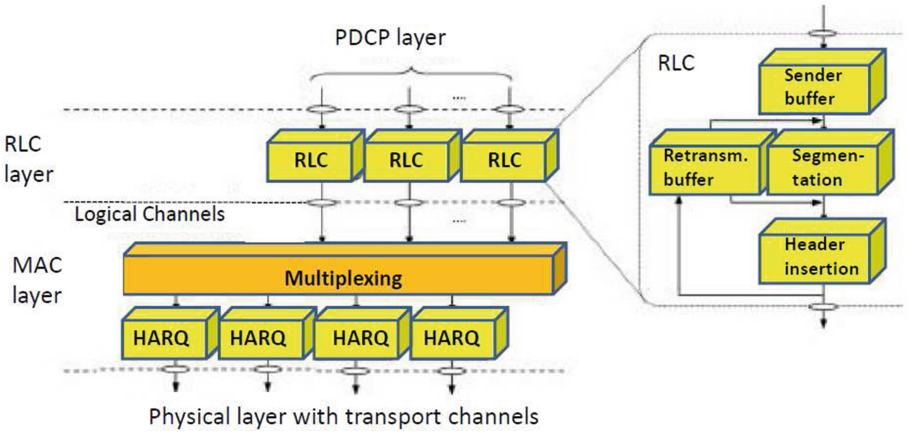


Fig. 2. RLC frame processing along downlink channels of an LTE eNodeB.

2.2 Derivation of a Queueing Model for ARQ-SR Error Control

We consider a stream of certain application packets arriving at a considered LTE eNodeB. They are entering its protocol stack at layer 2 after some processing steps as discussed previously. As these RLC PDUs of a tagged stream may be multiplexed into a single MAC PDU during the mapping of a logical downlink traffic channel onto a transport channel in the LTE transport system, we may model the transfer of the corresponding transport blocks of this tagged and other RLC streams included in the MAC PDU on the associated physical channels by a service period of related mean length $1/\bar{\mu}_2$. We assume that the service distribution of the PDU transmission follows an exponential distribution and that the service times of all frames are independent.

As the radio channel is subject to signal impairments, a transmission may fail with certain probabilities q_i that depend on the state $i \in \{1, \ldots, S\}$ of the channel environment J_C (cf. [8,11]). We further adopt this dependency on the channel states i for the service periods of a PDU. Thus, we assume that the exponential service rates are also state-dependent objects and described in state i by the service rate $\mu_{2,i} > 0$. We suppose that a failed transport of a tagged PDU occurs with probability $q_i > 0$. It models the aggregated negative acknowledgments of all those erroneous transport blocks associated with the transferred PDUs that are received by the multiple stop-and-wait mechanisms of the hybrid ARQ scheme in the MAC layer. A successful service completion occurs with probability $1 - q_i$. It means that the rate of a successful transport service in state i is given by $\mu_{2,i}(1 - q_i) > 0$. Then the frame forever leaves the inner subsystem of the QNW describing the successful frame transmission along the LTE air interface. Hence, a PDU slot becomes available in the flow control window. Furthermore, the overall mean service rate is given by $\bar{\mu}_2 = \sum_{i=1}^{S} \eta_i \cdot \mu_{2,i}$ where η denotes the unique steady-state distribution of J_C, i.e. $\eta^t \cdot Q_C = 0, \eta^t \cdot e = 1$. The incoming RLC frames of a specific stream of packets

and their correlated interarrival times are modelled by an ordinary Markovian arrival process (MAP) controlled by a finite Markovian environment J_A with $E \in \mathbb{N}$ states $d \in \{1, \ldots, E\}$ and a generator matrix $Q_A \in \mathbb{R}^{E \times E}$ (cf. [2,7]). It is independent of the modulating environment J_C of the channel. We can assume that this arrival process is the result of the superposition of independent MAP arrival streams of those processes that describe the integration of packets into a single MAC PDU. Further, we assume that only single arrivals occur with the nonnegative rate matrix $0 \leq D_1 \in \mathbb{R}^{E \times E}$. Here and in the following we apply the ordering relations $0 \leq v, 0 < v$ to vectors or matrices v with an element-by-element meaning. It is paraphrasing the nonnegativity or positivity of all individual elements. Internal phase shifts on E without any arrivals are described by the regular Metzler-Leontief matrix $D_0 \in \mathbb{R}^{E \times E}$, i.e. $-D_0$ is an invertible M-matrix with positive diagonal elements $-D_0 \cdot e = D_1 \cdot e > 0$ where $e \in \mathbb{R}^E$ is the vector of all ones. Then the generator of the random environment J_A is determined by $Q_A = D_0 + D_1 \in \mathbb{R}^{E \times E}$. We assume that Q_A is irreducible. The mean number of arrivals is determined by $\lambda = \xi \cdot D_1 \cdot e$ where $0 < \xi \in \mathbb{R}^E$ denotes the steady-state vector of the modulating environment J_A, i.e. $\xi^t \cdot Q_A = 0, \xi^t \cdot e = 1$ (cf. [7]).

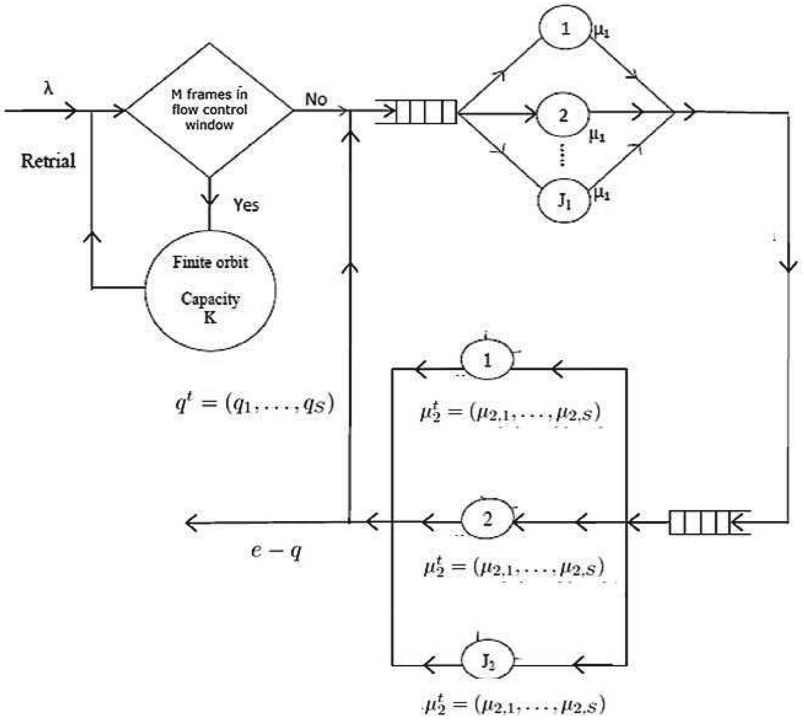


Fig. 3. Queueing network model of the error control for RLC PDUs that is based on an ARQ-SR controlled frame transport along a logical downlink traffic channel at an LTE eNodeB.

If an arriving RLC frame comes in and there is an idle slot in the flow control window of size M it can enter the queueing network of the RLC transport system. If all slots are occupied it is entering an orbit queue and waits for an empty slot in the ARQ flow control window. The status is periodically inspected. This process is modelled by an orbit with $K \in \mathbb{N}$ places including a single server with exponentially distributed independent service times with common mean $1/\nu$. The service time is modelled as a multiple of the basic transmission slot, i.e. the length of the transmission time interval (TTI) of the LTE transport system corresponding to one millisecond, and approximates the feedback delay of the hybrid ARQ status reports along the transport channels (cf. [3,15]).

The capacity of the overall queueing network with its two inner stations and the outer orbit covers $\tau = M + K \in \mathbb{N}$ RLC PDUs. If all slots are occupied incoming frames are lost in our model. The overall queueing system of the RLC error control is depicted in Fig. 3.

3 A Markovian Model of the ARQ-SR Error Control in an LTE Transmission System

3.1 Structure of the Markov Chain

Following the investigation of a related multi-server queueing network with retrials by Kumar et al. [9], we can describe the state $X(t)$ of the derived queueing model of ARQ-SR error control in LTE at time $t \geq 0$ by a vector process

$$X(t) = (O(t), R(t), P(t), J(t)) \in \Sigma,\ t \geq 0,$$

on the finite state space $\Sigma \subset \mathbb{N}_0^5$. Here $O(t) = n \in \{0, 1, \ldots, K\}$ denotes the RLC frames waiting in the orbit station for admittance to the flow control window $W(t) = R(t) + P(t)$ of size $M > 0$. Without loss of generality we assume $M \geq J_1 + J_2 \geq 3$. $R(t) = l \in \{0, 1, \ldots, M\}$ represents the number of frames in the first substation of the inner model with J_1 servers that describes the PDU processing on the transport channels of the LTE transmission system. $P(t) = k = m - l \in \{0, 1, \ldots, M\}$ denotes the number of frames in the second substation of the LTE transmission system that describes the RLC PDU transport along the physical LTE channels. We impose here the condition $0 \leq R(t) = l \leq m \leq M$. Then $W(t) = R(t) + P(t) = m \in \{0, 1, \ldots, M\}$ frames are in the queueing network modeling the RLC PDU transport along the air interface at each time t. It is limited by the maximal number of frames $M \in \mathbb{N}$ in the flow control window, i.e. $W(t) = m$ represents the state of the flow control window at time $t > 0$.

In the following let $I_S, I_E, I_{SE}, I_{M+1-l}$ denote identity matrices of sizes S, E, SE, and $M + 1 - l$, respectively.

$J(t) = (J_C(t), J_A(t))$ denotes the overall state of the modulating environment comprising the state $J_C(t)$ of the channel impairment model and the modulator $J_A(t)$ of the arrival process. It is governed by the common generator matrix $Q_J = Q_C \oplus Q_A = Q_C \otimes I_E + I_S \otimes Q_A$ where $\oplus$ denotes the Kronecker sum

defined in terms of the Kronecker product $\otimes$ of the individual generator matrices of J_C and J_A, respectively.

The five-dimensional vector process $X(t) = (O(t), R(t), P(t), J(t)) = x \in \Sigma, t \geq 0$, is a continuous-time Markov chain on a finite state space $\Sigma \subset \{0, \ldots, K\} \times \{0, \ldots, M\}^2 \times \{1, \ldots, S\} \times \{1, \ldots, E\}$ since all components are determined by a state-dependent set of independent memoryless distributions of sojourn times in the states $x \in \Sigma$.

As the restriction $0 \leq W(t) = R(t) + P(t) = m \leq M$ holds, we realize that for given $R(t) = l \in \{0, 1, \ldots, M\}$ we get state-dependent subsets $S_l = \{0, 1, \ldots, n_l - 1\}$ of size $n_l = M - l + 1 \in [1, M + 1] \subset \mathbb{N}$ as realizations of $P(t)$. These varying dimensions govern the structure of the underlying generator matrix Q of the CTMC $X(t)$. Then we realize that $W(t) = (R(t), P(t)) = (l, m - l)$ varies in a set of $w = \sum_{l=0}^{M} n_l = \frac{(M+1)(M+2)}{2}$ states and, hence, the state space Σ has the finite size $\kappa = \frac{1}{2}(M + 1)(M + 2)(K + 1)SE$.

The associated generator matrix $Q \in \mathbb{R}^{\kappa \times \kappa}$ of the CTMC $X(t)$ has a block tridiagonal structure:

$$Q = \begin{pmatrix} Q_{0,0} & Q_{0,1} & \mathcal{O} & \mathcal{O} & \ldots & \ldots & \mathcal{O} \\ Q_{1,0} & Q_{1,1} & Q_{1,2} & \mathcal{O} & \ddots & \ddots & \vdots \\ \mathcal{O} & Q_{2,1} & Q_{2,2} & Q_{2,3} & \ddots & \ddots & \vdots \\ \mathcal{O} & \mathcal{O} & Q_{3,2} & Q_{3,3} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \mathcal{O} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & Q_{K-1,K} \\ \mathcal{O} & \ldots & \ldots & \ldots & \mathcal{O} & Q_{K,K-1} & Q_{K,K} \end{pmatrix} \tag{1}$$

All blocks $Q_{i,j}, 0 \leq i, j \leq K$, have the same size $\kappa' = \frac{1}{2}(M + 1)(M + 2)SE$ and state-dependent internal structures. Here and subsequently, $\mathcal{O}$ denotes block matrices with all zeros whose dimension is determined by those block matrices of the surrounding context. $X(t)$ represents a quasi-birth-and-death (QBD) process where $O(t) = n \in \{0, \ldots, K\}$, is the level. $(R(t), P(t), J(t)) \in \Sigma_n$ is the phase process where Σ_n stems from the projection of Σ for fixed n.

The diagonal block matrices $Q_{i,i}, 0 \leq i \leq K$, represent again block tridiagonal matrices of size κ'. We first consider

$$Q_{0,0} = \begin{pmatrix} {C^{(0)}}_{0,0} & {A^{(0)}}_{0,1} & \mathcal{O} & \mathcal{O} & \ldots & \mathcal{O} \\ {B^{(0)}}_{1,0} & {C^{(0)}}_{1,1} & {A^{(0)}}_{1,2} & \mathcal{O} & \ddots & \vdots \\ \mathcal{O} & {B^{(0)}}_{2,1} & {C^{(0)}}_{2,2} & {A^{(0)}}_{2,3} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \mathcal{O} \\ \vdots & \ddots & \ddots & \ddots & \ddots & {A^{(0)}}_{M-1,M} \\ \mathcal{O} & \ldots & \ldots & \mathcal{O} & {B^{(0)}}_{M,M-1} & {C^{(0)}}_{M,M} \end{pmatrix} \tag{2}$$

with $n = 0$ elements in the orbit. Then $R(t) = l \in \{0, \ldots, M\}$ varies the number l of frames in the first substation of the LTE transmission system.

The upper diagonal blocks $A^{(0)}{}_{l,l+1}, 0 \leq l \leq M-1$, are $n_l SE \times n_{l+1} SE = (M-l+1)SE \times (M-l)SE$ matrices and correspond to RLC PDU arrivals according the internal MAP structure J_A with rate matrix $0 < D_1 \in \mathbb{R}^{E \times E}$, i.e.

$$A^{(0)}{}_{l,l+1} = \begin{pmatrix} M^{(0)}{}_{0,0} & \mathcal{O} & \mathcal{O} & \dots & \mathcal{O} \\ M^{(0)}{}_{1,0} & M^{(0)}{}_{1,1} & \mathcal{O} & \ddots & \vdots \\ \mathcal{O} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \mathcal{O} \\ \mathcal{O} & \dots & \mathcal{O} & M^{(0)}{}_{M-l-1,M-l-2} & M^{(0)}{}_{M-l-1,M-l-1} \\ \mathcal{O} & \dots & \mathcal{O} & \mathcal{O} & M^{(0)}{}_{M-l,M-l-1} \end{pmatrix} \tag{3}$$

where we get for $k \in \{0, \dots, M-l-1\}$:

$$M^{(0)}{}_{k,k} = I_S \otimes D_1 \in \mathbb{R}^{SE \times SE} \tag{4}$$

The lower diagonal blocks are given for $s_k = \min(k, J_2)$, $k \in \{0, \dots, M-l\}$, by matrices

$$M^{(0)}{}_{k,k-1} = s_k \cdot Z_S(q, \mu_2) \otimes I_E \in \mathbb{R}^{SE \times SE} \tag{5}$$

with the diagonal matrix

$$Z_S(q, \mu_2) = \begin{pmatrix} q_1\mu_{2,1} & 0 & \dots & 0 \\ 0 & q_2\mu_{2,2} & \ddots & \vdots \\ \vdots & \ddots & q_{S-1}\mu_{2,S-1} & 0 \\ 0 & \dots & 0 & q_S\mu_{2,S} \end{pmatrix} \in \mathbb{R}^{S \times S}. \tag{6}$$

It represents the departure rates $\mu_2^t = (\mu_{2,1}, \dots, \mu_{2,S})$ of the second substation of the LTE transmission system that are modulated by the state s of the channel environment J_C and its state-dependent feedback probabilities $q^t = (q_1, \dots, q_S)$ of an unsuccessful frame transmission.

The lower diagonal blocks $B^{(0)}{}_{l,l-1}$, $l \in \{1, \dots, M\}$, describe the departure events $R(t) = l$ to $R(t) = l-1$. They correspond to the occurrence of a service completion of the first station among the $r_l = \min(l, J_1)$ active servers with rates $r_l\mu_1$. Then these rates are taken into account by the upper diagonal blocks of this block matrix

$$\begin{aligned} B^{(0)}{}_{l,l-1} &= \begin{pmatrix} \mathcal{O} & r_l\mu_1 \otimes J_{0,1} & \mathcal{O} & \dots & \mathcal{O} \\ \mathcal{O} & \mathcal{O} & r_l\mu_1 \otimes J_{1,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathcal{O} \\ \mathcal{O} & \mathcal{O} & \dots & \mathcal{O} & r_l\mu_1 \otimes J_{M-l,M-l+1} \end{pmatrix} \\ &= \left(\mathcal{O}, r_l\mu_1 I_{M+1-l} \otimes I_S \otimes I_E\right) \in \mathbb{R}^{n_l SE \times n_{l-1} SE} \end{aligned} \tag{7}$$

where $J_{k,k+1} = I_{SE} = I_S \otimes I_E, 0 \leq k \leq M-l$.

The diagonal block matrices ${C^{(0)}}_{l,l} \in \mathbb{R}^{n_l SE \times n_l SE}, 0 \le l \le M$, describe the changes within the state variable $P(t) = k \in \{0, \ldots, M-l\}$ given $R(t) = l$ and reflect a tridiagonal structure of the local QBD behavior

$$C^{(0)}{}_{l,l} = \begin{pmatrix} {G^{(0,l)}}_{0,0} & \mathcal{O} & \mathcal{O} & \cdots & \mathcal{O} \\ {F^{(l)}}_{1,0} & {G^{(0,l)}}_{1,1} & \mathcal{O} & \ddots & \vdots \\ \mathcal{O} & {F^{(l)}}_{2,1} & {G^{(0,l)}}_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \mathcal{O} \\ \mathcal{O} & \cdots & \mathcal{O} & {F^{(l)}}_{M-l,M-l-1} & {G^{(0,l)}}_{M-l,M-l} \end{pmatrix}. \quad (8)$$

The lower diagonal blocks ${F^{(l)}}_{k,k-1}, k = 1, \ldots, M-l$, correspond to departures $P(t) = k$ to $P(t) = k-1$ from the second LTE station after a successful frame transmission with probability $1 - q_s$ which is modulated by the state s of the channel environment J_C. Thus, it is given by

$$\begin{aligned} {F^{(l)}}_{k,k-1} &= \begin{pmatrix} (1-q_1)s_k\mu_{2,1} & 0 & \cdots & 0 \\ 0 & (1-q_2)s_k\mu_{2,2} & \ddots & \vdots \\ \vdots & \ddots & (1-q_{S-1})s_k\mu_{2,S-1} & 0 \\ 0 & \cdots & 0 & (1-q_S)s_k\mu_{2,S} \end{pmatrix} \\ &\quad \otimes I_E \\ &= s_k \cdot Z(e-q, \mu_2) \otimes I_E \in \mathbb{R}^{SE \times SE} \end{aligned} \quad (9)$$

with $s_k = \min(k, J_2)$ active servers in the second substation. Here $e \in \mathbb{R}^S$ is the vector of all ones.

The diagonal block matrices ${G^{(0,l)}}_{k,k}, 0 \le k \le M-l-1 = n_l - 2$, of an idle orbit with $n = 0$ comprise the rates $S_0 = Q_C \in \mathbb{R}^{S \times S}$ of the channel environment $J_C(t)$, the rates $D_0 \in \mathbb{R}^{E \times E}$ of the modulating MAP arrival process $J_A(t)$ among those states of J_A without incoming frames, and a compensating diagonal vector $\Delta_{n,l,k}$. The latter is such that $Qe = 0$ is satisfied where we disregard the diagonal elements $-D_1 e = D_0 e$ of D_0 and S_0. Then we may represent ${G^{(0,l)}}_{k,k}, 0 \le k \le M-l-1$, in the form

$${G^{(0,l)}}_{k,k} = S_0 \oplus D_0 + \mathrm{diag}(\Delta_{0,l,k}). \quad (10)$$

There is no arrival for $P(t) = k = M - l$ since an incoming frame is blocked and sent to the orbit if $O(t) = n < K$ holds. Incrementing $O(t)$ it is taken into accounted in terms of $Q_{n,n+1}$.

Apart of the variability of the diagonal compensation ($\Delta_{n,l,k}$) all matrices $Q_{n,n}$, $0 \le n < K$, have the same structure. Only if the orbit is full in the last state $O(t) = K$ and the flow control window is also fully occupied, i.e. $W(t) = R(t) + P(t) = M$, in the blocking states $B = \{\chi = (l, M-l) \mid 0 \le l \le M\}$, an incoming frame is blocked from the system and lost. This event is reflected by the modified structure

$${G^{(K,l)}}_{M-l,M-l} = S_0 \oplus (D_0 + D1) + \mathrm{diag}(\Delta_{K,l,M-l}) \quad (11)$$

of the diagonal blocks in those states χ with a corresponding compensating diagonal vector $(\Delta_{K,l,M-l})$.

The upper diagonal block matrices $Q_{i,i+1}, 0 \leq i \leq K-1$, of Q cover the arrivals of the MAP process to the orbit. It is accounted for by the state variable $O(t) = n < K$ which is incremented to $O(t) = n+1$. It is starting with the idle orbit $n = 0$ and ends with the last empty slot $n = K-1$ of the orbit. It is driven by a full flow control window with the state $W(t) = R(t) + P(t) = M$ of the inner queueing system in the overall network. Only in this state the MAP arrival is redirected to the orbit and the residual states generate no increment of $O(t) = n < K$. These events are represented by the block matrix

$$Q_{n,n+1} = \begin{pmatrix} H_0 & \mathcal{O} & \dots & \mathcal{O} \\ \mathcal{O} & H_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathcal{O} \\ \mathcal{O} & \dots & \mathcal{O} & H_M \end{pmatrix}, \ 0 \leq n \leq K-1. \tag{12}$$

It is described by a set of block diagonal matrices H_l along the diagonal

$$H_l = \begin{pmatrix} 0 \otimes I_{M-l} \otimes I_S \otimes I_E & \mathcal{O} \\ \mathcal{O} & I_S \otimes D_1 \end{pmatrix} \in \mathbb{R}^{n_l SE \times n_l SE}, \ 0 \leq l \leq M-1 \tag{13}$$

$$H_M = \left(I_S \otimes D_1 \right) \in \mathbb{R}^{SE \times SE}. \tag{14}$$

If $R(t) = l \in \{0, \dots, M\}$ holds then only in the last state $0 \leq P(t) = M - l \leq M$, i.e. the last state $W(t) = R(t) + P(t) = M$ of the flow control window, this redirection to the orbit occurs.

The lower diagonal block matrices $Q_{n,n-1}, 1 \leq n \leq K$, represent departures from the orbit in state $O(t) = n$ and arrivals to the first inner station of the LTE transmission system. They are accounted by state variable $R(t) = l < M$ which is incremented by one arrival to $R(t) = l+1 \leq M$ with rates ν of the exponentially distributed flow-control inspection interval. If there are $W(t) = R(t) + P(t) = l + k = m < M$ frames in the inner system in the case $O(t) = n \leq K$, such an arrival is possible, otherwise we get again a redirection to the orbit in state $W(t) = R(t) + P(t) = M$. Then the lower diagonal block matrix $Q_{n,n-1}, 1 \leq n \leq K$ is characterized by an upper block tridiagonal matrix

$$Q_{n,n-1} = \begin{pmatrix} \mathcal{O} & I_{0,1} & \mathcal{O} & \dots\dots & \mathcal{O} \\ \mathcal{O} & \mathcal{O} & I_{1,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \mathcal{O} \\ \vdots & \ddots & \ddots & \ddots & I_{M-1,M} \\ \mathcal{O} & \dots & \dots & \mathcal{O} & 0 \otimes I_S \otimes I_E \end{pmatrix} \tag{15}$$

which is identical for all $n \in \{1,\ldots,K\}$. The matrices $I_{l,l+1}, 0 \leq l \leq M-1$, represent the arrival to the flow control window $W(t) = R(t) + P(t) = l + k$ with the exception of the final state $R(t) = l, P(t) = M - l$. They are given by

$$I_{l,l+1} = \begin{pmatrix} \nu \cdot I_{n_{l+1}} \\ \zeta^t \end{pmatrix} \otimes I_S \otimes I_E \in \mathbb{R}^{n_l SE \times n_{l+1} SE} \tag{16}$$

with identity matrices $I_{n_{l+1}}, I_S, I_E$ of size $n_{l+1} = M - l, S, E$, respectively, and a row vector $\zeta^t = (0,\ldots,0)$ with $M - l$ zeros.

3.2 Computation of the Steady-State Distribution

The Markovian queueing network $\{X(t), t \geq 0\}$ modeling the ARQ-SR error control scheme in the LTE transmission system with a MAP arrival process has a QBD structure on a finite state space Σ. Imposing the condition on the irreducibility of the generator matrix $Q_J = Q_C \oplus Q_A$ of the modulating environment $J(t) = (J_C(t), J(_A(t))$, we may conclude that the CTMC $X(t)$ has an irreducible generator matrix Q with a block tridiagonal structure and a single, zero eigenvalue $\rho(Q) = 0$. Thus, the corresponding steady-state vector $\Pi \in \mathbb{R}^{\kappa \times \kappa}$ is the unique solution of the following balance equations with the zero vector $0 \in \mathbb{R}^{\kappa}$ as right hand side and the additional normalization condition:

$$\Pi^t \cdot Q = 0 \qquad e^t \cdot \Pi = 1 \tag{17}$$

The QBD structure of Q on the finite state space implies that there exists a superposition of two matrix-geometric terms that constitute the steady-state vector Π (cf. [7,9,13,17,18]). To describe its structure, we partition the steady-state vector according to the levels $O(t) = n$ of the CTMC $X(t)$ as

$$\Pi^t = (\pi_0^t, \pi_1^t, \ldots, \pi_K^t).$$

According to the phase process $(R(t), P(t), J(t)) = (R(t), P(t), J_C(t), J_A(t)) = (l, k, s, d)$ and a given fixed $n \in \{0,\ldots,K\}$ each component vector π_n on a level $O(t) = n$ is further decomposed into $\pi_n^t = (\pi_{(n,l,k,s,t)} \mid \forall (l,k,s,t) \in \Sigma_n)$.

Let us define the block matrices $A_2 = Q_{n,n-1}, n \in \{1,\ldots,K\}$, $A_1 = Q_{n,n}, n \in \{1,\ldots,K-1\}$, and $A_0 = Q_{n,n+1}, n \in \{1,\ldots,K-1\}$ which are independent of the level n. We define the matrix polynomial

$$\begin{aligned} &\mathcal{A}(z) = z^2 \cdot A_2 + z \cdot A_1 + A_0, \quad z \in \mathbb{C} \\ &\mathcal{A}(1) = A_2 + A_1 + A_0 \end{aligned} \tag{18}$$

and assume that $z \in \mathbb{C}$ exists such that $\det(\mathcal{A}(z)) \neq 0$.

Then we conclude that the steady-state vector Π can be represented by a sum of two scaled marix-geometric solutions in the following way (cf. [7]):

$$\pi_n^t = \begin{cases} \pi_0^t & , \quad n = 0 \\ \pi_1^t \cdot R^{n-1} + \pi_K^t \cdot \Phi^{K-n} & , \quad 1 \leq n \leq K \end{cases} \tag{19}$$

The basic matrices R, Φ are determined as minimal nonnegative solutions of the fundamental matrix-geometric equations:

$$\mathcal{O} = R^2 \cdot A_2 + R \cdot A_1 + A_0 \tag{20}$$

$$\mathcal{O} = A_2 + \Phi \cdot A_1 + \Phi^2 \cdot A_0 \tag{21}$$

Then a normalized variant of the steady-state vector $\Pi = (\pi_n)_n$ that satisfies

$$\pi_0^t \cdot e + \left(\pi_1^t \cdot \sum_{n=0}^{K-1} R^n + \pi_K^t \cdot \sum_{n=0}^{K-1} \Phi^n \right) \cdot e = 1 \tag{22}$$

fulfills the balance equations and normalization condition (17).

Hajek [4] has shown that there exists a related nonnegative matrix $G(R)$ satisfying the relation $0 = A_2 + A_1 \cdot G(R) + A_0 \cdot G^2(R)$ as its minimal nonnegative solution. The nonnegative solution matrices $R, G(R)$ are related in terms of the following relations which are used to compute the final solution Π (cf. [7]):

$$W(R) = A_1 + A_0 \cdot G(R) = A_1 + R \cdot A_2$$
$$R = A_0 \cdot [-W(R)]^{-1}, \quad G(R) = [-W(R)]^{-1} \cdot A_2$$

The missing initial terms $x^t = (\pi_0^t, \pi_1^t, \pi_K^t)$ of the matrix-geometric representation (19) of the steady-state vector Π can be computed as normalized solution $x \neq 0$, $x^t \cdot \widehat{A} = 0, x^t \cdot e = 1$, of the boundary system:

$$\widehat{A} = \begin{pmatrix} \widehat{A}_{0,0} & \widehat{A}_{0,1} & \mathcal{O} \\ \widehat{A}_{1,0} & \widehat{A}_{1,1} & \widehat{A}_{1,K} \\ \widehat{A}_{K,0} & \widehat{A}_{K,1} & \widehat{A}_{K,K} \end{pmatrix} \tag{23}$$

$$\widehat{A}_{0,0} = Q_{0,0}, \quad \widehat{A}_{1,0} = Q_{1,0}, \quad \widehat{A}_{K,0} = \Phi^{K-1} \cdot Q_{1,0}$$
$$\widehat{A}_{0,1} = Q_{0,1}, \quad \widehat{A}_{1,1} = W(R), \quad \widehat{A}_{K,1} = \Phi^K \cdot Q_{1,2}$$
$$\widehat{A}_{1,K} = R^{K-2} \cdot [Q_{1,2} + R \cdot Q_{K,K}], \quad \widehat{A}_{K,K} = \Phi \cdot Q_{1,2} + Q_{K,K}$$

4 Performance Metrics of the ARQ-SR Error Control Model

The basic performance parameters of the ARQ-SR error control system in steady state can be computed in terms of the matric-geometric solution vector $\Pi = (\pi_n)_n$. They comprise the following probabilistic metrics and basic mean values:

- the probability that there are $O = n \in \{0, \ldots, K\}$ frames waiting for transmission in the orbit queue: $p_n = \pi_n^t \cdot e$
- the probability that the orbit is idle: $p_0 = \pi_0^t \cdot e$
- the utilization probability that at least one RLC frame is waiting in the orbit: $p_U = 1 - p_0 = 1 - \pi_0^t \cdot e$

– the blocking probability of the overall system when there are $O = K$ frames in the orbit and $W = R + P = M$ in the flow control window:

$$p_{Loss} = \sum_{l=0}^{M} \sum_{s=1}^{S} \sum_{d=1}^{E} \pi_{(K,l,M-l,s,d)}$$

– the probability that there are $W = R + P = M$ in the flow control window and an arriving frame is either going to the orbit or rejected:

$$p_W = \sum_{n=0}^{K} \sum_{k=0}^{M} \sum_{s=1}^{S} \sum_{d=1}^{E} \pi_{(n,k,M-k,s,d)}$$

– the throughput of the overall flow control model based on the MAP input:

$$T = (1 - p_{Loss}) \cdot \lambda = \xi \cdot D_1 \cdot e \cdot (1 - p_{Loss})$$

– the mean number of frames waiting for flow control admittance in the orbit:

$$\mathbb{E}(O) = \sum_{n=0}^{K} n \cdot \pi_n^t \cdot e = \sum_{n=1}^{K} \left(n \cdot \pi_1^t \cdot R^{n-1} \cdot e + n \cdot \pi_K^t \cdot \Phi^{K-n} \cdot e \right)$$

– the mean number of frames in the LTE MAC processing queue:

$$\mathbb{E}(R) = \sum_{l=1}^{M} l \cdot \left(\sum_{n=0}^{K} \sum_{k=0}^{M-l} \sum_{s=1}^{S} \sum_{d=1}^{E} \pi_{(n,l,k,s,d)} \right)$$

– the mean number of frames in the LTE transmission queue:

$$\mathbb{E}(P) = \sum_{k=1}^{M} k \cdot \left(\sum_{n=0}^{K} \sum_{l=0}^{M-k} \sum_{s=1}^{S} \sum_{d=1}^{E} \pi_{(n,l,k,s,d)} \right)$$

– the mean number of frames in both the LTE MAC processing and transmission queues of the flow control window when there are $W = R + P = m \in \{0, \ldots, M\}$ frames processed:

$$\mathbb{E}(W_{FCW}) = \sum_{m=1}^{M} m \cdot \left(\sum_{l=0}^{m} \sum_{n=0}^{K} \sum_{s=1}^{S} \sum_{d=1}^{E} \pi_{(n,l,m-l,s,d)} \right)$$

– the mean sojourn time of a random frame in the flow control window until a successful transmission:

$$\mathbb{E}(S_{FCW}) = \frac{\mathbb{E}(W_{FCW})}{(1 - p_{Loss}) \cdot \lambda} = \frac{\mathbb{E}(W_{FCW})}{(1 - p_{Loss}) \cdot \xi \cdot D_1 \cdot e}$$

– the mean sojourn time of a random frame in the orbit:

$$\mathbb{E}(S_O) = \frac{\mathbb{E}(O)}{(1 - p_{Loss}) \cdot \lambda} = \frac{\mathbb{E}(O)}{(1 - p_{Loss}) \cdot \xi \cdot D_1 \cdot e}$$

5 Conclusion

The paper is devoted to the transmission of RLC PDUs in downlink direction at an LTE eNodeB. It focuses on the modeling of the ARQ error control functionality at layer 2 of the LTE protocol stack, namely the ARQ selective repeat policy complimented by a hybrid ARQ scheme (cf. [3,15]). We have modelled the flow control of radio link control frames on an LTE air interface in terms of a simple queueing network with three stations. Regarding the error control, we have described those RLC frames encapsulated by MAC PDUs that are waiting for the acceptance by the flow control window in terms of repeated objects. The latter are collected in an orbit in front of a queueing network with two multi-server stations that models the transport and physical channels of an LTE eNodeB. We have described the correlated arrivals of RLC frames by a general Markovian arrival process and the transfer of these PDUs by transport blocks along the OFDM channels by state-dependent service processes and varying channel capacities.

We have derived a basic finite Markovian queueing model as major outcome of our LTE modeling approach and stated the associated generator matrix with its block tridiagonal structure. It is shown that its steady-state distribution can be computed in terms of a level-independent QBD process. The latter is realized as a superposition of two matrix-geometric terms. Then we have determined the basic performance metrics of the ARQ error control model in terms of the latter steady-state distribution.

Our future investigations will concern further theoretical studies regarding the accuracy of the ARQ error control model of an LTE air interface and the derivation of optimal design parameters with respect to the flow control system. It will also be necessary to estimate the important parameters of our queueing model and to compare its plausibility based on channel simulations or emulations of the LTE air interface. (cf. [6]). Furthermore, extensive comparisons with existing hybrid ARQ models, e.g. by Zorzi et al. [1,14], will be required to assess the quality and benefits of the new modeling approach.

Acknowledgment. The authors appreciate the anonymous reviewers for their valuable comments which helped them to improve their presentation.

References

1. Badia, L., Levorato, M., Zorzi, M.: Markov analysis of selective repeat type II hybrid ARQ using block codes. IEEE Trans. Commun. **56**(9), 1434–1441 (2008)
2. Buchholz, P.: An EM-algorithm for MAP fitting from real traffic data. In: Kemper, P., Sanders, W.H. (eds.) TOOLS 2003. LNCS, vol. 2794, pp. 218–236. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45232-4_14
3. Dahlman, E., Parkvall, S., Skold, J.: 4G, LTE-Advanced Pro and The Road to 5G, 3rd edn. Academic Press, Cambridge (2016)
4. Hajek, B.: Birth-and-death processes on the integers with phases and general boundaries. J. Appl. Prob. **19**, 488–499 (1982)

5. Hoßfeld, T., Schatz, R., Krieger, U.R.: QoE of YouTube video streaming for current Internet transport protocols. In: Fischbach, K., Krieger, U.R. (eds.) MMB & DFT 2014. LNCS, vol. 8376, pp. 136–150. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-319-05359-2_10
6. IS-Wireless: 5G Toolset - Complete framework for research and education on 4G and 5G. Il. Pulawska 45b, 05–500 Piaseczno, Poland (2017)
7. Krieger, U.R., Naoumov, V., Wagner, D.: Analysis of a finite FIFO buffer in an advanced packet-switched network. IEICE Trans. Commun. **E81–B**(5), 937–947 (1998)
8. Krieger, U.R., Savoric, M.: Performance evaluation of ABR flow-control protocols in a wireless ATM network. In: Proceedings of Wowmom 1998, USA, pp. 73–82, October 1998
9. Kumar, K.B., et al.: Analysis of multiprogramming-multiprocessor retrial queueing systems with Bernoulli vacation schedule. Technical report, Anna University, Chennai, India (2017)
10. Kurose, J., Ross, K.W.: Computer Networking: A Top Down Approach, 7th edn. Pearson Education Ltd., Upper Saddle River (2017)
11. Liu, H., et al.: Error control schemes for networks: an overview. Mob. Netw. Appl. **2**, 167–182 (1997)
12. Liu, W., Krieger, U.R., Akyildiz, I.: An admission control model through outband signalling management. In: Proceedings of the IEEE INFOCOM 1992, Florence, Italy, May 1992
13. Latouche, G., Ramaswami, V.: A logarithmic reduction algorithm for quasi-birth-death processes. J. Appl. Prob. **30**, 650–674 (1993)
14. Rossi, M., Badia, L., Zorzi, M.: SR ARQ delay statistics on N-state Markov channels with non-instantaneous feedback. IEEE Trans. Wirel. Commun. **5**(6), 1526–1536 (2006)
15. Sauter, M.: From GSM to LTE-Advanced Pro and 5G: An Introduction to Mobile Networks and Mobile Broadband, 3rd edn. Wiley, Hoboken (2017)
16. Savoric, M., Wolisz, A., Krieger, U.R.: The impact of handover protocols on the performance of ABR flow-control algorithms in a wireless ATM network. Eur. Trans. Telecommun. **11**(4), 419–430 (2000)
17. Ye, J., Li, S.-Q.: Folding algorithm: a computational method for finite QBD processes with level-dependent transitions. IEEE Trans. Commun. **42**(2–4), 625–639 (1994)
18. Ye, Q.: On Latouche-Ramaswami's logarithmic reduction algorithm for quasi-birth-and-death processes. Stoch. Models **18**(3), 449–467 (2002)
19. Zorzi, M., Rao, R.R.: On the statistics of block errors in bursty channels. IEEE Trans. Commun. **45**(6), 660–667 (1997)

Catching Corner Cases in Network Calculus – Flow Segregation Can Improve Accuracy

Steffen Bondorf[1(✉)], Paul Nikolaus[2], and Jens B. Schmitt[2]

[1] National University of Singapore (NUS), Singapore, Republic of Singapore
bondorf@comp.nus.edu.sg
[2] Distributed Computer Systems (DISCO) Lab,
TU Kaiserslautern, Kaiserslautern, Germany

Abstract. Worst-case bounds on flow delays are essential for safety-critical systems. Deterministic network calculus is a methodology to compute such bounds. It is actively researched regarding its modeling capabilities as well as analysis accuracy and performance. We provide a contribution to the major part of the analysis: bounding the arrivals of cross flows. In particular, it has been believed that an aggregate view on cross flows outperforms deriving a bound for each cross flow individually. In contrast, we show that the so-called cross-flow segregation, can outperform the aggregation approach under certain conditions. We give a proof of concept, combine the alternative approaches into an analysis computing best bounds, and evaluate accuracy improvements as well as computational effort increases. To that end, we show that flows known to suffer from overly pessimistic delay bounds can see this pessimism reduced by double-digit percentages.

1 Introduction

In safety-critical, distributed systems that operate in public spaces, formal verification of performance guarantees is often a prerequisite for certification. For example, bounding the end-to-end delay of data transmissions is an integral demand of x-by-wire applications such as steer-by-wire or brake-by-wire. Thus, even small gains in its accuracy can be of importance for the outcome of a certification process. Deterministic Network Calculus (DNC) provides an analytical framework to compute worst-case delay bounds on data transmissions. For instance, it has found application in the avionics industry to demonstrate fulfillment of strict aircraft network requirements. To be precise, DNC's (min,+)-algebraic branch is often applied to analyze these avionics networks. A model bounding supply and demand of the network's forwarding resources is transformed to a (min,+)-equation that bounds a specific flow's end-to-end delay.

The step deriving an equation from the model has been steadily evolved in order to improve the accuracy of the resulting delay bound. The first such

This work has been conducted at the Distributed Computer Systems (DISCO) Lab, TU Kaiserslautern, Germany, with support of a Carl Zeiss Foundation grant.

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 218–233, 2018.
https://doi.org/10.1007/978-3-319-74947-1_15

improvement was to virtually separate the analyzed flow from all other flows in the network and to establish the worst-case scenario exclusively for this flow of interest (foi). This principle, separation of the foi, was shown to result in better bounds than the previous approach to anaylse the totality of the flows. However, subsequent work shows that only the analyzed flow of interest can be entirely removed while constructing a worst-case scenario. Efforts to implement this principle for the remainder of the network analysis were shown to result in an issue called segregation. In arbitrary multiplexing, i.e., no knowledge about the multiplexing of flows is assumed, attempting to separate multiple flows can result in situations where these simultaneously assume worst-case scenarios that are mutually exclusive in the real system. While resulting bounds remain valid, they are overly pessimistic. Therefore, the predominating objective of analyses is to aggregate all flows except the flow of interest. The literature provides a generic analysis procedure that maximizes aggregation and minimizes segregation when bounding the arrivals of the foi's cross flows. This procedure, known as Aggregate Arrival Bounding (AggrAB) [5], was extended by various analysis enhancements that can further increase aggregation of flows and improve delay bounds [1,6]. Based on the objective to maximize aggregation, an accurate and fast analysis was eventually presented [2]. The delay bounds it derives are shown to only deviate slightly from those bounds derived with the optimization branch of DNC [9]. However, while the optimization becomes computationally infeasible, the algebraic analysis scales well with increasing network size. Its execution time stays several orders of magnitude below the optimization's one [2].

In this paper, we focus on further closing the gap between algebraic and optimization-based DNC delay bounds. To be precise, we identify a peculiar corner case that is defined by a very specific combination of flow entanglements, resource demand, (left-over) resource supply and implemented DNC principle. Against the trend to prevent segregation of flows by aggregating them as much as possible, we prove that the reverse can actually result in better delay bounds. The mutually exclusive worst-case assumptions of two flows add less pessimism than the AggrAB does. We use this knowledge to contribute an arrival bounding method that catches these corner cases. It is modeled after [8] but as it relies solely on the PMOO principle, we call it SegrPMOO. Moreover, we combine it with AggrAB's latest evolutionary step, the Tandem Matching Analysis (TMA) [2], to an exhaustive search for best arrival bounds, TMA+SegrPMOO.

We evaluate our contribution by extending the numerical results providing insight on the gap between TMA and optimization. We show that the preconditions for these corner cases can be fulfilled fairly often during a network analysis, yet, in most cases it only helps to close the accuracy gap by less than 10%. A noteworthy exception to this observation can be found when investigating outliers. E.g., in the TMA evaluation's network of 20 devices, outliers are common and we show that the largest gaps to optimization can be reduced by over 30%.

The remainder of the paper is structured as follows: Sect. 2 provides the necessary background on DNC. In Sect. 3, we present the trend to improve bounds by aggregating flows. To that end, we provide the DNC analysis prin-

ciples implemented in TMA. Based on these insights, Sect. 4 contributes and proves the potential benefit of SegrPMOO. We extend the previously applied cross-traffic arrival bounding with it and evaluate our contribution in Sect. 5. We provide results on accuracy improvement as well as computational effort increase. Section 6 concludes the paper.

2 Deterministic Network Calculus Background

DNC is based on a simple network model [12] consisting of functions from the set

$$\mathcal{F}_0 := \left\{ f : \mathbb{R} \to \mathbb{R}_\infty^+ \mid f(0) = 0,\ \forall s \leq t : f(s) \leq f(t) \right\},$$
$$\mathbb{R}_\infty^+ := [0, +\infty) \cup \{+\infty\}.$$

Cumulative data arrivals are upper bounded in interval time:

Definition 1. Given a flow producing data according to function A in the time domain, a function $\alpha \in \mathcal{F}_0$ is an arrival curve for the flow iff

$$\forall t \forall d\ 0 \leq d \leq t : A(t) - A(t-d) \leq \alpha(d).$$

I.e., arrival curves bound the maximum data arrivals of a flow during any duration of length d. $\mathcal{F}_{\text{TB}} \subseteq \mathcal{F}_0$ is a commonly used set of curve shapes to bound flow arrivals. It bounds flows shaped to comply with token bucket regulation:

$$\mathcal{F}_{\text{TB}} := \{\gamma_{r,b} \mid \gamma_{r,b}(0) = 0, \gamma_{r,b}(d) = b + r \cdot d \quad \forall d > 0\}.$$

A server's forwarding of arriving data is lower bounded in interval time:

Definition 2. If the service provided by a server S for a given input function A results in an output function A', then S offers a service curve $\beta \in \mathcal{F}_0$ iff

$$\forall t : A'(t) \geq \inf_{0 \leq d \leq t} \{A(t-d) + \beta(d)\}.$$

A common set of curves $\mathcal{F}_{\text{RL}} \subseteq \mathcal{F}_0$ bounds service with a rate and a latency:

$$\mathcal{F}_{\text{RL}} := \{\beta_{R,T} \mid \beta_{R,T}(d) = \max\{0, R \cdot (d - T)\}\}.$$

A number of servers fulfill a stricter definition of service curves. They guarantee a higher output during periods of queued data, the so-called backlogged periods of a server.

Definition 3. Let $\beta \in \mathcal{F}_0$. Server S offers a strict service curve β iff, during any backlogged period of duration d, its output is at least equal to $\beta(d)$.

Basic operations to manipulate curves while conserving the model's deterministic worst case were cast in a $(\min, +)$-algebraic framework [11,15]:

Definition 4. The main (min,+)-algebraic DNC operations for $f, g \in \mathcal{F}_0$ are

$$\begin{aligned} \textit{aggregation} : (f+g)(t) &:= f(t) + g(t), \\ \textit{convolution} : (f \otimes g)(t) &:= \inf_{0 \le s \le t} \{f(t-s) + g(s)\}, \\ \textit{deconvolution} : (f \oslash g)(t) &:= \sup_{u \ge 0} \{f(t+u) - g(u)\}. \end{aligned}$$

With these operations, we can bound performance characteristics of flows:

Theorem 1. *Consider a server S offering a service curve β. Assume flow f with arrival curve α crosses S. We obtain these two performance bounds for f:*

$$\textit{Delay}: \quad \forall t \in \mathbb{R}^+ : D(t) \le \inf\{d \ge 0 \mid (\alpha \oslash \beta)(-d) \le 0\},$$

where it is assumed that the order of data in f is not altered.

$$\textit{Output}: \quad \forall d \in \mathbb{R}^+ : \alpha'(d) = (\alpha \oslash \beta)(d),$$

where α' is an arrival curve for A'.

Theorem 2. *Consider a single flow f crossing a tandem of servers $S_1, \ldots, S_n$ where each S_i offers a service curve β_{S_i}. The overall service curve for f is the concatenation of servers, achieved by convolution*

$$\beta_{S_1} \otimes \cdots \otimes \beta_{S_n} = \bigotimes_{i=1}^{n} \beta_{S_i}.$$

Theorem 3. *Consider a server S offering a strict service curve β_S. Let S be crossed by two flows f_0 and f_1 with arrival curves α^{f_0} and α^{f_1}, respectively. Then f_1's worst-case residual resource share without knowledge about multiplexing (so-called* arbitrary multiplexing*) at S, i.e., its left-over service curve at S, is*

$$\beta_S^{l.o.f_1} = \beta_S \ominus \alpha^{f_0},$$

where $(\beta \ominus \alpha)(d) := \sup_{0 \le u \le d} \{(\beta - \alpha)(u)\}$ denotes the non-decreasing upper closure of $(\beta - \alpha)(d)$.

The above left-over service curve operation is applicable to single systems only. Multiple DNC left-over service curve computations for tandems $\langle S_1, \ldots, S_n \rangle$ have been proposed in the literature. We include a common notation for these in Table 1 and discuss them as required in the next section.

Lastly, note that the optimization analyses LP and ULP [9] do not derive a left-over service curve. Instead, they each derive optimization formulations, linear programs, whose result bounds the foi's end-to-end delay.

Table 1. Deterministic network calculus notation.

Notation	Definition
foi	Flow of interest
$\{f_n, ..., f_m\}$	Flow aggregate containing flows $f_n, ..., f_m$
$\langle S_x, \ldots, S_y \rangle$	Tandem of consecutive servers S_x to S_y
α^f, $\alpha^{\{f_n,...,f_m\}}$	Arrival curve (flow, aggregate)
α_S^f, $\alpha_S^{\{f_n,...,f_m\}}$	Arrival bound at server S (flow, aggregate)
β_S	Service curve of server S
$\beta^{\text{l.o.}f}$, $\beta^{\text{l.o.}\{f_n,...,f_m\}}$	Left-over service curve (flow, aggregate)
$\beta_S^{\text{l.o.}f}$, $\beta_S^{\text{l.o.}\{f_n,...,f_m\}}$	Left-over service curve of server S
$\beta^{\text{l.o.}f}_{\langle S_x,...,S_y \rangle}$, $\beta^{\text{l.o.}\{f_n,...,f_m\}}_{\langle S_x,...,S_y \rangle}$	Left-over service curve of tandem S_x to S_y

3 Aggregation as the Objective of DNC Analyses

Basic DNC (min,+)-operations have been composed to analyses that achieve varying degrees of accuracy as they implement different sets of principles. Yet, not all principles can be fully attained at the same time. Some are even mutually exclusive under current DNC analyses. The most impactful principles and thus the best choice of analysis depends on the network scenario and the flow of interest to be bounded. Moreover, algebraic DNC analysis is compositional. It requires to combine tandem analyses to a network analysis. This paper focuses on the search for the best composition. In this section, we give detailed background on DNC weaknesses and previous attempts to mitigate or solve them by different analysis principles.

3.1 DNC Analyses and Principles

The ultimate goal of a DNC analysis is to give an upper bound on the delay for the foi. This flow is thus the starting point of an analysis, its path defines the first tandem of servers whose left-over service curve has to be derived. This computation requires bounds on data arrivals of interfering flows. Therefore, these cross flows are backtracked, their respective left-over service curve is derived, and the output from their path – another tandem of servers – is bounded. This procedure repeats recursively in order to consider all flows that impact the foi directly or indirectly [4]. Thus, a compositional network analysis is composed of many tandem analyses, each computing a left-over service curve $\beta^{\text{l.o.}}_{\langle S_1,...,S_n \rangle}$. To allow for selecting the best analysis per tandem, we present common principles.

Aggregation of Flows (Agg) [12,13]**.** Aggregation of flows crossing a server (Definition 4) is the earliest principle of DNC. Total Flow Analysis (TFA), the first DNC analysis, proposes to aggregate the totality of flows at each server. This is generally beneficial for output bounding, yet, an aggregate's delay bound

depends on the multiplexing discipline. In FIFO multiplexing, the horizontal deviation between aggregate arrival curve and service curve gives a valid bound (Theorem 1). For arbitrary multiplexing as considered in this paper, the intersection does. Separating the foi with a left-over service operation (Theorem 3) allows for horizontal deviation and results in better delay bounds.

Pay Bursts Only Once (PBOO) [15]**.** Computing the foi's left-over service curve enables a key principle of algebraic DNC analysis: convolving all left-over service curves of servers crossed by the foi (Theorem 2). Then, the tandem analysis does not compute the foi's arrivals at every server, but its burst term is only considered once and the principle is known as pay bursts only once. Separation of the foi and convolution of left-over service curves are key to the Separate Flow Analysis (SFA). Thus, the SFA implements the PBOO principle. Its delay bounds invariably outperform TFA delay bounds.

Pay Multiplexing Only Once (PMOO) [17]**.** In case cross flows share multiple consecutive servers with the foi, their burst terms appear in each server's left-over computation. I.e., multiplexing with cross-traffic bursts is paid for multiple times and the principle to counteract this issue is known as pay multiplexing only once. The eponymic analysis, PMOO Analysis (PMOOA), suggests to reverse SFA's order of operations. Servers are convolved before cross traffic is subtracted. [17] provides a tandem left-over service curve computation achieving PMOO. Note, that PMOO implies PBOO due to foi separation and convolution.

Order of Servers (Order) [16]**.** (min,+)-convolution is a commutative operation. Thus, the order of crossed servers is lost when applying it. This impacts the PMOOA as the slowest server defines the tandem service. For that reason, SFA's per-hop cross-traffic considerations can arbitrarily outperform PMOOA [16]. Therefore, in [16] the first optimization-based analysis (OBA) is proposed that derives a tandem left-over service curve implementing PBOO, PMOO and accounting for the order of servers.

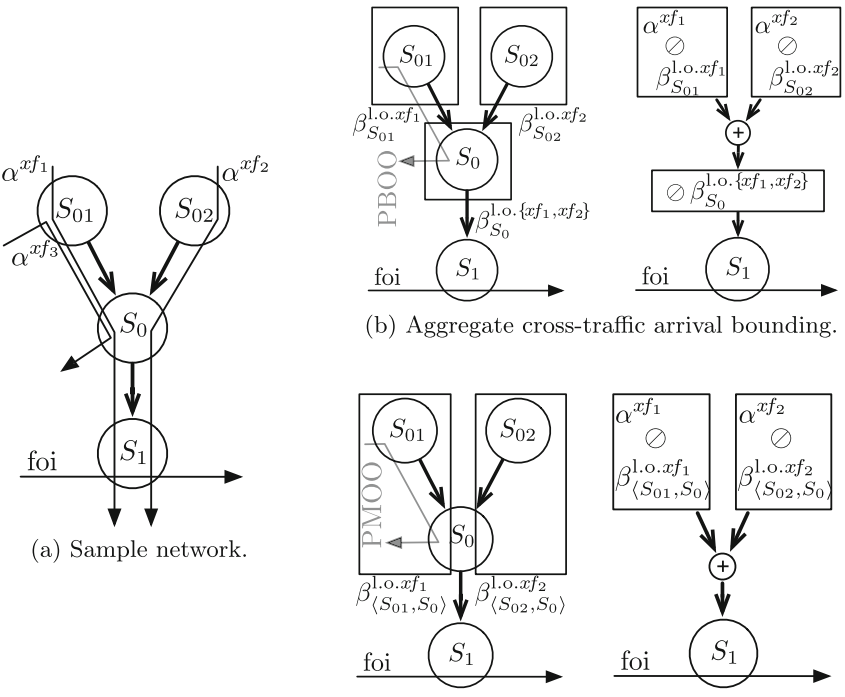
Output Burstiness Cap (OBC) [6]**.** While separation of flows is beneficial for delay bounding, it was shown in [6] that the output bound computation suffers. A subset of flows' output burstiness after a server can supersede the maximum amount of backlogged data by the totality of flows – a cap for a server's output burstiness that also holds for any subset of flows crossing the respective server.

Pay Segregation Only Once (PSOO) [7]**.** Applying the left-over service computation implicitly assumes higher priority for the subtracted flow. If, at a single server, two flows applying the left-over thus assume incompatible priorities, segregation is paid for more than once. Counteracting this issue results in better performance bounds.

Flow Prolongation (FP) [1]**.** A strategy to benefit from aggregation, similar to OBC, is to prolong flows over servers they do not cross in reality. The analysis has to work with increased load assumptions at these servers, however, if prolonged flows then share a path with other flows, aggregation is possible and its benefits can supersede the pessimism added to the model. FP was shown to be inherently infeasible [1] and is thus not included as a viable principle in Table 2.

3.2 Compositional Approaches for Arrival Bounding

As stated above, DNC composes tandem left-over service computations to a feed-forward network analysis. The literature proposes two alternatives. Both start with the foi but differ beyond its path, in the so-called cross-traffic arrival bounding. It analyzes the network between locations of interference with the foi and sources of relevant cross flows. It is usually the largest part of the analysis, as exemplified in Fig. 1.



(a) Sample network.

(b) Aggregate cross-traffic arrival bounding.

(c) Segregated cross-flow arrival bounding.

Fig. 1. Decomposition of a network (a) for cross-traffic arrival bounding: (b) depicts the alternative to aggregate cross-flows [8] that restricts to PBOO, (c) shows the segregation approach that can benefit from the PMOO principle.

Segregated Arrival Bounding (SegrAB) [8]**.** A straightforward extension of the SFA was proposed for cross-traffic arrival bounding [8]. Each flow interfering with the foi is analyzed using SFA's approach: compute per-server left-over service curves from source to the server before meeting the foi, convolve these service curves, compute an output bound. Then, all flows' arrival bounds are summed up. However, it was shown that, for $\alpha \in \mathcal{F}_{\mathrm{TB}}$ and $\beta \in \mathcal{F}_{\mathrm{RL}}$, the PSOO violations explicitly enforced by this approach cannot be set off by the implemented PBOO principle [5]. In Fig. 1c, the PSOO violation takes place at S_0. Due to the SFA-based extension, only PBOO is implemented in the two left-over service curves required to bound arrivals of cross flows, $\beta^{\mathrm{l.o.}xf_1}_{\langle S_{01},S_0\rangle}$ and $\beta^{\mathrm{l.o.}xf_2}_{\langle S_{02},S_0\rangle}$.

Aggregate Arrival Bounding (AggrAB) [5]**.** The approach immediately resulting from the insights on segregated PBOO arrival bounding is to strongly prefer aggregation. To maximize aggregation benefits, the length of tandems is reduced such that all analyzed flows take the same path over it and can thus be considered forming a single flow aggregate. Figure 1b depicts this approach: Instead of a PSOO violation, a single left-over service curve for xf_1 and xf_2 suffices at S_0. To achieve this, the flows' arrivals to S_0 need to be computed, PBOO is enforced. In total, the arrival bounding will thus consist of three left-over operations on shorter tandems instead of two on longer tandems.

3.3 Network Analyses

The above compositions of tandem analyses mostly take static, tandem-local information such as flow entanglement into account. In contrast, network analyses take a more network-wide view and break with the strict composition rules of SegrAB and AggrAB.

Tandem Matching Analysis (TMA) [2]**.** We abbreviate the algebraic search for best bounds presented in [2] as Tandem Matching Analysis (TMA). From a conceptual point of view, it matches differently sized tandems onto the entire feed-forward network to define an order of tandems to be analyzed. This is done in an exhaustive fashion, yet, based on PBOO-applying segregation's inferiority, paths of flow *aggregates* are a restricting factor. Thus, AggrAB becomes one of the alternatives TMA considers – in Fig. 1, no (sub)tandem has length >1 and thus TMA behaves exactly like AggrAB. TMA also leverages information about the order of subtandems and employs the output burstiness cap. This added flexibility in the tandem decomposition proved key for the most accurate algebraically derived DNC delay bounds to date. TMA also mitigates the combinatorial explosion of effort and shows good scaling of the analysis.

Linear Program/Unique Linear Program (LP/ULP) [9]**.** Instead of searching for the best combination of algebraic operations and analyses to apply, LP and ULP analysis directly search for the best delay bound. I.e., these

optimization-based analyses do not derive a left-over service curve (unlike OBA). They convert the entire network into an optimization formulation, a (set of) linear program(s), that relates backlogged periods of servers. Yet, the ultimately tight LP analysis suffers from unmitigated combinatorial explosion. In evaluations, the proposed heuristic ULP has been shown to only contribute little accuracy over TMA but at significantly longer analysis execution times [2].

4 Accuracy Improvements by SegrPMOO Cross-Traffic Arrival Bounding

We investigate the only not yet investigated principle in the TMA column in Table 2. In Fig. 1c, it enforces a segregation at S_1 but it allows for the PMOO principle for xf_1 and xf_2. In case these suffer from cross-traffic themselves, xf_3 interfering with xf_1 in our example, we can benefit from PMOO where aggregate arrival bounding (Fig. 1b) is only capable of implementing the PBOO principle. In this section, we show that the segregation/PMOO-tradeoff in SegrPMOO can outperform the aggregation/PBOO-tradeoff provided by current AggrAB.

4.1 Introducing SegrPMOO

We call the SegrAB strategy that exclusively applies the PMOO analysis for each left-over service curve derivation SegrPMOO. Next we give a proof that SegrPMOO can indeed outperform AggrAB.

Proposition 1. *Cross-flow segregation paired with a PMOO analysis is able to obtain lower bounds on flow arrivals than its aggregating counterpart. That is, none of these arrival bounding alternatives is a dominating approach.*

Table 2. Feature matrix of all current, mutually exclusive DNC analyses. Principle implementation: ✓ full, (✓) partial/selective, ✗ none, *NA* not applicable. [1]SFA requires arrival bounding for servers on the analyzed tandem.

Principle	Tandem Analysis				Network Analysis		
	TFA	SFA	PMOOA	OBA	TMA	ULP	LP
Agg	✓	(✓)	(✓)	(✓)	(✓)	✓	✓
PBOO	✗	✓	✓	✓	✓	✓	✓
PMOO	✗	✗	✓	✓	(✓)	✓	✓
Order	✗	✓	✗	✓	(✓)	✓	✓
OBC	✓	✗	*NA*		✓	*NA*	
PSOO	*NA*	✗[1]	*NA*		(✓)	(✓)[7]	✓
SegrAB	*NA*				✗	*NA*	
AggrAB	*NA*				✓	*NA*	
Good scaling	✓	✓	✓	✗[14]	✓	✗	✗

Proof. The superiority of AggrAB employing PBOO over the segregated version has been discussed in [5]. For the case that AggrAB implements either PBOO or PMOO and SegrAB implements PMOO, we give an example where cross-flow segregation yields a better result. Let us therefore consider the setting as in Fig. 1 with token-bucket arrivals ($\mathcal{F}_{\mathrm{TB}}$) and rate-latency service ($\mathcal{F}_{\mathrm{RL}}$). First, we derive the arrival bound when aggregating cross flows:

$$
\begin{aligned}
&\alpha_{S_1}^{\mathrm{Aggr}\{xf_1,xf_2\}} \\
&\quad = \alpha_{S_0}^{\{xf_1,xf_2\}} \oslash \beta_{S_0}^{\mathrm{l.o.}\{xf_1,xf_2\}} \\
&\quad = \left(\left(\alpha^{xf_1} \oslash \beta_{S_{01}}^{\mathrm{l.o.}xf_1}\right) + \left(\alpha^{xf_2} \oslash \beta_{S_{02}}^{\mathrm{l.o.}xf_2}\right)\right) \oslash \left(\beta_{S_0} \ominus \alpha_{S_0}^{xxf_1}\right) \\
&\quad = \left(\left(\alpha^{xf_1} \oslash \left(\beta_{S_{01}} \ominus \alpha^{xf_3}\right)\right) + \left(\alpha^{xf_2} \oslash \beta_{S_{02}}\right)\right) \oslash \left(\beta_{S_0} \ominus \left(\alpha^{xf_3} \oslash \beta_{S_{01}}^{\mathrm{l.o.}xf_3}\right)\right) \\
&\quad = \left(\left(\alpha^{xf_1} \oslash \left(\beta_{S_{01}} \ominus \alpha^{xf_3}\right)\right) + \left(\alpha^{xf_2} \oslash \beta_{S_{02}}\right)\right) \oslash \left(\beta_{S_0} \ominus \left(\alpha^{xf_3} \oslash \beta_{S_{01}}\right)\right) \\
&\quad = \left(\gamma_{r_1,b_1} \oslash \left(\beta_{R_{01},T_{01}} \ominus \gamma_{r_3,b_3}\right)\right) \\
&\qquad + \left(\left(\gamma_{r_2,b_2} \oslash \beta_{R_{02},T_{02}}\right) \oslash \left(\beta_{R_0,T_0} \ominus \left(\gamma_{r_3,b_3} \oslash \beta_{R_{01},T_{01}}\right)\right)\right).
\end{aligned}
$$

We continue with

$$
\begin{aligned}
&\alpha_{S_1}^{\mathrm{Aggr}\{xf_1,xf_2\}} \\
&\quad = \left(\left(\gamma_{r_1,b_1} \oslash \beta_{R_{01}-r_3,\frac{R_{01}\cdot T_{01}+b_3}{R_{01}-r_3}}\right) + \gamma_{r_2,b_2+r_2\cdot T_{02}}\right) \oslash \left(\beta_{R_0,T_0} \ominus \gamma_{r_3,b_3+r_3\cdot T_{01}}\right) \\
&\quad = \left(\gamma_{r_1,b_1+r_1\cdot\frac{R_{01}\cdot T_{01}+b_3}{R_{01}-r_3}} + \gamma_{r_2,b_2+r_2\cdot T_{02}}\right) \oslash \beta_{R_0-r_3,\frac{R_0\cdot T_0+b_3+r_3\cdot T_{01}}{R_0-r_3}} \\
&\quad = \gamma_{r_1+r_2,b_1+b_2+r_1\cdot\frac{R_{01}\cdot T_{01}+b_3}{R_{01}-r_3}+r_2\cdot T_{02}} \oslash \beta_{R_0-r_3,\frac{R_0\cdot T_0+b_3+r_3\cdot T_{01}}{R_0-r_3}} \\
&\quad = \gamma_{r_1+r_2,b_1+b_2+r_1\cdot\frac{R_{01}\cdot T_{01}+b_3}{R_{01}-r_3}+r_2\cdot T_{02}} + (r_1+r_2)\cdot\frac{R_0\cdot T_0+b_3+r_3\cdot T_{01}}{R_0-r_3}.
\end{aligned}
$$

At this point, please note that the PBOO property is preserved as b_1 and b_2 occur only once. The PMOO property, on the other hand, does not hold anymore, as b_3 is included twice. The segregated version yields

$$
\begin{aligned}
\alpha_{S_1}^{\mathrm{Segr}\{xf_1,xf_2\}} &= \alpha_{S_1}^{xf_1} + \alpha_{S_1}^{xf_2} \\
&= \left(\alpha^{xf_1} \oslash \beta_{\langle S_{01},S_0\rangle}^{\mathrm{l.o.}xf_1}\right) + \left(\alpha^{xf_2} \oslash \beta_{\langle S_{02},S_0\rangle}^{\mathrm{l.o.}xf_2}\right) \\
&= \left(\gamma_{r_1,b_1} \oslash \beta_{R_{\langle S_{01},S_0\rangle}^{\mathrm{l.o.}xf_1},T_{\langle S_{01},S_0\rangle}^{\mathrm{l.o.}xf_1}}\right) + \left(\gamma_{r_2,b_2} \oslash \beta_{R_{\langle S_{02},S_0\rangle}^{\mathrm{l.o.}xf_2},T_{\langle S_{02},S_0\rangle}^{\mathrm{l.o.}xf_2}}\right) \\
&= \gamma_{r_1,b_1+r_1\cdot T_{\langle S_{01},S_0\rangle}^{\mathrm{l.o.}xf_1}} + \gamma_{r_2,b_2+r_2\cdot T_{\langle S_{02},S_0\rangle}^{\mathrm{l.o.}xf_2}} \\
&= \gamma_{r_1+r_2,b_1+b_2+r_1\cdot T_{\langle S_{01},S_0\rangle}^{\mathrm{l.o.}xf_1}+r_2\cdot T_{\langle S_{02},S_0\rangle}^{\mathrm{l.o.}xf_2}}.
\end{aligned}
$$

Using

$$T^{\text{l.o.}xf_1}_{\langle S_{01},S_0\rangle} = T_{01} + T_0 + \frac{b_2 + b_3 + r_3 \cdot T_{01} + (r_2 + r_3) \cdot T_0}{(R_{01} - r_3) \wedge (R_0 - r_2 - r_3)},$$
$$T^{\text{l.o.}xf_2}_{\langle S_{02},S_0\rangle} = T_{02} + T_0 + \frac{b_1 + b_3 + (r_1 + r_3) \cdot T_0}{R_{02} \wedge (R_0 - r_1 - r_3)}$$

computed with [17] gives us

$$\begin{aligned}\alpha^{\text{Segr}\{xf_1,xf_2\}}_{S_1} &= \gamma_{r_1+r_2,b_1+b_2+r_1\cdot\left(T_{01}+T_0+\frac{b_2+b_3+r_3\cdot T_{01}+(r_2+r_3)\cdot T_0}{(R_{01}-r_3)\wedge(R_0-r_2-r_3)}\right)} \\ &\qquad +r_2\cdot\left(T_{02}+T_0+\frac{b_1+b_3+(r_1+r_3)\cdot T_0}{R_{02}\wedge(R_0-r_1-r_3)}\right) \\ &= \gamma_{r_1+r_2,b_1+b_2+r_1\cdot T_{01}+r_1\cdot T_0+r_1\cdot\frac{b_2+b_3+r_3\cdot T_{01}+(r_2+r_3)\cdot T_0}{(R_{01}-r_3)\wedge(R_0-r_2-r_3)}} \\ &\qquad +r_2\cdot T_{02}+r_2\cdot T_0+r_2\cdot\frac{b_1+b_3+(r_1+r_3)\cdot T_0}{R_{02}\wedge(R_0-r_1-r_3)}\end{aligned}$$

where the PMOO principle is implemented per flow xf_1 and xf_2. Yet, overall, b_3 appears twice. We bound all arrivals with equal token buckets and continue by comparing burst terms. As we are free to choose parameters, we set $T_0 = T_{01} = T_{02} = b_1 = b_2 = 0$ and the arrival rates to be homogeneous ($r_1 = r_2 = r_3 =: r > 0$). We further assume the burst term b_3 to be >0. Assume now that the claim does not hold true yielding for the burst term

$$\begin{aligned} & b^{\text{Aggr}\{xf_1,xf_2\}}_{S_1} < b^{\text{Segr}\{xf_1,xf_2\}}_{S_1} \qquad (1)\\ \Leftrightarrow \quad & r \cdot \frac{b_3}{R_{01} - r} + r \cdot \frac{b_3}{R_0 - r} + r \cdot \frac{b_3}{R_0 - r} \\ & \qquad < r \cdot \frac{b_3}{(R_{01} - r) \wedge (R_0 - 2r)} + r \cdot \frac{b_3}{R_{02} \wedge (R_0 - 2r)} \\ \Leftrightarrow \quad & \frac{1}{R_{01} - r} + \frac{2}{R_0 - r} < \frac{1}{(R_{01} - r) \wedge (R_0 - 2r)} + \frac{1}{R_{02} \wedge (R_0 - 2r)}. \end{aligned}$$

In order to contradict the claim and prove the proposition, it is sufficient to give an example where Eq. (1) cannot hold. For this, see Example 1 below.

Example 1. Choosing $r = 1$, $R_0 = 4$, and $R_{01} = R_{02} = 2$ in Eq. (1) results in $\frac{5}{3} \lightning \frac{3}{2}$.

5 Numerical Evaluation

Previous evaluations of the TMA [2] showed that its delay bounds are close to those computed with the ULP optimization. Yet, there is still a gap that can be significant for some outliers. In this section, we extend the previous evaluation by SegrPMOO to check if it can mitigate the cause for this gap and the outliers; either by being independently executed as a stand-alone arrival bounding or by deeply integrating it into to the search executed by TMA.

Analyzed Networks. [2] provides Internet-like topologies, generated according to the general linear preference GLP model [10] ($m_0 = 20$, $m = 1$, $p = 0.4695$, $\beta_{\text{GLP}} = 0.6447$). We extend the analysis of these homogeneous networks of sizes 20 and 40 devices (Table 3) with arrival curves set to $\gamma_{5\text{Mbps},5\text{Mb}} \in \mathcal{F}_{\text{TB}}$ and service curves set to $\beta_{10\text{Gbps},0} \in \mathcal{F}_{\text{RL}}$.

Accuracy Metric. We are interested in the gap that algebraic DNC analyses have to close to achieve the ULP optimization's accuracy. Our metric of choice is therefore

$$\text{Gap Closing}[\%] = \frac{|\text{Delay}_{\text{TMA}} - \text{Delay}_{\text{NewAnalysis}}|}{|\text{Delay}_{\text{TMA}} - \text{Delay}_{\text{ULP}}|}. \tag{2}$$

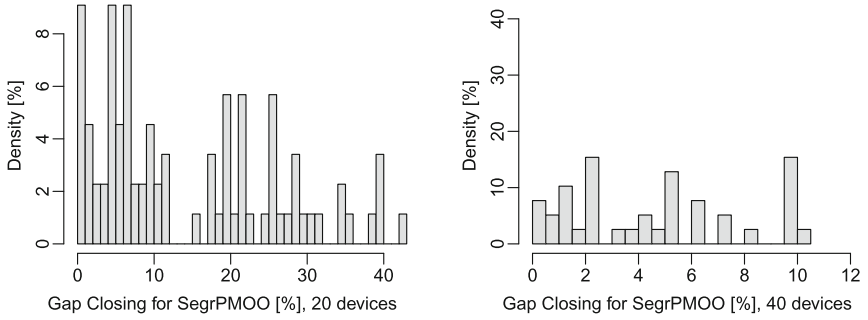
Execution Time Metric. To allow for meaningful extension and comparison of execution time measurements, computations were executed with the same tools (DiscoDNC [3] v2.2.3 and IBM CPLEX version 12.6.2) on the same hardware platform (2x Intel Xeon E5420 CPU, 12 GB main memory) as in [2]. We measure the time it takes to analyze all flows in a given network.

5.1 Accuracy

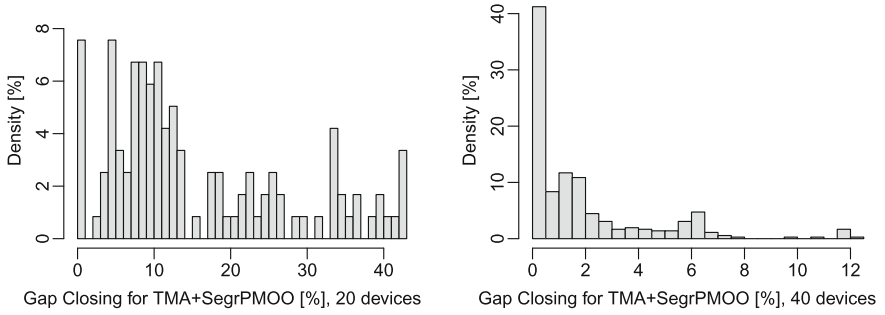
SegrPMOO Arrival Bounding vs. Optimization Analysis. We first evaluate SegrPMOO in isolation. That is, we analyze the foi with TMA and compute the required cross-traffic arrival bounds with SegrPMOO only. This strategy defines the first *NewAnalysis* in Eq. (2). Table 3a provides the results for networks of size 20 and 40 devices, translating to 152 and 472 flows to be analyzed respectively. While the share of improved delay bounds in the smaller network exceeds 50%, it already decreases to 10% in the larger network. Table 3 also gives the max and mean gap closing whereas Fig. 2a shows the gap closing distributions for the flows that showed improved delay bounds.

Table 3. Closing the gap between TMA and ULP.

Devices	Servers	Flows		Gap closing [%]	
		Total	Improved	Max	Mean [improved]
(a) TMA foi Analysis, only SegrPMOO Arrival Bounding					
0	38	152	88	42.36299	14.514300
40	118	472	39	10.03384	4.594554
(b) TMA foi Analysis, TMA+SegrPMOO Arrival Bounding					
20	38	152	119	42.92102	15.50147
40	118	472	342	12.00559	1.877968



(a) Gap closing density of SegrPMOO for 20 devices (left) and 40 devices (right).



(b) Gap closing density of TMA+SegrPMOO for 20 devices (left) and 40 devices (right).

Fig. 2. Closing the gap to ULP delay bounds.

Combined AggrAB and SegrPMOO vs. Optimization Analysis. Second, we integrate SegrPMOO into AggrAB-based TMA arrival bounding. At every recursion level of TMA, i.e., when flows split up, a new arrival bounding is started (see Fig. 1b, above server S_0). Here, we additionally execute SegrPMOO. For these SegrPMOO arrival boundings, the same holds vice versa. When they need to recursively bound arrivals of cross traffic, bounds are additionally derived with TMA. In both cases only the smaller of the derived bounds is considered.

Results are depicted in Table 3b: a rather steady share of all flows, 78.3% (20 devices) and 72.5% (40 devices), respectively, sees improvements. I.e., in these cases, applying at least one cross-flow segregation during the entire arrival bounding process was beneficial over TMA only. Also, the shares are considerably larger than with SegrPMOO only. This result reveals a rather large amount of situations leading to segrPMOO superiority, although it depends on specific flow entanglements and parameter combinations. Note, that the latter solely occurs due to curve transformations as we evaluate homogeneous networks. The improvements themselves, however, are in proximity of the SegrPMOO results and considerably less pronounced in the larger network. The maximum reduction of the gap to the ULP shrinks from 42.9% to 12% and the mean from 15.5% to 1.88%. Figure 2b shows the gap closing distribution. Compared to SegrPMOO,

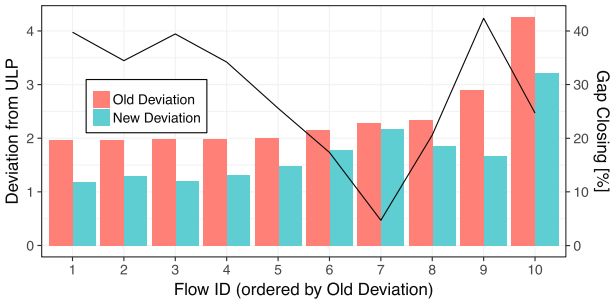


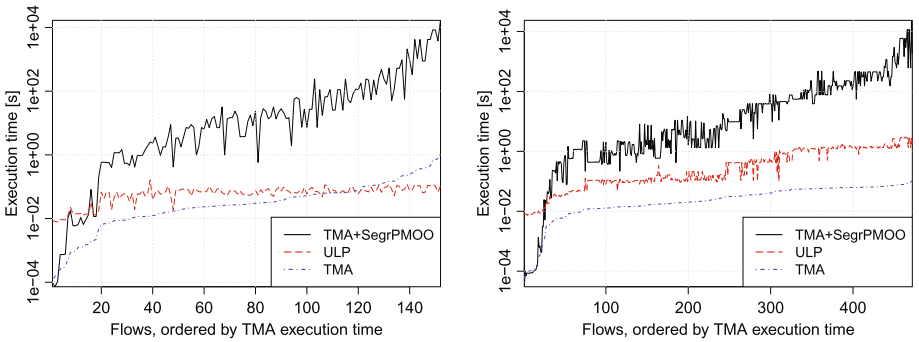
Fig. 3. Gap closing with TMA+SegrPMOO for outliers, 20 devices network.

mostly flows with small improvements are added. Yet, the 20 devices network sees a noticeable growth of the flows having their gap closed by >40%.

Catching Outliers. We also evaluated where TMA+SegrPMOO's impact is concentrated. Figure 3 depicts old and new deviation as well as gap closing for the 10 outlier flows that previously suffered from the largest gap to ULP. 8 out of 10 see their gap closed by more than 20% and the largest outlier even benefits from an improvement narrowing its gap from 4.26% to 3.21%. Our results show that the peculiar corner cases where segregation helps concentrate at the outliers.

5.2 Computational Effort

The enhanced delay bound computation of TMA+SegrPMOO comes at the price of additional computational cost, as more arrival bounds are computed at each recursion level of the TMA analysis. We measured the execution time of each full analysis, i.e., for all flows in each network. The results are shown in Fig. 4,



(a) 20 devices. (b) 40 devices.

Fig. 4. Execution time comparison.

ordered by the previously known TMA execution times. We observe that for cases with a very fast TMA computation, the added SegrPMOO overhead is negligible. In these cases, the recursion is not deep as little flows are involved and paths of relevant cross flows are rather short. Yet, the ULP analysis does not seem to be accelerated for these flows and their smaller optimization problem.

However, the amount of flows not requiring much arrival bounding is very small. Thus, we observe a sharp increase of computation times for the majority of flow analyses. TMA+SegrPMOO even takes considerably more time than the ULP analysis. Analyzing the 20 and 40 devices network only seems manageable due to their small sizes. In fact, we also observed that the 60 devices network presented in [2] becomes infeasible to fully analyze in acceptable time. This means design space exploration with TMA+SegrPMOO is out of scope although this network only consists of 164 servers and 656 flows. Nonetheless, if only a small number of flow delay bounds exceed their predefined deadlines, a selective, additional analysis of these flows comes at an acceptable execution overhead.

A TMA+SegrPMOO Heuristic. Last, let us remark the potential for a heuristic that trades accuracy for faster computation. The later SegrPMOO is applied in the recursive arrival bounding, the shorter the tandem it analyzes. Thus, it becomes less likely that AggrAB enforces PBOO where the analysis could benefit from PMOO. This is reflected in Table 3 where the improvement from SegrPMOO to TMA+SegrPMOO seems small but might still be decisive. We leave heuristics selectively removing SegrPMOO from TMA for future work.

6 Conclusion

In this paper, we demonstrate that cross-flow segregation combined with the PMOO principle can outperform the predominating objective to aggregate flows. We contribute an analysis that incorporates this SegrPMOO approach into the existing TMA. Our numerical evaluations show that this new analysis outperforms others for the majority of analyzed flows. However, the improvement's amplitude can be small and comes at a considerably increased analysis cost. Our new TMA+SegrPMOO is thus most suitable for small networks or a follow-up analysis for selected flows, for instance to ensure strict certification requirements.

References

1. Bondorf, S.: Better bounds by worse assumptions - improving network calculus accuracy by adding pessimism to the network model. In: Proceedings of IEEE ICC (2017)
2. Bondorf, S., Nikolaus, P., Schmitt, J.B.: Quality and cost of deterministic network calculus - design and evaluation of an accurate and fast analysis. ACM POMACS **1**(1), 16:1–16:34 (2017)
3. Bondorf, S., Schmitt, J.B.: The DiscoDNC v2 - a comprehensive tool for deterministic network calculus. In: Proceedings of EAI ValueTools (2014)

4. Bondorf, S., Schmitt, J.B.: Boosting sensor network calculus by thoroughly bounding cross-traffic. In: Proceedings of IEEE INFOCOM (2015)
5. Bondorf, S., Schmitt, J.B.: Calculating accurate end-to-end delay bounds - You better know your cross-traffic. In: Proceedings of EAI ValueTools (2015)
6. Bondorf, S., Schmitt, J.B.: Improving cross-traffic bounds in feed-forward networks – there is a job for everyone. In: Remke, A., Haverkort, B.R. (eds.) MMB&DFT 2016. LNCS, vol. 9629, pp. 9–24. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31559-1_3
7. Bondorf, S., Schmitt, J.B.: Should network calculus relocate? An assessment of current algebraic and optimization-based analyses. In: Agha, G., Van Houdt, B. (eds.) QEST 2016. LNCS, vol. 9826, pp. 207–223. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-43425-4_15
8. Bouillard, A.: Algorithms and efficiency of network calculus. Habilitation thesis, École Normale Supérieure (2014)
9. Bouillard, A., Jouhet, L., Thierry, E.: Tight performance bounds in the worst-case analysis of feed-forward networks. In: Proceedings of IEEE INFOCOM (2010)
10. Bu, T., Towsley, D.: On distinguishing between internet power law topology generators. In: Proceedings of IEEE INFOCOM (2002)
11. Chang, C.-S.: Performance Guarantees in Communication Networks. Springer, London (2000). https://doi.org/10.1007/978-1-4471-0459-9
12. Cruz, R.L.: A calculus for network delay, Part I: network elements in isolation. IEEE Trans. Inf. Theory **37**(1), 114–131 (1991)
13. Cruz, R.L.: A calculus for network delay, Part II: network analysis. IEEE Trans. Inf. Theory **37**(1), 132–141 (1991)
14. Kiefer, A., Gollan, N., Schmitt, J.B.: Searching for Tight Performance Bounds in Feed-Forward Networks. In: Müller-Clostermann, B., Echtle, K., Rathgeb, E.P. (eds.) MMB&DFT 2010. LNCS, vol. 5987, pp. 227–241. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12104-3_18
15. Le Boudec, J.-Y., Thiran, P. (eds.): Network Calculus: A Theory of Deterministic Queuing Systems for the Internet. LNCS, vol. 2050. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45318-0
16. Schmitt, J.B., Zdarsky, F.A., Fidler, M.: Delay bounds under arbitrary multiplexing: when network calculus leaves you in the lurch ... In: Proceedings of IEEE INFOCOM (2008)
17. Schmitt, J.B., Zdarsky, F.A., Martinovic, I.: Improving performance bounds in feed-forward networks by paying multiplexing only once. In: Proceedings of GI/ITG MMB (2008)

QoE Analysis of the Setup of Different Internet Services for FIFO Server Systems

Tobias Hoßfeld[1(✉)], Martín Varela[2], Poul E. Heegaard[3], and Lea Skorin-Kapov[4]

[1] Modeling of Adaptive Systems, University of Duisburg-Essen, Essen, Germany
tobias.hossfeld@uni-due.de
[2] Independent Researcher, Oulu, Finland
martin@varela.fi
[3] Department of Information Security and Communication Technology, NTNU, Norwegian University of Science and Technology, Trondheim, Norway
poul.heegaard@item.ntnu.no
[4] Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia
Lea.Skorin-Kapov@fer.hr

Abstract. Queueing systems following a first-in-first-out (FIFO) strategy are well understood and various results are known for the response time of the system. However, the question arises how the results look like when taking into account a user-centric point of view. To this end, an M/M/1-FIFO system is investigated for the setup of different Internet services. In this tutorial paper, the impact of the system's delay on Quality of Experience (QoE) is considered for (1) YouTube video (initial playout delay), (2) authentication in social networks, (3) wireless 3G Internet connection setup. Existing QoE models are used to map the response time in the system, corresponding to the waiting time for users until the service is setup, to Mean Opinion Scores (MOS) as a measure of QoE. The system is then evaluated in terms of overall QoE and QoE fairness for the three services considered, under different load scenarios. The results show how different such systems and response times are perceived by users of different services. Further, the dimensioning of FIFO systems with respect to QoE only requires us to consider the overall QoE.

1 Introduction

Quality of Experience is the "degree of delight or annoyance of the user of an application or service" [2]. Furthermore, "it results from the fulfillment of his or her expectations with respect to the utility and/or enjoyment of the application or service in the light of the user's personality and current state". In real services, the quality perceived by the user is heavily affected by the performance of the underlying system, and in particular, of the network. In general, the factors influencing QoE can be classified into Human, System and Context influence factors (IFs) [2]. In practical applications, the Human IFs are hard (if even

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 234–248, 2018.
https://doi.org/10.1007/978-3-319-74947-1_16

possible) to measure and affect, and often times, the Context factors are similarly intractable (though some Context-related factors can be measured and taken into account e.g., in QoE models).

System IFs, in contrast, are both better understood and somewhat possible to control. We can further refine System IFs into Application and Resource IFs, as described in [17]. Application factors can relate to e.g., choice of encoding, use of error concealment/correction mechanisms. Resource factors relate to, for example, device capabilities, network resources and state, etc.

In this tutorial paper we focus on the analysis of systems from a QoE perspective, and in particular, on the network performance. We illustrate the approach through the use of simple FIFO queues. This type of approach has been used with good results for instance for analysing the effects of Forward Error Correction (FEC) on VoIP streams [1,16]. Of course, there are more realistic and complex models, but the core message of the paper can be best explained with an analytically simple queuing system, without loss of generality.

Based on existing studies [6], QoE models for the following Internet services are utilized by mapping response times to Mean Opinion Score (MOS) values, (1) YouTube until the video playout starts, (2) authentication in social networks, (3) wireless 3G Internet connection setup. Once again, the use of the MOS provides the simplest possible way to perform a QoE analysis, despite it being sub-optimal for e.g., control or business purposes [7].

The remainder of the paper is structured as follows. Section 2 revisits existing results on the performance of the M/M/1-FIFO queue. In particular, the response time distribution is available. Section 3 provides a background on existing QoE models for waiting times and introduces the mapping functions used in the study of the FIFO system. In addition, the QoE fairness metric and its computation is discussed. Section 4 shows the methodology to analytically and numerically derive the QoE results. The numerical results for the different services are analyzed in Sect. 5. Section 6 concludes this work with an outlook on future work.

2 Performance of the M/M/1-FIFO Queue

The setup of the different Internet services is modeled as M/M/1-FIFO queue. The different user requests arrive in the system and are served in a first-in-first-out manner, i.e. in the order of user arrival. Since there is only one server, users may have to wait until they are served. The total response time of the system (also called sojourn time) includes the waiting time and the processing time of a user request (also called a job in queueing theory). The M/M/1 system is well investigated, see [10] or e.g., [5,11] for more recent textbooks, but the main results are briefly revisited to give the reader a tutorial-like overview on the QoE analysis of M/M/1-FIFO systems. Please note that the full Kendall notation of the system is M/M/1/∞/FIFO, as the waiting room is not limited.

2.1 Concepts and Notation

The concepts and notation frequently used in the paper are summarized below in Table 1. Random variables (RV) are typically denoted by upper case letters. We first describe general concepts, before the system and QoE relevant parameters are introduced.

Table 1. Notation and variables

	Notation	Meaning
General	$\mathrm{E}[X]$	Expected value of a random variable X with probability density function $x(t)$, $\mathrm{E}[X] = \int_{-\infty}^{\infty} tx(t)dt$
	$\mathrm{Var}[X]$	Variance of a random variable X, $\mathrm{Var}[X] = E[X^2] - E[X]^2$
	$\mathrm{Std}[X]$	Standard deviation of a random variable X, $\mathrm{Std}[X] = \sqrt{\mathrm{Var}[X]}$
	$F_X(t)$	Cumulative distribution function (CDF) of the RV X, $F_X(t) = P(X \leq t)$
	$f_X(t)$	Probability density function (PDF) of the RV X, $f_X(t) = \frac{d}{dt}F_x(t)$
$M/M/1 - FIFO$	λ	Arrival rate of user requests in the system (1/s)
	μ	Service rate of user requests (1/s) with mean service time $E[X] = 1/\mu$
	X	Service time (RV) of requests (s)
	ρ	Load in the system corresponding to the system utilization $\rho = \lambda/\mu$
	W	Waiting time (RV) of a user in the system (s)
	R	Response time (RV) of the system (s)
QoE Mapping	$f(t)$	Generic mapping function $f(t) = -a\log_{10}(t+b) + H$ between waiting times and QoE, see Eq. (9), with service-dependent parameters a, b and upper QoE bound H
	Q_m	Maximum possible QoE, i.e. $Q_m = \min(H, f(0))$
	t_m	Largest waiting time for which QoE still reaches its maximal value, i.e. $Q_m = f(t_m) = f(t)$ for $t \leq t_m$
	t_L	Minimum QoE is L for any $t \geq t_L$
	$f_C(t)$	Initial delays for YouTube obtained via crowdsourcing, see Eq. (10)
	$f_L(t)$	Initial delays for YouTube tested in a laboratory setting, see Eq. (11)
	$f_S(t)$	Authentication in social networks, see Eq. (12)
	$f_3(t)$	Wireless 3G Internet connection setup, see Eq. (13)
QoE Values	Y	QoE values (RV) obtained by mapping response times to QoE, $Y = f(R)$, thus Y is a continuous random variable
	L	Lower bound of the QoE domain, i.e. $L \leq Y$
	H	Upper bound of the QoE domain, i.e. $L \leq Y \leq H$, e.g. $L = 1$ and $H = 5$ for a 5-point scale
	$\mathrm{E}[Y]$	Overall QoE reflecting the expected QoE Y
	$\mathcal{F}$	QoE fairness of QoE values Y

2.2 First Moments of the Response Time

In an $M/M/1$ system, let the expected service time be denoted $E[X] = 1/\mu$, where X is the random variable for the service time. The expected sojourn time in the system is then

$$\mathrm{E}(R) = \frac{1}{1-\rho}\frac{1}{\mu} = \frac{1}{\mu-\lambda},\ \lambda < \mu\ . \tag{1}$$

The expected queueing, or waiting, time $E[W]$ is then the expected sojourn time in the system minus the expected service time.

$$\mathrm{E}(W) = \mathrm{E}(R) - \mathrm{E}(X) = \frac{\lambda}{\mu(\mu-\lambda)},\ \lambda < \mu\ . \tag{2}$$

The expected waiting time given that the customer has to wait (delayed customer) $\mathrm{E}(W \mid W > 0)$ is found using the law of total expectation.

$$\mathrm{E}(W \mid W > 0) = \frac{1}{\mu-\lambda} = \mathrm{E}(R),\ \lambda < \mu\ . \tag{3}$$

2.3 Response Time Distribution

In contrast to the expected times in the system, the time distributions depend on the queueing discipline. In this paper, we consider the simplest case of FIFO queueing for three cases of waiting (queuing) time distribution, (i) waiting time for a "tagged" customer who sees q customers ahead on arrival, (ii) waiting time for customers who have to wait, and (iii) response time for all customers.

Specific (tagged) Delayed Customer. First, we consider the conditional waiting time distribution for customers who are delayed due to queuing. Assume that the system is in state $i = q + 1$, where $q \in N$ is the number of customers in queue immediately before a customer enters the system. This customer has to wait $q + 1$ service completions before being served. When the server is busy, the system completes customers with a constant rate μ (negatively exponentially distributed service times). Therefore, the waiting time for a customer that has to wait (delayed customer) given that there are q customers ahead in the queue is Erlang-$(q + 1)$ distributed. The cumulative distribution function (CDF) is then

$$F_{W|W>0}(t \mid q) = \sum_{j=q+1}^{\infty} \frac{(\mu t)^j}{j!} e^{-\mu t},\ q \in N,\ t > 0\ . \tag{4}$$

Conditional for Customers Who Have to Wait. The unconditional waiting time distribution for delayed customers is derived using the *law of total probability.*

$$F_{W|W>0}(t) = \sum_{q=0}^{\infty} F_{W|W>0}(t \mid q) p_{W>0_q} \tag{5}$$

where $p_{W>0_q}$ is the probability that there are q customers in the queue, i.e., $q+1$ customers in the system, immediately before a new customer enters the system given that the server is busy, i.e., given that this new customer has to wait.

Hence, in an $M/M/1/\infty/FIFO$ system the waiting time distribution for delayed customers is negatively exponentially distributed with parameter $(1-\rho)\mu = (\mu - \lambda),\ \lambda < \mu$, thus

$$F_{W|W>0}(t) = 1 - e^{-(\mu-\lambda)t}\,. \tag{6}$$

This implies that if a customer counts the number q of customers in the queue when entering the system the waiting time is Erlang-$(q+1)$ distributed with parameter μ, while if not the waiting time is negatively exponentially distributed with parameter $(\mu - \lambda)$.

Response Time Distribution. Following a similar approach as for the waiting time distribution, if an $M/M/1$ system with FIFO queueing is in state i immediately before a customer enters the system, the response time for this customer is Erlang-$(i+1)$ distributed, and the response time, or sojourn time, for an arbitrary customer is negatively exponentially distributed with parameter $(\mu - \lambda),\ \lambda < \mu$.

Due to the PASTA[1] property [20] for a stationary system with Poisson arrivals, the probability that an arrival finds the system in state q is equal to the probability that an outside observer finds the system in state q at an arbitrary point of time. The arrival- and the time-stationary distributions are identical.

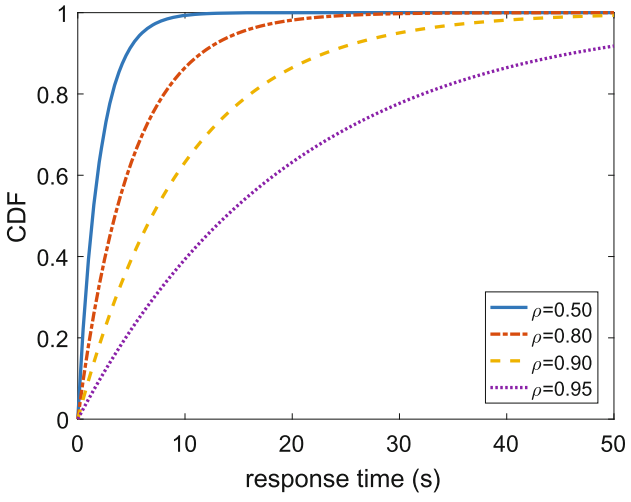


Fig. 1. Cumulative distribution function (CDF) of the response times of an M/M/1-FIFO queue with different load ρ. The response time (also referred to as sojourn time) is the waiting time in the FIFO queue and the processing time of a job at the server.

[1] "Poisson Arrivals See Time Averages".

The response time distribution in an $M/M/1/\infty/FIFO$ system is negatively exponentially distributed with parameter $(\mu - \lambda)$, $\lambda < \mu$. For the sake of simplicity, the CDF of the response time is also written as $R(t)$ in this paper.

$$R(t) = F_R(t) = 1 - e^{-(\mu-\lambda)t} \tag{7}$$

Figure 1 provides the CDF of the response time depending on the utilization ρ. With an increasing system load, the response times are increasing, but also the variance of the response time is increasing. Next, we will use the response time and map it to QoE values to investigate the system in a user-centric way.

3 QoE Models and QoE Fairness

In order to map the system response times to the user-perceived quality, we use existing QoE mapping functions. These are based on subjective studies [6]. The QoE mapping allows to objectively estimate QoE values in the $M/M/1$ system. The users are not differentiated, and therefore a response time r is mapped to a QoE value y for any given user. This *objective* view on the user-perceived quality allows a system provider to, e.g., do QoE management in a meaningful way. In the literature, there are several measures to quantify QoE [7]. The most commonly used QoE measure is the Mean Opinion Score (MOS), which represents the quality experienced by a hypothetical "average user". For a service provider allocating resources to users, it may however be more important to consider other measures of quality, such as the 10% most annoyed users, which may be expressed by the 10%-quantile. Service providers may also want to consider the percentages of users judging a service as "poor or worse" (%PoW) or "good or better" (%GoB). Those users who are experiencing lower quality than the MOS (mean) would suggest, are the ones who might be more susceptible to churn, or open help-desk tickets, etc., all of which has direct business consequences.

In this paper, we use MOS as our QoE measure, simply because there are good mappings available between response times and it in the literature [6], but in actual usage by, say, a service provider, other measures might be better suited.

For the user-centric analysis of the M/M/1 system, the response times are mapped to QoE values $Y = f(R)$ and the overall QoE as well as QoE fairness are investigated. The definition of both notions is introduced in Sect. 3.2.

3.1 Existing Mapping Functions Between Waiting Times and QoE

The QoE of Internet applications and services is often shaped by waiting times before — or during — service consumption. Those waiting times may be a result of insufficient resources (e.g., limited transmission capacity, limited cloud computing resources), network impairments (e.g., packet loss or high latency), or simply time-consuming operations. In [6], subjective user studies were conducted to analyze the differences in the user perception of initial delays for different interactive services. In the studies, the users evaluated the QoE on a so-called 5-point

absolute category rating (ACR) scale with the following meaning: 5 - excellent, 4 - good, 3 - fair, 2 - poor, 1 - bad quality. Then for each waiting time and Internet service under test, the average rating score was computed reflecting the MOS value for that test condition. Based on those subjective results, the relationship between QoE and the waiting times were derived. In [3,4], a hypothesis was formulated that the relationship between waiting time and its QoE evaluation on a linear ACR scale is logarithmic, motivated by the logarithmic form of the well-known Weber-Fechner law [19]. In [6], the mapping function is formulated as follows:

$$f^*(t) = -a \log_{10}(t + b) + H, \tag{8}$$

where the constant $H = 5$ reflects the upper bound of the ACR scale and a and b are service specific parameters obtained from subjective studies.

Please note that the mapping function maps a continuous response time $t \in \mathbb{R}^+$ to a continous MOS score $f(t) \in \mathbb{R}^+$. Thus, a response time distribution R can be mapped to a continous QoE distribution Y. Since Eq. (8) does not respect the limits of the rating scale (higher bound $H = 5$ and lower bound $L = 1$), the mapping functions needs to be refined by considering the related bounds $t_m = \max(0, 1 - b)$ and $t_L = 10^{(H-L)/a} - b$.

$$f(t) = \begin{cases} L & \text{for } t \geq t_L = 10^{(H-L)/a} - b \\ -a \log_{10}(t + b) + H, & \text{for } t_L \leq t \leq t_m \\ Q_m, & \text{for } t \leq t_m = \max(0, 1 - b) \end{cases} \tag{9}$$

The maximum QoE being observed is $Q_m = f^*(t_m) = -a \log_{10}(t_m + b) + H$, while the minimum QoE is L for any $t \geq t_L$. In the following, we simplify the notation and only provide the logarithmic function and the bounds.

Initial Delays in YouTube Video Streaming. In general, HTTP streaming utilizes a video buffer to both decrease the impact of network jitter and to decrease the probability of interruptions during the video playout. For YouTube video streaming, a certain buffer level is to be reached [18] before the video starts playing. For the evaluation of the encountered initial delays, a laboratory study as well as a crowdsourcing study were conducted. The two different test methodologies are not of importance for this paper, but the small deviations in the mapping functions are of interest if they are relevant for different load scenarios in the $M/M/1$ system. As a result, the following mapping functions were found for the crowdsourcing and the laboratory setting, respectively [6].

$$f_C(t) = -0.963 \log_{10}(t + 5.381) + 5,\ Q_m = 4.2962,\ t_m = 0, t_L = 14240.40 \tag{10}$$

$$f_L(t) = -0.862 \log_{10}(t + 6.718) + 5,\ Q_m = 4.2869,\ t_m = 0, t_L = 43682.19 \tag{11}$$

Authentication in Social Networks. The second Internet service addresses the user authentication in social networks. In [12,13], users evaluated the perceived quality of web-based login operations using a laptop. In the subjective

experiments, a remote OpenID server was run for authenticating the users as backend of the web page of the social network. The waiting times of the users were changed with a traffic shaper. To be more precise, the shaper induced predetermined response times for the authentication procedure when the user logged in. After the delayed login, the users were asked to rate how they experienced the login with regards to the response time resulting in the following mapping function [6].

$$f_S(t) = -2.816 \log_{10}(t + 1.378) + 5,\ Q_m = 4.6079,\ t_m = 0, t_L = 24.95 \quad (12)$$

Wireless 3G Internet Connection Setup. The next use case considers the perception of waiting times for wireless 3G Internet connection setup. In [15], subjective experiments were conducted where test users were sitting in front of a laptop for the 3G connection. For simulating different waiting times, a network emulator was customized in such a way, that the time span from pressing a "Connect" button to successful connection establishment was delayed for a defined time period. After the successful connection setup, the users evaluated again the QoE on a 5-point ACR scale how satisfied they were with the performance of the connection setup. The corresponding QoE mapping function is as follows.

$$f_3(t) = -1.577 \log_{10}(t + 0.742) + 5,\ Q_m = 5,\ t_m = 0.2580, t_L = 343.18 \quad (13)$$

Figure 2 illustrates the different QoE mapping functions for the different Internet services. It can be seen that the waiting time perception across different services strongly diverges. As a concrete example, let us consider an initial delay of 10s. In case of YouTube, this leads to good QoE (3.86 crowdsourcing, 3.95 laboratory), whereas for the 3G setup users perceive this somewhere between fair and good (3.37). In case of the social network authentication, the quality is perceived as bad (2.03). Hence, considerable differences are observed for the same waiting times across services. Please note that these are all very simple examples of QoE mappings, as they are all univariate, and consider only one single aspect of quality. The same type of approach can be used to build more complex QoE models, taking more quality-influencing factors into account.

3.2 Definition of Overall QoE and QoE Fairness

Overall QoE. For a user-centric analysis of the system, the overall QoE and the QoE fairness of the system are investigated. We can define the overall QoE as the expected QoE for an arbitrary user in the system. Please note that due to the nonlinear mapping function, the overall QoE does not follow by mapping the mean response time to QoE. In the paper, the overall QoE will be derived numerically from the CDF of QoE values. Due to Jensen's inequality [9] it is known that

$$\mathrm{E}[Y] = \mathrm{E}[f(R)] \geq f(\mathrm{E}[R]) \quad (14)$$

for a convex function f which is the case for the considered mapping functions.

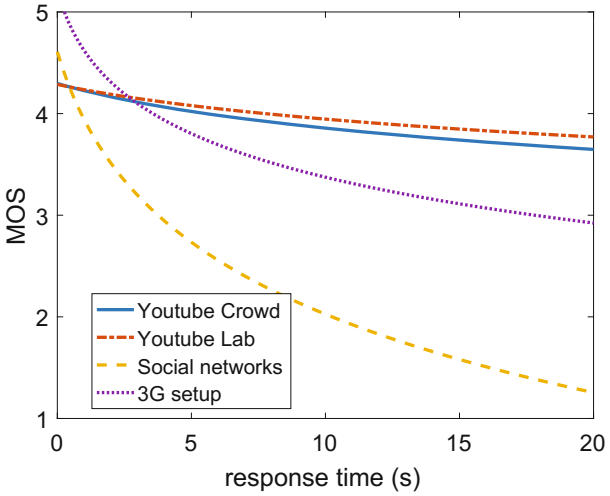


Fig. 2. QoS-QoE mapping function provided in [6] for the different Internet services. The QoS parameter is the response time of the system until the service starts. YouTube considers the initial delay until the video playout starts for two different subjective studies conducted in a laboratory and via crowdsourcing. The authentication in social networks maps response times for the authentication procedure when the user logged in to MOS values. The wireless 3G Internet connection setup considers the time for a successful connection establishment and how user perceive this delay.

QoE Fairness. The QoE fairness index is defined in [8] and computed over the observed QoE values Y in a system. In particular, the standard deviation $\sigma = \mathrm{Std}[Y]$ is linearly transformed. When the QoE values are given on a QoE scale with lower bound L and higher bound H, then the fairness index is

$$\mathcal{F} = 1 - 2\frac{\sigma}{H - L} \tag{15}$$

which is on the 5-point scale with $L = 1$ and $H = 5$ as used in the paper

$$\mathcal{F} = 1 - \sigma/2\,. \tag{16}$$

The QoE fairness metric has some nice properties and has an intuitive meaning. $\mathcal{F}$ is a continuous value bounded in the interval $[0; 1]$, see [8] for a formal proof. A high value of $\mathcal{F}$ if the system is QoE-fair, low values if the system is unfair. $\mathcal{F} = 1$ means perfect fairness and all users experience the same QoE. In contrast, $\mathcal{F} = 0$ is a totally unfair system. This is observed for example if one user obtains best QoE $H = 5$ and the other gets $L = 1$ in a system of two users. Please note that the QoE fairness metric is scale- and metric-independent. Thus, it does not matter if the QoE mapping function is provided on a 5-point scale or linearly transformed to any other scale, e.g., normalized values in the interval $[0; 1]$. Due to its definition, the fairness index is also independent of the actual QoE level, i.e., whether the system achieves good or bad QoE. A system can be evaluated in terms of QoE by providing the overall QoE $\mathrm{E}[Y]$ as well as the QoE fairness $\mathcal{F}$.

4 Derivation of QoE Results

For the M/M/1-FIFO system, the response time distribution $R(t)$ of a system with given λ and μ is mapped to QoE using the corresponding QoE mapping function $f(t) = y$. Hence, the QoE distribution is $Y = f(R)$ being a continuous random variable. In order to derive the CDF $F_Y(x) = P(Y \leq y)$ of the QoE values, the inverse QoE mapping function $f^{-1}(y) = t$ is required, cf. Eq. (9),

$$f^{-1}(y) = 10^{(H-y)/a} - b = e^{(H-y)/a'} - b \text{ with } a' = a/\log 10\,. \tag{17}$$

Then, the QoE distribution is as follows.

$$\begin{aligned} F_Y(y) &= P\left(Y \leq y\right) = P\left(f(R) \leq y\right) \\ &= P\left(R \leq f^{-1}(y)\right) = F_R\left(f^{-1}(y)\right) \\ &= 1 - e^{-(\mu-\lambda)\cdot\left(e^{(H-y)/a'}-b\right)} \end{aligned} \tag{18}$$

Although the analytical solution of the QoE distribution is specified, the expression for the overall QoE (first moment), as well as for the QoE fairness requiring the second moment leads to rather complex equations. Therefore, the results are derived numerically. The PDF $p(y) = \frac{dF_y}{dy}$ is numerically derived based on the complex-step derivative approximation [14].

The overall QoE is then numerically derived by taking into account the bounds of the QoE scale, see for example the PDF in Fig. 3,

$$E[Y] = \int_{-\infty}^{\infty} y \cdot p(y)dy = \int_{1}^{Q_m} y \cdot p(y)dy + L \cdot P_L + Q_m \cdot P_m \tag{19}$$

with the probability for the lower bound $P_L = P\left(Y = L\right) = P\left(R \geq t_L\right)$ and the probability of the upper bound $P_m = P\left(Y = Q_m\right) = P\left(R \leq t_m\right)$. Please note that the upper bound may be a value $Q_m < 5$, see Eq. (9). In a similar way, the second moment $E[Y^2]$ is derived which allows to compute the variance $\mathrm{Var}[Y] = \mathrm{E}\left[Y^2\right] - \mathrm{E}[Y]^2$, standard deviation $\mathrm{Std}[Y] = \sqrt{\mathrm{Var}[Y]}$, and QoE fairness $\mathcal{F} = 1 - \mathrm{Std}[Y]\,/2$. Please note that the symbolic math toolbox from MATLAB®was used to exactly compute the (lengthy and complex) expressions for the first and second moments of the QoE values which are omitted here. Instead, only the numerical results are provided in the following section.

5 Numerical Results

In this section we briefly discuss the results of joining the performance analysis of the $M/M/1-FIFO$ system with the QoE models described in Sect. 3.

Previously, on Fig. 1, we saw how the response time distribution for the $M/M/1$ system varies with the load ρ. We can also see, in Fig. 2, how the QoE

mappings approximate the quality perceived by the users, for the three Internet services, as a function of the response time.

In Fig. 3, we can see the CDF and PDF of the QoE estimates (MOS values, in this case) in a case where the system is highly loaded ($\rho = 0.9$). These results were obtained by composing the QoE mappings with the response time distribution observed for in the $M/M/1 - FIFO$ system with the given load.

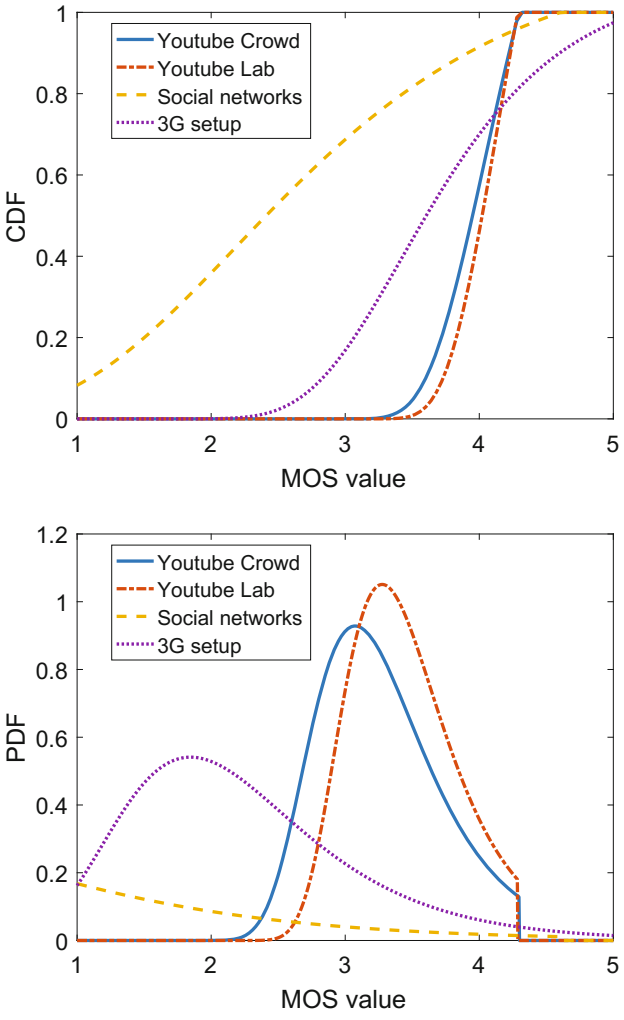


Fig. 3. CDF and probability density function (PDF) of the QoE quantified as MOS values for different applications with a system load $\rho = 0.9$. The QoE mapping functions (see Fig. 2) were applied to the distribution response time R observed in an M/M/1–FIFO queue with $\rho = 0.9$.

In Figs. 4 and 5, we can see the overall QoE and QoE fairness, respectively, for the $M/M/1-FIFO$ system, as a function of the system load. As the reader probably noticed in the previous figures, there are clear differences between services in how the users perceive the impact of response time on the quality. In particular we note that the social network login case shows the worst quality of the lot. When looking at the fairness of this service, we can see that it goes up at the end of the load scale (after ∼0.9). This indicates that a large proportion of users is already experiencing the lowest possible quality at that stage, and hence the fairness goes up once more (remember that QoE fairness is independent of the overall QoE; it only reflects the variation in quality observed among users).

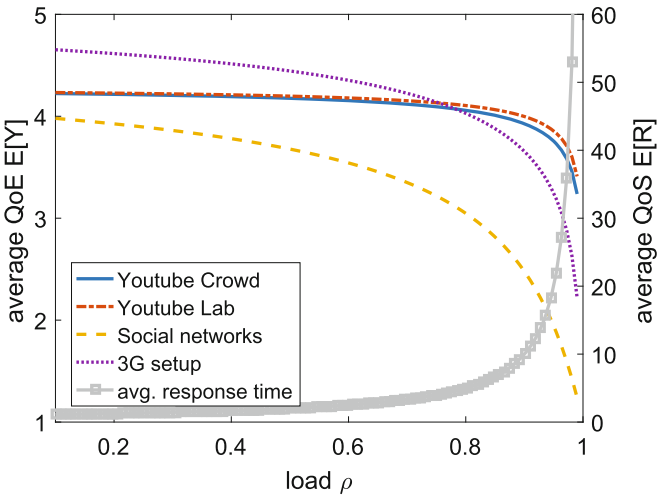


Fig. 4. The overall QoE of the system is expressed as average QoE $E[Y]$ over all users who experience QoE Y. The QoE Y is a random variable which is a function of the response time R, i.e., $Y = f(R)$ with the corresponding mapping functions $f(r)$ for the different services. The average response time $E[R]$ is plotted on the right y-axis depending on the various system utilizations ρ.

Finally, Fig. 6 plots the overall QoE against QoE fairness. We can observe, once again, very different behavior between the social media login case, and the others, as well as an overall lower fairness of both the 3G setup and the social media login cases when compared to the video streaming ones (which is to be intuitively expected, as some initial delay in video streaming is common and thus expected by the users). The variation in the shape of the QoE fairness curves can also be related to the distribution of the QoE scores, as observed in the PDF plots in Fig. 3, where both the social media login and 3G connection setup show a larger variability in the scores, as well as a higher skew towards the lower end of the quality scale.

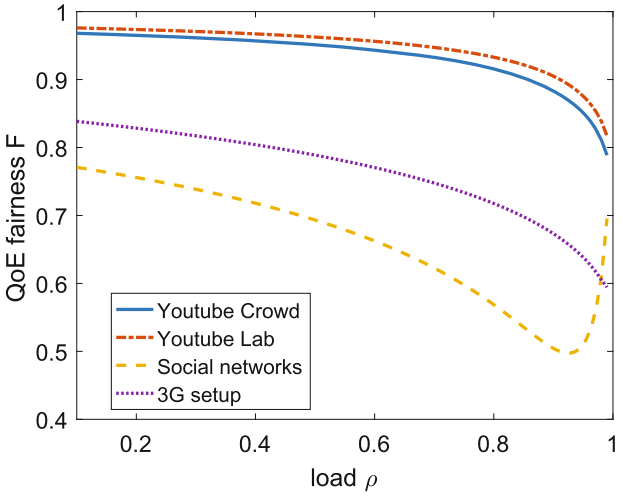


Fig. 5. The QoE fairness $\mathcal{F}$ of the system is investigated for different services.

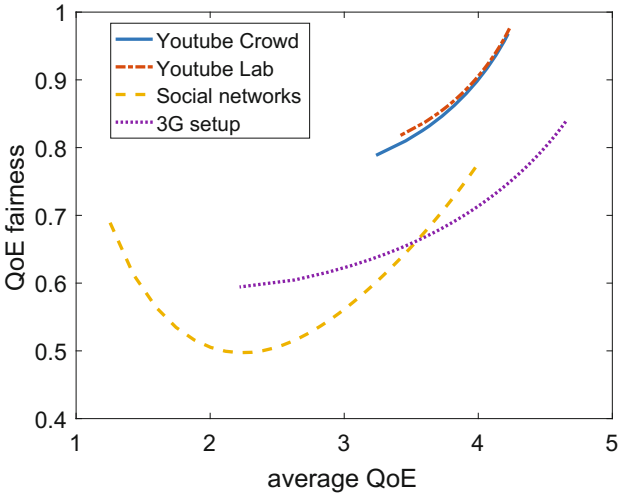


Fig. 6. Scatter plot of the average QoE $E[Y]$ and the QoE fairness values $\mathcal{F}$.

6 Conclusions and Outlook

This tutorial paper introduces a framework for the user-centric analysis of queueing systems in which response times are mapped to QoE values. For the end user, those response times manifest as waiting times before service consumption. As a simple example, an M/M/1–FIFO system is investigated for the setup of different Internet services: (1) YouTube until the video playout starts, (2) authentication in social networks, (3) wireless 3G Internet connection setup. For the analysis

of the system, the overall QoE as well as the QoE fairness are used. These two measures allow for example to properly dimension a system such that the users obtain a good QoE while achieving fairness in terms of QoE among users. The numerical results suggest that the interpretation of the system behavior in terms of QoE significantly differs for certain services. But it can also be seen that for the dimensioning of the service rates it is sufficient to consider the overall QoE only. A lower overall QoE reduces also the QoE fairness up to a certain point. This arises from the fact that higher system load in FIFO queues also leads to higher variances in the response time. If the load in the system exceeds a certain threshold, then the overall QoE is poor or even worse. When all users are suffering, the fairness increases due to decreased variances in QoE, but the system is not working in an acceptable way for the end users.

Future work will address different scheduling strategies to evaluate them in terms of overall QoE and QoE fairness. To this end, it is also interesting to investigate more sophisticated QoE metrics like 10%-quantiles or ratio of satisfied users which may be more appropriate for QoE dimensioning. Nevertheless, the same framework may be followed to analyze such systems with different metrics. This type of analysis can also be extended to consider other types of systems, as well as other types of services. For example, bounding the system capacity (i.e., an $M/M/1/K$ system) allows us to consider other performance aspects beyond time, such as the loss process in the network (e.g., deriving loss rates and average loss burst sizes from the system's load), which in turn allow us to consider other QoE models for e.g., real-time media applications, for which losses are a very important influencing factor. In the case of interactive media services, both delays and losses are important, and this type of approach allows us to attack this problem.

References

1. Altman, E., Barakat, C., Ramos, V.M.R.: Queueing analysis of simple FEC schemes for IP telephony. In: Proceedings of INFOCOM 2001, pp. 796–804 (2001)
2. Brunnström, K., Beker, S.A., De Moor, K., Dooms, A., Egger, S., Garcia, M.N., Hossfeld, T., Jumisko-Pyykkö, S., Keimel, C., Larabi, M.C., et al.: Qualinet white paper on definitions of Quality of Experience (2013)
3. Egger, S., Hoßfeld, T., Schatz, R., Fiedler, M.: Waiting times in quality of experience for web based services. In: Fourth International Workshop on Quality of Multimedia Experience (QoMEX), pp. 86–96. IEEE (2012)
4. Egger, S., Reichl, P., Hoßfeld, T., Schatz, R.: "Time is bandwidth"? narrowing the gap between subjective time perception and quality of experience. In: IEEE International Conference on Communications (ICC), pp. 1325–1330 (2012)
5. Emstad, P.J., Heegaard, P.E., Helvik, B.E., Paquereau, L.: TTM4110 Dependability and Performance with Discrete Event Simulations. Textbook in MSc course. Norwegian University of Science and Technology (NTNU) (2016, Unpublished)
6. Hoßfeld, T., Egger, S., Schatz, R., Fiedler, M., Masuch, K., Lorentzen, C.: Initial delay vs. interruptions: between the devil and the deep blue sea. In: Fourth International Workshop on Quality of Multimedia Experience (QoMEX), pp. 1–6. IEEE (2012)

7. Hoßfeld, T., Heegaard, P.E., Varela, M., Möller, S.: QoE beyond the MOS: an in-depth look at QOE via better metrics and their relation to MOS. Qual. User Experience **1**(1), 2 (2016)
8. Hoßfeld, T., Skorin-Kapov, L., Heegaard, P.E., Varela, M.: Definition of QoE fairness in shared systems. IEEE Commun. Lett. **21**(1), 184–187 (2017)
9. Jensen, J.L.W.V.: Sur les fonctions convexes et les inégalités entre les valeurs moyennes. Acta Math. **30**(1), 175–193 (1906)
10. Kleinrock, L.: Queueing Systems. Theory, 1 edn., vol. 1. Wiley-Interscience, New York (1975)
11. Kobayashi, H., Mark, B.L., Turin, W.: Probability, Random Processes, and Statistical Analysis. Cambridge University Press, Cambridge (2011)
12. Lorentzen, C., Fiedler, M., Johnson, H.: On user perception of safety in online social networks. Int. J. Commun. Netw. Distrib. Syst. **11**(1), 77–91 (2013)
13. Lorentzen, C., Fiedler, M., Johnson, H., Shaikh, J., Jørstad, I.: On user perception of web login–a study on QoE in the context of security. In: Australasian Telecommunication Networks and Applications Conference (ATNAC), pp. 84–89. IEEE (2010)
14. Martins, J.R., Sturdza, P., Alonso, J.J.: The complex-step derivative approximation. ACM Trans. Math. Softw. (TOMS) **29**(3), 245–262 (2003)
15. Reichl, P., Egger, S., Schatz, R., D'Alconzo, A.: The logarithmic nature of QoE and the role of the Weber-Fechner law in QoE assessment. In: IEEE International Conference on Communications (ICC), pp. 1–5 (2010)
16. Rubino, G., Varela, M.: Evaluating the utility of media–dependent FEC in VoIP flows. In: Solé-Pareta, J., Smirnov, M., Van Mieghem, P., Domingo-Pascual, J., Monteiro, E., Reichl, P., Stiller, B., Gibbens, R.J. (eds.) ICQT/QofIS/WQoSR - 2004. LNCS, vol. 3266, pp. 31–43. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30193-6_4
17. Skorin-Kapov, L., Varela, M.: A multi-dimensional view of QoE: the ARCU model. In: 2012 Proceedings of the 35th International Convention, MIPRO 2012, Opatija, Croatia, 21–25 May 2012, pp. 662–666 (2012)
18. Wamser, F., Casas, P., Seufert, M., Moldovan, C., Tran-Gia, P., Hossfeld, T.: Modeling the YouTube stack: from packets to quality of experience. Comput. Netw. **109**, 211–224 (2016)
19. Weber, E.: Annotationes anatomicae et physiologicae (anatomical and physiological annotations). CF Koehler, Leipzig (1851)
20. Wolff, R.W.: Poisson arrivals see time averages. Oper. Res. **30**(2), 223–231 (1982)

Industrial, Practical Experience and PhD Track Papers

VirtuWind – An SDN- and NFV-Based Architecture for Softwarized Industrial Networks

Ermin Sakic[1,2(✉)], Vivek Kulkarni[1(✉)], Vasileios Theodorou[3], Anton Matsiuk[4], Simon Kuenzer[4], Nikolaos E. Petroulakis[5], and Konstantinos Fysarakis[5]

[1] Siemens AG, Munich, Germany
{ermin.sakic,vivekkulkarni}@siemens.com
[2] Technical University of Munich, Munich, Germany
[3] Intracom SA Telecom Solutions, Athens, Greece
[4] NEC Europe Ltd., NEC Laboratories Europe, Heidelberg, Germany
[5] Foundation for Research and Technology-Hellas, Heraklion, Greece

Abstract. VirtuWind proposes the application of Software Defined Networking (SDN) and Network Functions Virtualization (NFV) in critical infrastructure networks. We aim at introducing network programmability, reconfigurability and multi-tenant capability both inside isolated and inter-connected industrial networks. Henceforth, we present the design of the VirtuWind architecture that addresses the requirements of industrial communications: granular Quality of Service (QoS) guarantees, system modularity and secure and isolated per-tenant network access. We present the functional components of our architecture and provide an overview of the appropriate realization mechanisms. Finally, we map two exemplary industrial system use-cases to the designed architecture to showcase its applicability in an exemplary industrial wind park network.

1 Introduction and Background

SDN and NFV promise the programmable connectivity and rapid service provisioning [1]. However, in their current state, a number of modifications to the state-of-art SDN-/NFV-architectures are required to accommodate the requirements of critical infrastructure providers. VirtuWind aims to fill this gap by defining a unified SDN- and NFV-architecture that provides for QoS-constrained end-to-end (E2E) connectivity in intra- and inter-domain connectivity scenarios.

A typical power control system comprises a collection of various monitoring and control components connected to a remote operator's grid control system. For example, in wind parks Supervisory Control and Data Acquisition (SCADA) is the main monitoring and management component deployed locally on-site. It is utilized for data collection and analytics, as well as for the remote configuration of setpoints of the turbine-internal controllers. The SCADA server is a sub-component responsible for controlling the power output of multiple different wind turbines, and adaptation of the total power output to the requirements received from the grid operator. Based on our traces from an operational wind park, a full cycle of a correct SCADA control loop execution (i.e. the collection of sensor measurements and SCADA's response) has a periodicity of $100 - 200$ ms,

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 251–261, 2018.
https://doi.org/10.1007/978-3-319-74947-1_17

from which we derive the uni-directional E2E delay requirement of $<\sim 30$ ms. Network availability requirements of a wind park correlate with the application availability requirement. The fail-over time of ~ 50 ms [2], as well as the requirement of 99,99% availability (equaling 50 min downtime p.a. [3]) impose an important failure resilience task for both SDN control-and data-planes.

The VirtuWind architecture aims to fulfill the four key objectives depicted in Fig. 1, using a combination of SDN- and NFV-technologies in a multi-operator ecosystem. We define three deployment steps necessary to enable the appropriate solution. The corresponding architecture is then derived in Sect. 2.

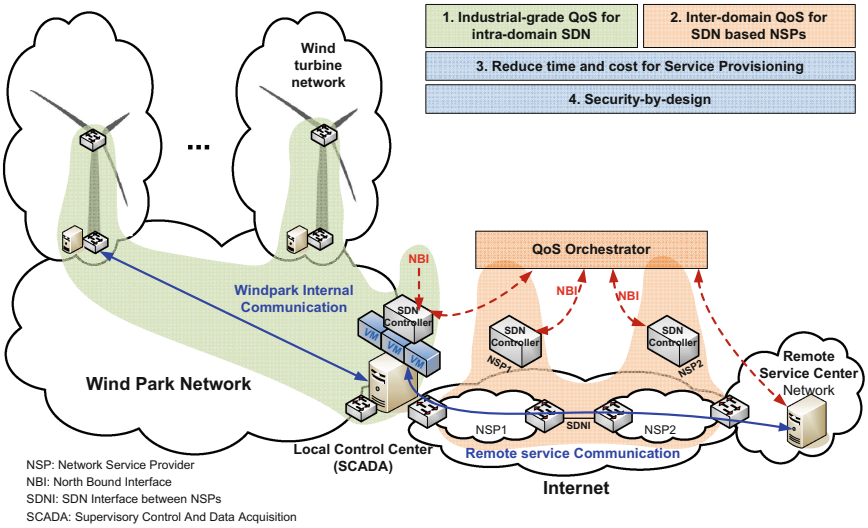


Fig. 1. VirtuWind architecture objectives

Step 1 - System Bootstrapping: The deployed wind turbines are often dislocated from the centralized SCADA, thus deployment of an *in-band* (shared) control network represents a strict requirement, opposite to the *out-of-band* (exclusive) SDN controller-switch links often encountered in the data-center SDN deployments. Second, static appliances such as the data historians that collect and store large amounts of wind turbine monitoring data, should be placed so to optimize the resource sharing (i.e. the storage and compute resources). In the current non-virtualized deployments, the data historian components are deployed on dedicated industrial PCs inside the wind turbines. Thus, large cost savings are achievable by the centralization (and appropriate high-availability mechanisms in place) of such software appliances on dedicated "micro-cloud" nodes. Third, security components, such as firewalls, must be deployed for securing the external access to the industrial intra-domain network.

Step 2 - Enabling Intra-Domain Connectivity: Traditionally, during system updates/upgrades the risk of impacting the SCADA control process is high,

but for the regulatory reasons updates need to be performed regularly within a given time-frame, e.g. adapting the SCADA process services invoked by application or new service updates. The specific requirements in an industrial architecture come from e.g. an exemplary device upgrade workflow: (1) Maintenance tenant initializes a firmware update process using the SCADA interface; (2) The request is handled by the authentication entity in SCADA and the user is allowed or denied the service access. (3) If the user's request is accepted, the part of SCADA responsible for interactions with the SDN Controller triggers an event and request network "slice" with fixed time schedule and QoS support.

Step 3 - Enabling Inter-Domain Connectivity: The third deployment step enables the management of multiple wind park sites from remote locations, e.g. by a third party grid operator. To ensure a successful remote management, an inter-domain QoS enabled E2E connectivity is required. We foresee a centralized QoS approach, where an inter-domain coupling of SDN controllers and a centralized QoS orchestrator enables the E2E connectivity. Our approach involves four phases: 1. *Domain Registration*; 2. *Announcement of network path segments*; 3. *Centralized E2E path computation* and 4. *Path establishment.*

2 The VirtuWind Architecture

VirtuWind envisions a layered architecture leveraging the control and programmability offered by the SDN paradigm and exploiting the flexibility of NFV. Each authorized application or tenant that needs connectivity requests an appropriate Virtual Tenant Network (VTN), and later issues communication service flow requests to the system. The flow requests are a combination of different connectivity and QoS service requirements, ranging from E2E-delay and bandwidth requirements, to different path protection schemes (e.g. duplication or fast-failover). The isolation of tenants is administered using the VTN northbound interface of the SDN Controller, while the service requirements define a communication service interface as per tenant's intent specification. Each application service is mapped to a unique VTN and a network tenant.

Figure 2 depicts the key architecture blocks of the VirtuWind architecture: (1) *Business Applications* that interact with the underlying network and impose new service requirements at the centralized controller; (2) *SDN Controller* and *QoS Orchestrator*, which receive application requests, execute the centralized decision-making and configure the infrastructure; (3) *NFV Management and Orchestration (MANO)* which orchestrates virtual network functions and service function chains of the industrial network; (4) *Edge Devices* that allow for stretching of VTNs down to the last hop, as well as intent monitoring.

In the remainder of this section, we give an overview of the components depicted in Fig. 2 and discuss their internal-workings. In Sect. 3 we then outline the mapping of two exemplary processes on top of the introduced components.

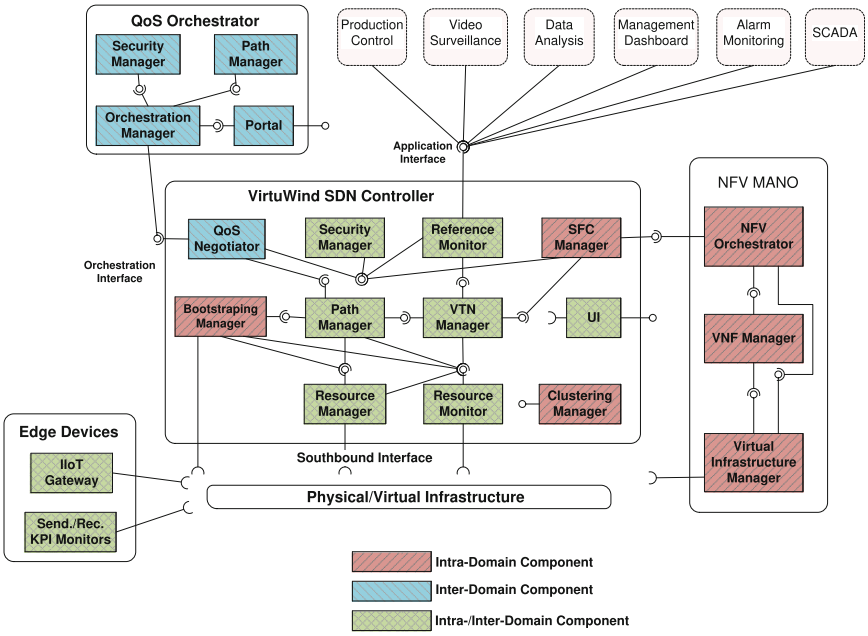


Fig. 2. The combined VirtuWind intra- and inter-domain system architecture.

2.1 Bootstrapping Manager

Industrial SDN networks require a highly-available, *in-band* control plane [4]. By means of an automated network bootstrapping procedure, VirtuWind guarantees a robust and resilient control plane configuration during the network runtime. The robustness to controller failures is ensured by bootstrapping a multi-controller state replication design. To handle the data plane failure effects on the control plane flows, we leverage redundant control flow embedding. While recent works propose slower, restoration-based techniques in industrial scenarios [4], we use 1+1 protection [5] by duplicating controller-to-controller and controller-to-switch TCP-based flows on maximally disjoint paths, thereby ensuring zero packet loss for control flows, at the expense of doubled bandwidth requirements per control flow connection. Since these typically have low bandwidth requirements, we do not consider it a crucial drawback in our approach.

2.2 Reference Monitor (RFM)

RFM [6] is a component that interacts with the northbound applications via the North-bound Interface (NBI). NBI is designed in line with the principles of Intent APIs [7], hiding complexity of the underlying infrastructure.

Intra- and Inter-domain: RFM is involved in authentication of the applications and authorization of their service requests. During application authentication phase, RFM acts as a proxy between the application and Security

Manager (SM). It receives initial credentials from the application and passes them to the SM. In case of a successful authentication, all subsequent application requests are evaluated against the authentication credentials approved by the SM. For authorization purposes all communication requirements contained in the service request are checked against pre-configured application access profiles.

Inter-domain: In inter-domain service setups, RFM additionally parses inter-domain requests and, if communication endpoints are located in different domains, splits application requests into intra- and inter-domain parts. The intra-domain part of an initial application request is forwarded to VTN Manager (VTNM) in which RFM specifies the VTNs gateway as the endpoint in the intra-domain part. The inter-domain part of the request is forwarded to QoS Negotiator and contains information about endpoint domains and service request requirements including QoS metrics. If QoS Negotiator receives a reply by the QoS Orchestrator that the inter-domain part requirements cannot be met, based on announced resources of all registered NSP domains, the request is declined.

2.3 VTN Manager (VTNM)

Intra-domain: VTNM represents an extension of an existing framework[1] that realizes network slicing and exposes a set of corresponding APIs. By means of the VTN APIs one can create isolated virtualized network slices (VTNs) and provide L2/L3/L4 network forwarding functionality for such slices via virtual API primitives. VTNM maps these virtualized slices into physical infrastructure and enables forwarding via flow rules, at the same time ensuring slice isolation. The VTNM interacts with the Path Manager (PMG) to request for path computation of best-effort and mission-critical paths. Additionally, VTNM provides a VTN specification interface for the IIoT Gateway (ref. Sect. 2.12).

Inter-domain: A specific virtual "gateway" interface is defined in every VTN, that is mapped to a physical interface of the domains border gateway, which serves as an exit point of the particular VTN in inter-domain.

2.4 Path Manager (PMG)

Intra-domain: For the guaranteed industrial QoS, i.e. the bandwidth provisioning, flow isolation and worst-case delay estimation, VirtuWind proposes using network calculus, a deterministic mathematical modeling framework for communication networks. Instead of basing its routing decision on a reactive control loop of network observations, VirtuWind's PMG provides mechanisms for admission control of new flows. By maintaining an accurate model of the network state and service embeddings in the control plane [8], PMG ensures per-flow isolation and worst-case guarantees at all times.

Inter-domain: PMG handles all path configurations that need to take place on the network service provider's (NSP) network slice that is available to the

[1] OpenDaylight's VTN project: https://wiki.opendaylight.org/view/VTN:Main.

VirtuWind architecture. In cross-NSP domains, traffic can flow through transit domains, for which QoS characteristics fluctuate over time in an unpredicted fashion. Thus, it is necessary for the NSP to continuously monitor the network performance and to be capable of applying corresponding actions once abnormal deviations are diagnosed. To this end, PMG maintains a detailed internal view of the network resources, and each NSP exposes to the QoS Orchestrator an aggregated view of the available path segments available within its domain.

2.5 Resource Manager (RMG) and Resource Monitor (RMT)

Intra-domain RMG: RMG is responsible for configuration management and network control tasks, i.e. embedding of L2/L3 OpenFlow flow rules into the network. RMG provides for embedding of: (i) real-time flows which require dedicated per-queue flow assignments; (ii) best effort flows, without queue considerations; and (iii) the meter structures for policing purposes. The generated flow rules and meter structures are persisted in the distributed data-store.

Intra-domain RMT: RMT is a utility component that monitors and exposes the network state information. It provides for functionality to fetch the topology, as well as the features of the forwarding devices, providing input for PMG's routing and flow-queue mapping decisions. Furthermore, it allows for real-time monitoring of KPIs related to served intents.

Inter-domain RMG: The main difference compared to the intra-domain RMG functionality is in the use of match filters (e.g. MPLS labeling and VTN tagging), in order to enhance the scalability over large infrastructures.

Inter-domain RMT: For operator networks, due to scalability concerns, RMT performs real-time monitoring using probing and OpenFlow statistics.

2.6 Security Manager (SM)

Intra-domain: SM realizes the authentication and accounting services to the rest of the SDN Controller as well as the users and applications that interact with the controller. With respect to authentication, the SM exposes interfaces for the administration of local SDN Controller accounts. Additional APIs are exposed for applications to present their credentials. If these credentials prove valid, the SM can issue an authentication token to the requesting party. The token can then be presented to the RFM when attempting to interact with the SDN Controller. The RFM is responsible for transferring these tokens to the SM internally for validation, so the former can then proceed to evaluate the request (i.e. if it is allowed based on the active policies). In the case of distributed authentication, the SM is responsible for presenting the tokens to the server for validation.

Inter-domain: To authenticate all entities in an inter-domain scenario, two different approaches can be applied: the direct authentication on the controllers and the federated authentication via a trusted third party, which acts as an identity provider. In the former case, QoS Orchestrator that requests path segment offers,

must have credentials on each controller to be authenticated locally. This creates the complexity of cross controllers entity credential data synchronization, so the SM module is extended to add the latter approach. In the case of distributed authentication setups, whereby the QoS Orchestrator has an account created at another server, the Token Bearer [9] authentication is used. QoS Orchestrator can be authenticated by requesting an authentication token by the OpenID Connect server. When a request arrives by the QoS Orchestrator to the QoS Negotiator of the controller, it passes the authentication token to the SM to verify it. The SM then validates the provided token at the identity provider.

2.7 Clustering Manager (CMG)

The issue of the controller's single point of failure is resolved by means of state replication and fail-over to a pre-configured backup controller on failure.

Centralized controller state registry: The CMG handles the controller relationships per-data state in the VirtuWind's distributed data-store. VirtuWind's controller state as well as the up-to-date network information is collected in a single registry shard that is replicated across the multiple controller instances.

Strong (SC) and adaptive consistency (AC) primitives for update ordering: Components that have stringent requirements on the data state staleness, such as the Path Manager which makes critical routing and resource reservation decisions, may require serialized updates. Serialization ensures no data-store updates are applied without having first observed the previous history of the updates made to that state. Such components make use of the controller state distribution based on *SC primitives* (e.g. on RAFT [10] consensus). Components that tolerate a certain degree of inconsistency may rely on *AC primitives*. Our AC framework enables eventually consistent state synchronization with staleness bounds [11]. The staleness bounds are realized by limiting the amount of de-synchronization in between controller replicas.

2.8 Service Function Chaining Manager (SFCM)

In industrial networks, Virtual Network Functions (VNFs) such as Firewall, IDS, DPI, and honeypot are pertinent to ensuring secure multi-tenant operation. SFCM is able to handle VNF chaining. It invokes the VTNM in order to register external ports of the SDN transport network and to declare and associate service instances to those external ports. It exposes an interface to fetch information on and modify the existing service chains, the VTN-to-SFC mappings, as well as the service instances of the VNFs. Having the SFCM as a separate from MANO offers the advantages of having one interface to business applications, and the application does not need to be aware of the underlying SFC.

2.9 NFV Management and Orchestration (MANO)

VirtuWind does not require an implementation of a fully-fledged NFVO in the broad sense, as an inter-domain orchestration of Virtual Network Functions

(VNFs) still lacks an appropriate use-case in the industrial domain. Nevertheless, we assume an implementation of an NFVO element (i.e. the OpenStack's HEAT-API) to stay compatible to the industry standard when VNF and SFC request specification is concerned. The VNFs deployed in industrial networks typically require static, long-lived configurations, but may need appropriate preconfigurations for correct security service function chaining. Provisioning of VNF requests is initiated by the user at the NFVO component and the requests are forwarded to VIM in order to request the deployment of the VNFs. When a new VNF instantiation request is received at the HEAT components, it interacts with the VIM to configure the infrastructure in order to provision the VNF. VIM guarantees the management and the allocation of the necessary virtual resources for the VNF deployment. Similar to data-center networks, the VIM is responsible for allocating the necessary resources in an industrial "private cloud".

2.10 QoS Orchestrator (QOR)

QoS Orchestrator (QOR) is responsible for setting up a QoS-enabled end-to-end connectivity service via multiple network operator domains. There are four phases in the lifecycle of a QoS-enabled end-to-end connectivity service: (1) *Registration*: An NSP registers its domain to the QOR. (2) *Path Segment Announcement*: The registered NSP advertises its available resources. (3) *Path Segment Instantiation*: The QOR instructs the NSP to assign resources for a path. (4) *Monitoring*: The NSP periodically sends performance statistics to QOR to verify that agreed constraints are met for allocated flows.

Orchestration Manager: Coordinator of all QOR activities, handling all communication with the SDN Controllers via encrypted REST APIs.

Path Manager: Implements all path calculation logic for the establishment of inter-domain paths. Using end-point addresses as input, as well as the advertised path segment offers and corresponding QoS values, Path Manager finds zero or more "best" end-to-end paths (consisting of multiple NSP path segment offers) that satisfy specified QoS properties.

Security Manager: Implements the authentication and authorization to secure the communication between the QOR and the SDN Controller.

Portal: Front-end for the QOR administrator to trigger QOR functions manually and to visualize status information at runtime.

2.11 QoS Negotiator (QON)

QoS Negotiator (QON) is responsible for the communication between the SDN Controller and the QOR and the translation of QOR's requests to domain-specific actions. After registration, the QON receives path segment offer announcement and instantiation requests from the QOR and upon authentication and authorization via Security Manager, it replies to the QOR and propagates its requests into network actions. The information revealed to the QOR

is high level enough so as not to expose sensitive internal details about the intra-domain configuration and characteristics. In this respect, path segment announcements only expose information about available path IDs, their ingress and egress border devices and their expected QoS characteristics. In addition, the QON periodically receives monitoring messages and notification events from the Path Manager, processes and filters them and sends to the QOR relevant information. This information comprises critical updates about the topology, e.g., path segments becoming unavailable or potential threats to maintaining the advertised QoS.

2.12 Industrial IoT Gateway (IIoTG)

IIoTGs are physical devices that seamlessly integrate sensor devices to the VirtuWind network. Each IIoTG maps the sensors to VTNs and takes care of the isolation of the tenants when interacting with the sensors. An IIoTG is connected on one both to the industrial SDN network and at least one sensor network. It is then able to distinguish data flows from sensors and to forward them appropriately to assigned tenant networks. This design enables integrating sensor networks to the VirtuWind architecture that are not SDN-programmable and cannot implement any isolation (e.g., LoRa devices). A software SDN switch operating on the IIoTGs hypervisor seamlessly integrates VNFs into the VirtuWind network. A VNF is instantiated for each sensor(s)–tenant pair and represents a set or a single sensor within a tenant. NFV is approached by exploiting the idea of *unikernels*, the purpose-built operating systems that run a single targeted application. Unikernels utilize remove unnecessary kernel components in order to reduce the required resource usage [12]. This enables running a high number of virtual instances on a single resource-constrained IIoTG [12].

3 Exemplary System Workflows

This section portrays the suitability of VirtuWind architecture to common use cases of isolated virtual tenant network addition, as well as the QoS-constrained network service embedding that spans multiple VirtuWind domains.

Figure 3 depicts the scenario in which a new critical infrastructure service addition, that considers a set of QoS requirements specific to the *User A*'s applications, is embedded into the SDN network. In addition to User A, an NBI-aware application (e.g. the depicted Network Management Station), may schedule and enforce a QoS-constrained intent at any point at network runtime.

Figure 4 shows the inter-domain end-to-end path establishment scenario. It requires the interaction between the QoS Orchestrator and all the SDN Controllers of the NSP domains involved (we depict only one of those SDN Controllers) and it consists of the relevant service flow establishment for each involved NSP domain, combined with the path embedding for the endpoint industrial domain.

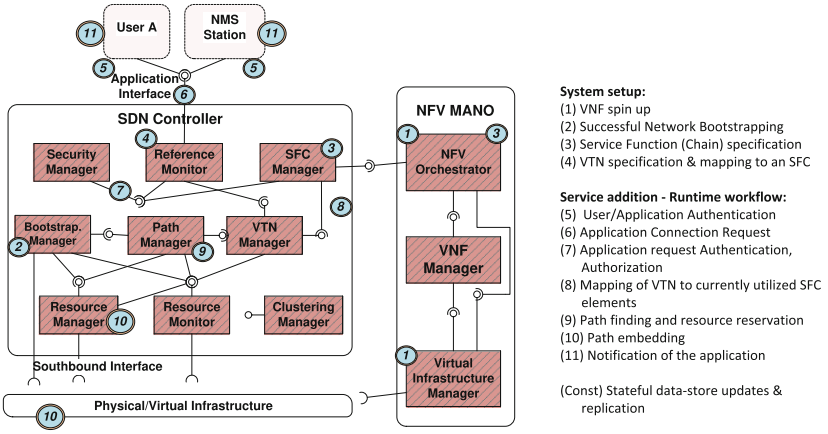


Fig. 3. An exemplary intra-domain VirtuWind system workflow.

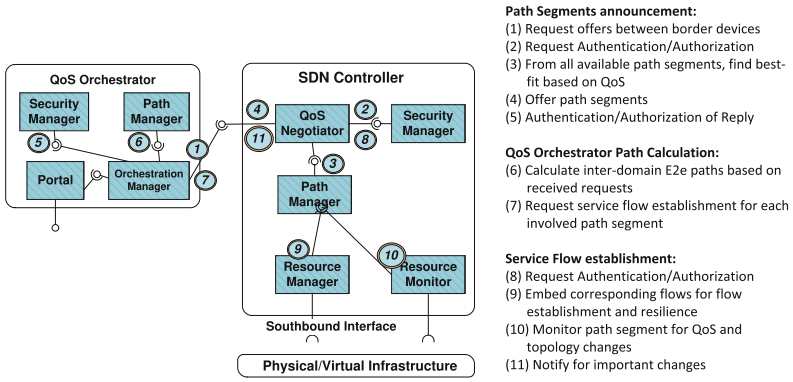


Fig. 4. An exemplary inter-domain VirtuWind system workflow.

4 Conclusion

This paper presents the VirtuWind architecture, that addresses the complex real-world requirements of an industrial network. The representative wind park control use case is briefly described and the design of an SDN- and NFV-architecture which considers industrial intra- and inter-domain requirements is illustrated in detail. We elaborate the mechanisms of individual components of the architecture and present their mapping to the architecture components.

Acknowledgement. This work has received funding from the EU's H2020 research and innovation programme under grant agreement No. 671648 VirtuWind.

References

1. 5G-PPP: White paper on "5G innovations for new business opportunities" (2017)
2. Alcatel-Lucent: White Paper: Smart Choices for the Smart Grid (2011)
3. 5G-PPP: White paper on "5G and Energy" (2014)
4. Vestin, J., et al.: Resilient software defined networking for industrial control networks. In: 10th International Conference ICICS. IEEE (2015)
5. Kirrmann, H., et al.: HSR: Zero recovery time and low-cost redundancy for industrial ethernet. In: ETFA. IEEE (2009)
6. Gkounis, D., et al.: Cases for including a reference monitor to SDN. In: Proceedings of the 2016 ACM SIGCOMM Conference, SIGCOMM 2016. ACM (2016)
7. Open Networking Foundation: Intent NBI - Definition and Principles (2016)
8. Guck, J., et al.: Achieving end-to-end real-time quality of service with sdn. In: IEEE 3rd International Conference on Cloud Networking (CloudNet). IEEE (2014)
9. Jones, M., et al.: The oauth 2.0 authorization framework: Bearer token usage. Technical report, Internet Engineering Task Force (IETF), RFC6750 (2012)
10. Ongaro, D., et al.: In search of an understandable consensus algorithm. In: USENIX Annual Technical Conference (2014)
11. Sakic, E., et al.: Towards adaptive state consistency in distributed SDN control plane. In: IEEE International Conference on Communications (ICC) (2017)
12. Kuenzer, S., et al.: Unikernels everywhere: the case for elastic CDNs In: VEE 2017. ACM (2017)

A Modular Environment to Test SCADA Solutions for Wind Parks

Jannik Hüls(✉) and Anne Remke(✉)

Westfälische Wilhelms-Universität, 48149 Münster, Germany
{jannik.huels,anne.remke}@wwu.de

Abstract. Supervisory Control and Data Acquisition (SCADA) systems monitor and control industrial processes, like power generation and distribution. Research on e.g. SCADA security and the implementation of new approaches is made difficult through the lack of accessible test beds. We investigate running a dedicated SCADA test bed on a Raspberry Pi cluster and report the experiences made during installation, configuration and administration. Build with production readiness in mind, we describe how a large level of automation contributes to both, ease of use and a reliable set-up. In the future this testbed will be used to implement new approaches for (i) process monitoring, (ii) intrusion-detection and the combination of different (iii) information and communication technologies for SCADA systems.

Keywords: SCADA · Test bed · Automation · Reliability · Testing

1 Introduction

Power distribution and (renewable) power generation, e.g. through solar panels and wind turbines is usually controlled through Supervisory Control And Data Acquisition (SCADA) systems. A highly reliable communication network is needed to ensure the secure and dependable operation of those systems. SCADA systems allow to maintain an overview of the physical system and execute commands in remote locations. Such SCADA systems consist of Remote Terminal Units (RTUs), that connect physical devices to read sensor data and to control actuators. Each RTU is connected to a SCADA master unit and the operator can view collected data and overrule control decisions via the Human Machine Interface (HMI). A communication infrastructure and dedicated protocols connect all components of the SCADA system.

The development of secure and reliable SCADA systems is a very broad and active research area. For example intrusion detection mechanisms [6] and machine learning [4,10] technologies are used in SCADA systems to develop reliable and secure *control mechanisms*. While *Security* vulnerabilities of SCADA networks have long been known [9], they have become more important since an increasing level of automation and has brought SCADA systems and corporate networks [5] closer together. Especially new *Fast Data* technologies, which

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 262–272, 2018.
https://doi.org/10.1007/978-3-319-74947-1_18

allow forecasts by processing real-time data [2] lead to interaction between those networks. However, the feasibility and potential of all those research directions need to be tested. While *Simulation* environments [1,13] can be adapted, building a simulation that closely resembles the real system and includes the control network can be cumbersome. Accessing and working with *real systems* is often impossible. Hence, a *test bed* might be an alternative for research and education in a self-controlled environment that allows working in a system that employs the same communication set-up, as the real system.

We report on experiences made during the installation, configuration and administration of a SCADA test bed dedicated to wind parks. Furthermore, we discuss possible tests that will be conducted in the test bed in the future. By using a high level of automation in the set-up and installation of the proposed SCADA test bed, we are building an environment that can readily be deployed.

Many SCADA test beds have been presented in the past. National governments replicate existing SCADA systems with identical devices, as e.g., the National SCADA Test bed (NSTB) implemented by the United States department of energy [3], the national SCADA test bed implementation in Idaho [12] and the European SCADA security test bed [9]. They provide high quality real-world SCADA systems, however access is restricted and reconstructing such a test bed, e.g. for local research or education is very costly.

As shown in [7], several implementations of smaller scale test beds exist. They primarily focus on testing system security and are often focused on a very specific application area. Instead, our test bed provides a highly reliable infrastructure to test different improvements of SCADA systems. We believe that a high level of automation eases extending the current implementation in a reliable manner. The proposed test bed is currently used in graduate-level education and allows master students to understand the ins and outs of SCADA systems.

The current implementation uses data from a wind park that is operated by a large German energy supplier to emulate wind engines, which currently prevents an open source publication. However, we plan to open source a limited version, based on simulation in the near future.

The report is organized as follows. Section 2 presents the system architecture and its functionality. Section 3 describes the implementation of the Remote Terminal Unit and Sect. 4 discusses the SCADA architecture. We outline the measures taken w.r.t. automation, testing and monitoring in Sect. 5. Section 7 concludes the paper and and gives an outlook into future work.

2 System Architecture

Supervisory Control And Data Acquisition (SCADA) systems, which control distributed industrial processes often use a layered architecture. We propose a two-layer architecture, where the second layer locally controls individual wind turbines and the first controls the wind park. Per wind turbine 70 data items are collected and can be controlled, individually. For example, the temperature is controlled locally such that it stays below a certain threshold to prevent damage

in the motor. Additionally, some variables like the total amount of produced power require control on an aggregated level, i.e., the first level of the presented architecture, which is also used to detect so-called outliers, i.e., wind turbines that behave differently from the others in the same wind park.

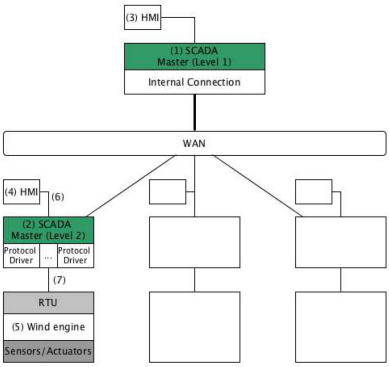


Fig. 1. Reference architecture of a SCADA wind park system.

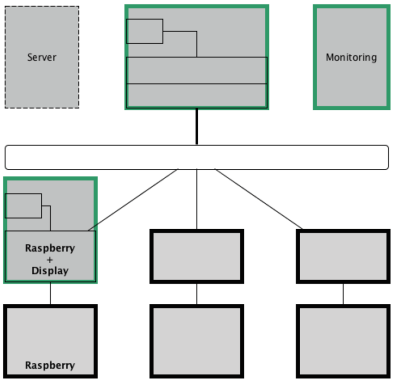


Fig. 2. Assignment of SCADA components to Raspberry Pies.

Figure 1 illustrates the two level architecture used in our test bed. Every wind turbine is controlled by a local SCADA Master (2). This SCADA System controls the local wind engine (5) data and the second level SCADA Master is able to use several protocols to communicate with the RTU of the wind engine. The test bed currently runs on Modbus/TCP [15], while we are implementing other protocols like IEC60870-5 [11] and MQTT-S [8]. The first level SCADA master (1) aggregates and controls data on a global level.

The separation of the two SCADA levels by a wide area network indicates their connection through other communication channels, potentially with higher latency and lower bandwidth. Human machine interfaces (HMI) on each level display state information and allow to regulate the wind turbines. We implemented a dedicated HMI using JavaFX which connects to the Java APIs of our SCADA implementation. The second level HMI (4) displays sensor information of the local wind engine. Furthermore it offers a graphical interface to control actuator states. The first level HMI (3) provides remote access to each second level SCADA master. It also displays the state of the whole wind park. Figures 3 and 4 show the GUI implemented for the second level SCADA System. Figure 3 shows the most important sensor data on an engine dashboard, giving the user clear information on the current system status. A detailed overview of all provided sensor data is also available, as shown in Fig. 4. Actuator control information can immediately be set in the user interfaces. To allow the user to remotely access a wind engine an additional GUI is provided for the first level.

Fig. 3. The HMI Dashboard showing most import sensor data information.

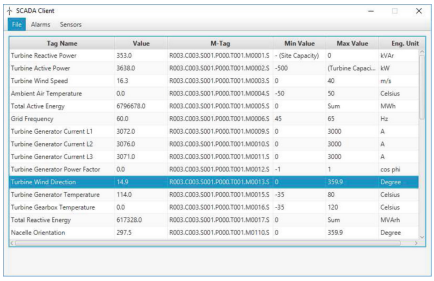


Fig. 4. Detailed overview of all sensor data provided by a wind turbine.

The test bed is implemented using a cluster of twelve Raspberry Pis, Model 3[1]. As such, it is large enough to implement a layered architecture, but can also serve as a portable showcase. Figure 2 illustrates the mapping between SCADA components and computing units. Each wind engine and each Level 2 SCADA master is implemented on a dedicated Raspberry Pi. To present a Level 2 HMI, one of the masters is connected to a seven inch touch screen. Likewise, the first level SCADA masters run on a Raspberry Pi connected to a touch screen. The overall monitoring node runs on a dedicated Raspberry Pi and uses a touchscreen to show the state of the test bed. The server acts as build pipeline, configuration manager and database and hence is implemented on a dedicated external computing node as it requires more computation power. To facilitate access to the test bed, we are currently developing a Docker compose, where test bed components are translated into multi-container Docker applications[2] which can run independently of the environment.

3 Implementation of Remote Terminal Units

The developed SCADA systems controls wind park data that is represented as a structured string Rx-Cx-Sx-Px-Tx-Mx. Each data field is separated using a dash. $\mathbf{x}$ is a placeholder for $\mathbf{x} \in \mathcal{N}_{>0}$. The meaning of each data field is summarized in Table 1. The tags R, C, S, P and T together uniquely identify each controlled wind turbine. The meta Tag M refers to the different sensor data fields provided by the turbine.

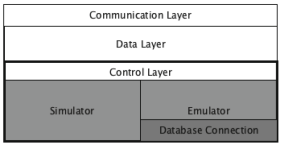


Fig. 5. Components of a remote terminal unit implementation.

Each Remote Terminal Unit (RTU) represents a wind turbine providing sensor data and the possibility to change the actuator states. It uses a layered

[1] Raspberry Pi 3 Model B having 1,2 GHz QuadCore 64Bit CPU. https://www.raspberrypi.org/.

[2] https://docs.docker.com/compose/.

Table 1. Wind park data: Fields and their meaning.

Rx	Cx	Sx	Px	Tx	Mx
Region	Country	Site	Park	Turbine	M-Tag

architecture as illustrated in Fig. 5. A *communication layer* currently implements Modbus/TCP. It abstracts from the specific communication channel to simplify extensions. The *data layer* ensures that all data gathered from the *content layer* contains at least the information shown in Table 1. Control information may be added to data that is sent or received, however this is transparent to the underlying *content layer*, which distinguishes between *emulation* and *simulation* mode. The former one reads data from the time series database InfluxDB[3]. The latter uses a wind engine simulator. A park in general consists of multiple wind engines. Once emulation is started, more engines can be added to the emulated park on the fly. However, note that they need to synchronize with the time of the SCADA system and the time stamps used in the database.

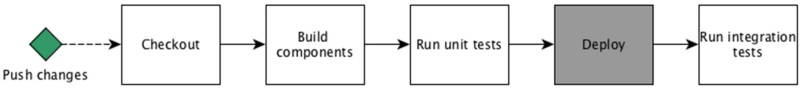


Fig. 6. Components of the Jenkins automation pipeline.

As *emulation* mode only works with prerecorded data it does not include adaptive behavior. *Simulation* mode implements a standalone test bed which incorporates true adaptive behavior and control. For example, the simulator may change the blade pitch angle with leads to slower rotation. We are both implementing a wind turbine simulator using Python as well as making use of existing wind engine models built using MathLab [14].

4 SCADA System Implementation

The implementation is based on the Java-based project Eclipse NeoSCADA[4] for both SCADA levels. NeoSCADA, while being open-source, is used in several productive 24/7 installations. It provides many out-of-the-box features but also offers well documented Java-APIs. Each level 2 SCADA node consists of a master instance which aggregates and controls data. Communication uses so called *device driver*. They read sensor data using Modbus/TCP or IEC 80870 by default and forward them to the master instance. Instead of drivers, Eclipse NeoSCADA also allows to connect multiple SCADA systems using an internal non-standardized communication channel, which we use to implement the

[3] InfluxDB - An open-source distributed time series database. http://influxdb.com.
[4] Eclipse NeoSCADA. https://www.eclipse.org/eclipsescada/.

proposed two level architecture. Configuring the test bed is a crucial step. We require a highly reliable set up, hence manual configuration is not an option. By default Eclipse NeoSCADA is configured using the Eclipse GUI. The global system configuration is organized as an Eclipse[5] project determining all the physical devices of the system as well as the hierarchy of the sensor data items. Once a configuration is build, a *.deb package is created and ready to be deployed to the corresponding SCADA master instance. All these manual steps are automated by extending Eclipse NeoSCADA with a custom OSGI[6] instance, which automates configuration based on configuration files or reading from configuration databases.

5 Automation for Reliability and Reproducibility

Since we aim to construct a productive environment, automation of system installation and configuration as well as monitoring the running SCADA system are important. Crucial steps are (i) the reproducible installation and configuration of new master instances and RTUs, (ii) zero downtime updates of existing nodes and (iii) ensuring a 24/7 operation. Furthermore, a high level of automation should ease changing and extending the system.

Installation and configuration of each Raspberry Pi is done automatically using Ansible[7]. Ansible allows creating so called Playbooks to install, configure and administrate software components. In contrast to manual steps being executed repeatedly, Ansible keeps infrastructure descriptions in a text-based format saved in version control systems. Hence, an audit of every change is possible and long lasting manual steps can be executed repeatedly in a reproducible way. Each of these Playbooks implicitly defines one layer of the complete configuration. Several layers and therefore Playbooks may be used to configure different infrastructure components.

Extended automation is implemented using Jenkins[8]. For the custom software implementations, like the HMI, that need to be build and deployed automatically. The Jenkins server automatically detects any change in the version control system and runs a specific build pipeline as presented in Fig. 6. Changes are pushed to the version control systems observed by Jenkins and automatically checked out if change is detected. Jenkins builds the artifacts and runs unit tests. If successful, again Ansible Playbooks are used to deploy these artifacts to the dedicated Raspberry Pis. It is crucial to ensure a so-called zero-downtime deployment, which ensures that service can continuously be provided also during updates. Finally, integration tests are used to ensure that the system still acts as expected. This high level of automation and testing enables the construction of a production-ready SCADA test bed based on state of the art software development and infrastructure configuration paradigms.

[5] Eclipse IDE. http://www.eclipse.org/.
[6] OSGI. A dynamic module system for Java. https://www.osgi.org/.
[7] Ansible. Automation of your IT Infrastructure. https://www.ansible.com/.
[8] Jenkins. Build and automation server. https://jenkins.io/.

The 24/7 operation of a SCADA system is very important. Even though our system is a test bed, we want to continuously observe its performance. Therefore the monitoring tool Icinga2[9] is used to both check system and service status. The monitoring master runs on a Raspberry Pi and is equipped with a touch display. Every other Raspberry Pi runs a monitoring agent to gather status information. Currently, system status like used memory, swap, cpu-load and ping as well as Java process information are gathered and displayed on the monitoring master. It provides a dashboard like GUI that displays the state of the complete SCADA test bed at a glance.

6 Measurements

In the following, we present some basic metrics that describe the performance of three different parts of the testbed: (i) the installation and configuration, (ii) SCADA-specific node metrics and (iii) network specific metrics.

Installation and Configuration. The installation and configuration phase is crucial and as we focus on a high degree of automation in this process, we translate several steps into Ansible playbooks. The steps needed to configure and install a SCADA node on a Raspberry Pi and their durations are summarized in Table 2.

Most configuration tasks have a comparable length between 100 ms and 300 ms, Tasks 3, 7 and 11 take about one order of magnitude longer and Task 4 even takes about 280 s to complete. It takes 295,8 s in total to configure a SCADA node using Ansible playbooks. The installation of a SCADA system takes another 170 s, with Tasks 23, 24 and 31 forming the bottlenecks of the installation. Long tasks are critical to the performance of a highly automated environment. Identifying such bottlenecks allows to split them into smaller tasks, which improves the overall performance especially in the presence of failures.

SCADA Node Metrics. To investigate the system under varying loads, Modbus messages are sent with different frequency to the Raspberry node hosting the SCADA component. The systems CPU share of the SCADA system and of the Modbus device have been measured and are plotted for inter-arrival times of 10 ms, 50 ms, 100 ms, 200 ms and 500 ms in Figs. 7 and 8, respectively. The relatively low CPU usage in the beginning and end of the plots is due to manually handling the measurements. Measuring the memory share under different loads has shown that memory usage is independent of the number of write requests sent.

Network. To obtain insight into network traffic, the system was monitored for 12 h using Bro. We observed three different categories of network traffic. First of all, we have counted the number of network packets sent or arrived on the different test bed nodes as presented in Fig. 9. The numbers are obtained every

[9] Icinga2. Monitoring platform. https://www.icinga.com/products/icinga-2/.

Table 2. Duration of steps to configure and install SCADA nodes.

#	Description - Configuration	Time	#	Description - Installation	Time
1	Change the access rights	0.16 s	21	Copy sources.list	3.11 s
2	Unmounting SD Card	0.15 s	22	Delete old sources.list.d	0.63 s
3	Unmounting SD Card	4.81 s	23	Install monitoring software	41.99 s
4	Writing Image to SD Card	280.29 s	24	Install Nagios-plugins	78.18 s
5	Flush Cache	0.17 s	25	Copy the new ntp.conf file	1.64 s
6	Renew Dev	0.16 s	26	Copying autostart-file	1.44 s
7	Sleep 5 s	5.37 s	27	Copying rc.local	1.52 s
8	Mount Partition 2 of SD-Card	0.32 s	28	Install gdebi	2.54 s
9	Mount Partition 1 of SD-Card	0.17 s	29	Delete OpenJDK	5.34 s
10	Activate ssh	0.31 s	30	Install oracle Java8	2.55 s
11	Create new hostname	1.68 s	31	Install JavaFX	29.91 s
12	Delete old hostname	0.17 s	32	Update Password	1.14 s
13	Delete hosts file	0.16 s			
14	Update IP address	0.31 s			
15	Create .ssh directory	0.17 s			
16	Change rights on .ssh directory	0.16 s			
17	Copy new authorized_keys	0.54 s			
18	Copy id_rsa	0.28 s			
19	Deploy components	0.24 s			
20	Unmounting SD Card	0.18 s			

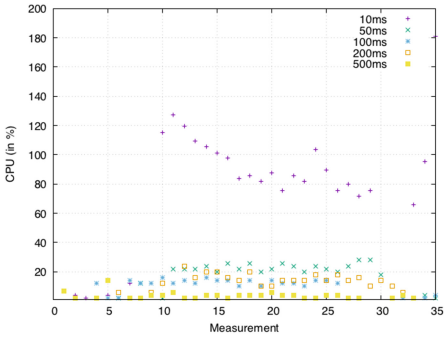


Fig. 7. CPU usage share of the SCADA master node under different loads.

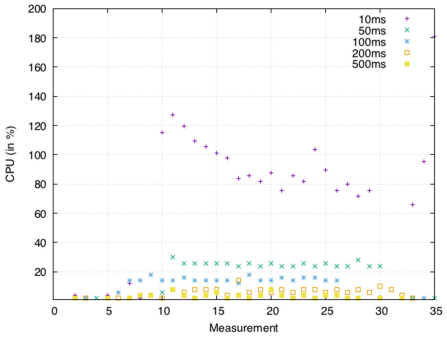


Fig. 8. CPU usage share of the modbus device under different loads.

15 min. Secondly, SCADA specific traffic was monitored. The wind turbine emulators send data using the Modbus protocol. Hence the number of Modbus specific packets is presented in Fig. 10. The Level 1 and Level 2 SCADA systems communicate using NGP, a NeoSCADA specific protocol using TCP/IP. Their

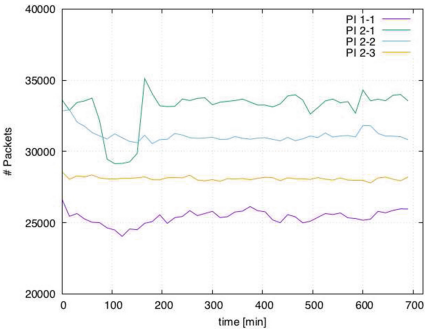


Fig. 9. The complete network traffic.

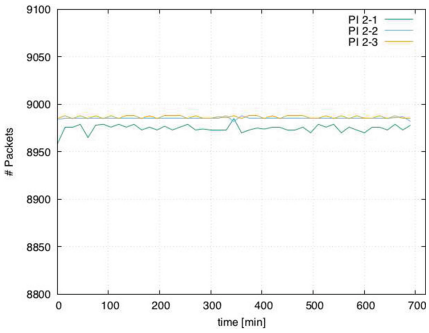


Fig. 10. Number of Modbus packets.

number of occurrence is presented in Fig. 11. It can be observed that the network traffic of the Level 1 SCADA node (label PI 1-1) consists only of NGP packets, as it does not communicate directly with a wind turbine. The Level 2 SCADA nodes (PI 2-1, PI 2-2 and PI 2-3) communicate both, via NGP with said Level 1 node and via Modbus with the connected wind turbines. As can bee seen in Figs. 9 and 11 the connection between PI 1-1 and PI 2-1 suffers around time 100 min, while the connection between PI 2-1 and its connected wind turbines is not impacted.

Every SCADA node requests new Modbus data in fixed time intervals. This explains the low variation in these measurements. The graphs furthermore show that Level 2 nodes have about 50 % of SCADA specific traffic. The remainder of the traffic can be attributed e.g. to TCP handshakes, responses, DNS requests, NTP and SSH traffic, necessary to organize the communication infrastructure.

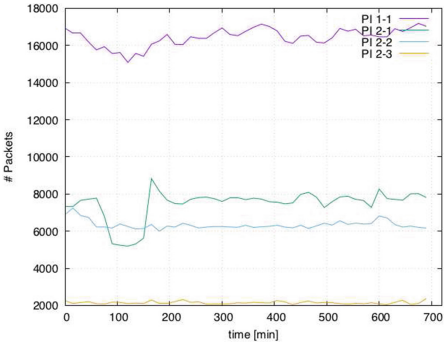


Fig. 11. Number of NGP packets.

7 Conclusions and Future Work

The implemented test bed mimicks a full stack real-world SCADA system used to control wind parks. The automation of installation, configuration and administration enables to research and educate in an environment that can readily be deployed. The test bed runs on a Raspberry Pi cluster and its emulation mode allows to replay real-world system data. Simulation mode implements adaptive behavior and allows to remotely control an RTU and its wind turbine. Every component is implemented in an extensible way, such that e.g. other protocols or SCADA nodes can easily be included.

Configuring and running the Eclipse NeoSCADA system has turned out to be a very complex task. The system is highly extensible and configurable, which

makes it difficult to reliably configure every node manually. Especially the automated generation of configuration packages, as well as the extension with a custom OSGI instance was helpful. For the automated configuration scripts have been created that contain all necessary specifications.

Future work will focus on extending the test beds components and implement a variety of communication protocols. The Level 1 SCADA system should be able to display aggregated wind park data. Hence, future work may focus on implementing replication techniques. Zero-downtime strategies, unit as well as integration tests need to be extended to increase the reliability of the test bed.

References

1. Davis, C.M., Tate, J.E., Okhravi, H., Grier, C., Overbye, T.J., Nicol, D.: SCADA cyber security testbed development. In: North American Power Symposium, pp. 483–488. IEEE (2006)
2. Banbura, M., Giannone, D., Modugno, M., Reichlin, L.: Now-casting and the real-time data flow. In: Elliott, G., Timmermann, A. (eds.) Handbook of Economic Forecasting, vol. 2, pp. 195–237. Elsevier, Amsterdam (2013)
3. Barnes, K., Johnson, B.: National SCADA test bed substation automation evaluation report. Technical report, Idaho National Laboratory (2009)
4. Beaver, J.M., Borges-Hink, R.C., Buckner, M.A.: An evaluation of machine learning methods to detect malicious SCADA communications. In: Machine Learning and Applications, vol. 2, pp. 54–59. IEEE (2013)
5. Cherdantseva, Y., Burnap, P., Blyth, A., Eden, P., Jones, K., Soulsby, H., Stoddart, K.: A review of cyber security risk assessment methods for SCADA systems. Comput. Secur. **56**, 1–27 (2016)
6. Chromik, J.J., Remke, A., Haverkort, B.R.: What's under the hood? improving scada security with process awareness. In: Cyber-Physical Security and Resilience in Smart Grids, pp. 1–6. IEEE (2016)
7. Cintuglu, M.H., Mohammed, O.A., Akkaya, K., Uluagac, A.S.: A survey on smart grid cyber-physical system testbeds. IEEE Commun. Surv. Tutor. **19**(1), 446–464 (2017)
8. Hunkeler, U., Truong, H.L., Stanford-Clark, A.: MQTT-S-A publish/subscribe protocol for wireless sensor networks. In: Communication Systems Software and Middleware and Workshops, pp. 791–798. IEEE (2008)
9. Igure, V.M., Laughter, S.A., Williams, R.D.: Security issues in SCADA networks. Comput. Secur. **25**(7), 498–506 (2006)
10. Maglaras, L.A., Jiang, J.: Intrusion detection in SCADA systems using machine learning techniques. In: Science and Information Conference, pp. 626–631. IEEE (2014)
11. Medina, V., Gómez, I., Oviedo, D., Dorronzoro, E., Martin, S., Benjumea, J., Sanchez, G.: International Symposium on Industrial Electronics, pp. 420–425. IEEE (2009)
12. Queiroz, C., Mahmood, A., Hu, J., Tari, Z., Yu, X.: Building a SCADA security testbed. In: Network and System Security, pp. 357–364. IEEE (2009)
13. Schütte, S., Scherfke, S., Tröschel, M.: Mosaik: a framework for modular simulation of active components in Smart Grids. In: Smart Grid Modeling and Simulation, pp. 55–60. IEEE (2011)

14. Singh, M., Muljadi, E., Jonkman, J., Gevorgian, V., Girsang, I., Dhupia, J.: Simulation for wind turbine generators-with FAST and MATLAB-simulink modules. Technical report, National Renewable Energy Laboratory (2014)
15. Swales, A., et al.: Open Modbus/TCP Specification, vol. 29. Schneider Electric (1999)

Evaluation of Single-Hop Beaconing with Congestion Control in IEEE WAVE and ETSI ITS-G5

Thomas Deinlein(✉), Reinhard German, and Anatoli Djanatliev

Chair of Computer Networks and Communication Systems, Friedrich-Alexander University Erlangen-Nürnberg, Erlangen, Germany
{thomas.deinlein,anatoli.djanatliev}@fau.de,
reinhard.german@informatik.uni-erlangen.de

Abstract. Car2X communication based on WLAN 802.11p is described in the communication protocols IEEE WAVE for the USA and ETSI ITS-G5 for Europe. In this paper we looked into the single-hop beaconing mechanisms and congestion controls of both standards. Each standard includes an own definition of beaconing and congestion control. By implementing both parts of the standards within the OMNeT++ framework VEINS different single-hop scenarios were looked into and an achievement assessment was provided concerning the congestion control mechanisms and the beaconing.

Keywords: Car2X · IEEE WLAN 802.11p · IEEE WAVE
ETSI ITS-G5 · Beaconing · DCC · BSM · CAM · VEINS

1 Introduction

The standards IEEE WAVE and ETSI ITS-G5 both base on the standard available for Car2X communication IEEE WLAN 802.11p [5]. In this standard the lower network layers (PHY and MAC) are described, which is the reason why IEEE WAVE and ETSI ITS-G5 have common characteristics on the lower layers. In the following we describe the individual characteristics of the generation process of beacons and the congestion control of both IEEE WAVE and ETSI ITS-G5.

IEEE WAVE. On the part of IEEE WAVE the generation of single-hop beacons occurs in the form of Basic Safety Messages (BSM, [6]) on the application layer. The congestion control is integrated in the generation process [9]. In Fig. 1 it is shown how the algorithm works in detail.

All parameters to be calculated and the respective intervals are indicated in the upper area of Fig. 1. There are three different intervals used for the calculation (100 ms, 1 s and 5 s). In the lower part of Fig. 1 the algorithm for BSM generation is shown. The dotted arrows originating from few parameters should make clear that only few of all calculated parameters are directly involved in the actual BSM generation algorithm.

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 273–282, 2018.
https://doi.org/10.1007/978-3-319-74947-1_19

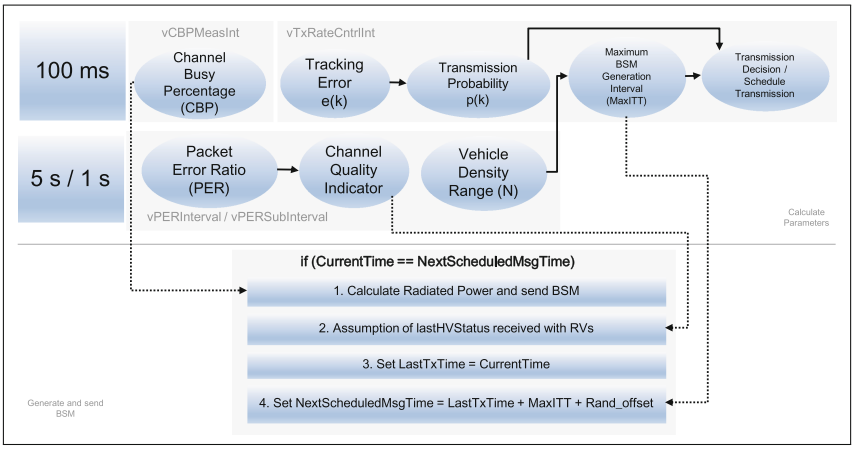


Fig. 1. Generation of BSM in IEEE WAVE [9]

ETSI ITS-G5. On the part of ETSI ITS-G5 the generation of single-hop beacons occurs the in form of so-called Cooperative Awareness Messages (CAM, [2]) on the facility layer which is located between the application layer and the transport layer. A Decentralized Congestion Control (DCC) is in this case intended on the MAC layer [3].

The DCC on MAC layer is a state machine which configures different parameters depending on the respective state. The state will change if the measured minimum or maximum channel load of the last 1 s or 5 s exceeds 15% and 40%, respectively. The current channel load is measured every 100 ms. In Fig. 2 the states are indicated and the standard parameters are shown in Table 1. The parameter *T_GenCam_DCC* is important for the generation process of CAMs. It has to be provided by the DCC and needs to be at least 100 ms. The value for this parameter is the current packet rate (see Table 1), which depends on the current state of the DCC. For the CAM generation algorithm it is the minimum time between generating two CAMs on the facility layer. The packet rate handles the packet flow on MAC layer, too. It is the minimum time between sending two packets on the MAC layer.

Table 1. Standard parameters of the DCC [3]

EDCA AC	State					
	Relaxed	Active				Restrictive
		AC_VO	AC_VI	AC_BE	AC_BK	
TxPower	33 dBm	ref	25 dBm	20 dBm	15 dBm	−10 dBm
Packet rate	0.04 s	ref	ref	ref	ref	1 s
Datarate	3 Mbit/s	ref	ref	ref	ref	12 Mbit/s
Sensitivity	−95 dBm	ref	ref	ref	ref	−65 dBm

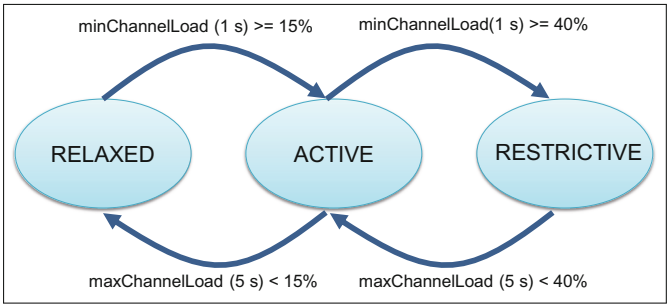


Fig. 2. DCC statemachine [3]

The generation of one CAM is a highly dynamic process and is carried out depending on the change of the current vehicle state in comparison to the state values contained in the last sent CAM (see [2], Chap. 6.1.3). The check occurs every 100 ms. Shorter time periods can also be set for this. Besides, it is checked whether at least the time *T_GenCam_DCC* (mentioned above) has elapsed since the last CAM generation, and:

- the difference of the current speed and the speed which is included in the last sent CAM is greater than 0.5 m/s, or
- the distance of the current position and the position which is included in the last sent CAM is greater than 4 m, or
- the amount of the difference of the current heading and the heading which is included in the last sent CAM is greater than 4°.

A CAM will be sent if one of the state changes above occurred and the time of at least *T_GenCam_DCC* has exceeded since the last generation of a CAM. *T_GenCam* represents the current CAM generation interval and is used to reset the generation interval to 1 s if no dynamic state changes occur. *T_GenCam* is set to the time which is passed by since the last generation of a CAM. If two further CAMs are sent within *T_GenCam* without a change of the state parameters above, *T_GenCam* will be set to 1 s.

Beside the standard parameters, two further variations have been considered for DCC in this study. In [4] the DCC profiles are described, which shorten the packet rate to 0.095 s in relaxed state, 0.190 s in active state and 0.250 s in restrictive state.

In [1] another variation is described how values of the packet rate can be dynamically determined. The calculation algorithm is based on the so-called LIMERIC algorithm [15]. It is a dynamic calculation of another individual parameter every 100 ms, which represents the interval that has to be expired before the next CAM can be generated.

Because of the great differences between the standards IEEE WAVE and ETSI ITS-G5 it was interesting to examine their performance in real traffic scenarios.

2 Related Work

There are several comprehensive studies about the European standard ETSI ITS-G5. In [12] the beaconing was compared between ETSI ITS-G5 and IEEE WAVE in 2013. However, the generation rate had a constant rate of 10 Hz in this study. This appears only in particular situations. In that study only the standard parameters of the DCC were considered. In spite of an oscillatory behaviour of the channel load on the part of ETSI ITS-G5 caused by the state change of the DCC it was found out that this approach leads to a better performance than IEEE WAVE.

The oscillation of the channel load in combination with the generation of CAMs was examined in [17]. It was ascertained that the combination of the generation rate of a CAM with other parameters like transmission power influences stability of the DCC.

The reactive behaviour dependent on the state of the DCC was compared in [11]. Oscillations of the DCC and unstable behaviour could be observed, too.

Other investigations in [10,16] came to similar results. In addition, [20] found out that the DCC can lead to instability and a low number of transferred CAMs.

With IEEE WAVE the need of a congestion control mechanism for generating BSMs was ascertained in [14] already in 2011. With introduction of [9] this was published in 2016.

In contrast to the mentioned papers before this paper looks into the current definition of both standards. In particular the IEEE WAVE mechanism described in [9] was not included in former studies.

3 Implementation in VEINS

The algorithms of both standards IEEE WAVE and ETSI ITS-G5 were integrated within the simulation framework VEINS. It is an often used open source framework for Car2X scenarios (see [18,19]). VEINS is a framework which bidirectionally couples the well-known network simulation tool OMNeT++ [8] and the traffic simulator SUMO [7]. SUMO uses open street map files to run traffic scenarios. With the help of this tool chain it is possible to run different traffic scenarios with our implemented code of both standards. In this study we used Version 4.4 of VEINS.

VEINS includes the protocol stack defined in IEEE WLAN 802.11p [5]. By default there are the physical, the MAC and the application layer implemented. For this study we changed the VEINS code of these layers. We did not implement further network layers such as the Geonetworking protocol. Our focus was the process of generating the beacons on application layer (both standards) and the behaviour of the DCC on the MAC layer on part of ETSI ITS-G5.

We have chosen three different scenarios to compare both standards. The scenarios are:

– Motorway junction: Typical for this scenario are high speeds (100 km/h up to 200 km/h) and rare changes of heading. The plane size is 2000 m × 2000 m.

- Urban: Low speed (5 km/h up to 50 km/h), stops and frequent heading changes characterise this scenario. The plane size is 800 m × 800 m.
- Traffic jam: In this scenario there is no movement of the cars and all cars are in radio range of each other. The plane size is 200 m × 200 m.

Table 2 shows an overview of all parameters which are set for this study.

Table 2. Simulation parameters

Parameter	Value
Simulation time	6 parallel runs for 10 s each, 1 run for 200 s and 500 s
Scenarios	Motorway junction, urban, traffic jam
Number of cars	100, 200
Loss modell	Simple path loss model (k = 2)
EDCA-Priority	AC_VO (BSM), AC_BE (CAM)
Beacon-Size	300 Bytes, 600 Bytes
Frequency	5.86 GHz (BSM), 5.9 GHz (CAM)

To evaluate the performance of both standards we determined several statistic values, which were saved in every run of a simulation scenario:

- Generation rate: This means the time between two generated beacons.
- Number of sent beacons: The number of all sent beacons in a scenario saved during simulation time.
- Number of received beacons: The number of all received beacons in a scenario saved during simulation time.
- Channel load: In both standards the channel load is calculated every 100 ms. It is the result of the time at which the channel was busy during the last 100 ms divided by 100 ms.
- Lost packets: The number of lost packets in a scenario saved during simulation time.

4 Comparison and Outcomes

After the implementation in VEINS we were able to compare both standards by the theoretical facts. The focus was on the parameters which are set by the DCC. In Table 3 the compared parameters are shown. Two facts are remarkable. First, the generation rate of beacons differs enormously. In IEEE WAVE the rate is up to 600 ms and in ETSI ITS-G5 it is up to 100 ms. Second, the default generation rate is contrary. In both standards the minimum rate is 100 ms. IEEE WAVE sets this value for default, in ETSI ITS-G5 the rate is 1000 ms by default and it will change only if there are dynamic state changes as described in Sect. 1.

Table 3. General comparison [3,9]

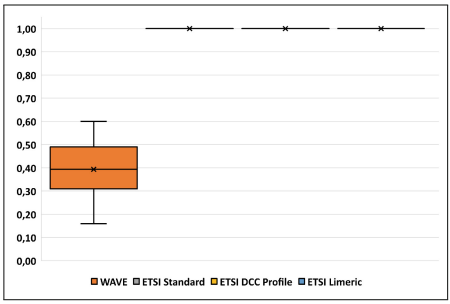
Parameter	IEEE WAVE	ETSI ITS-G5
TxPower	10 dBm up to 20 dBm	−10 dBm, 20 dBm or 23 dBm
Datarate	6 Mbit/s	3 Mbit/s or 12 Mbit/s
Sensitivity	−92 dBm	−65 dBm, −85 dBm or −95 dBm
Generation rate	100 ms up to 600 ms	100 ms up to 1000 ms
Default generation rate	100 ms	1000 ms

After running all three scenarios with different parameter values we came to the conclusion that the statistics of all three scenarios point into the same direction. Therefore, we chose values for Fig. 3 such that the differences between IEEE WAVE and ETIS ITS-G5 were most significant. The traffic jam scenario shows the greatest differences. Figure 3a and b show the generation rate and the channel load, respectively. Although all cars don't move, in IEEE WAVE the generation rate differs between about 100 ms and 600 ms. In ETSI ITS-G5 there is no change of the rate. This leads to more sent and received beacons in IEEE WAVE (see Fig. 3c and d). Especially the channel load shows that the congestion control in IEEE WAVE doesn't work well because the channel load has a constant value of about 75% during the whole simulation time of 200 s. In ETSI ITS-G5 the channel load is about 10% in all three variations. The differences between the three ETSI ITS-G5 variants are insignificant and will not be further discussed in this paper.

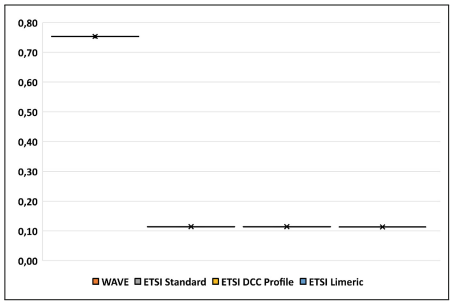
The behaviour of the IEEE WAVE mechanism was very surprising. Even when high channel load is measured, the mechanism doesn't change the parameters in such a way to reduce the channel load effectively. We looked into that behaviour deeper and found out that it depends on the parameter vPERRange, which is set to 100 m by default (see [9], Table 21). vPERRange is involved in the calculation of the parameter vVehicleDensityInRange, which is necessary for calculating the maximum generation rate. If there are at least 150 cars within this range, the generation rate will be set to the maximum value of 600 ms, which would lead to less sent beacons and therefore to smaller channel congestion. The problem is that one car can receive beacons from cars which are in a wider range. These cars, which are out of the vPERRange, aren't included in the calculation of the maximum generation rate. The result is a shorter generation rate although there are more cars in the real radio range from which beacons were received.

We compared the behaviour of the channel load if vPERRange is 100 m and 200 m in the scenario traffic jam. This scenario has a plane size of 200 m × 200 m. In Fig. 4 the results are shown. We also compared if there are differences in the behaviour when the beacon size is 300 bytes and 600 bytes, respectively. The value of 100 m for vPERRange leads to a shorter generation rate and thus to a higher channel load, even if the cars don't move all the time. This effect depends on the calculation of the maximum generation rate as mentioned above. In the range of 100 m less than 150 cars have been measured, although there are 200

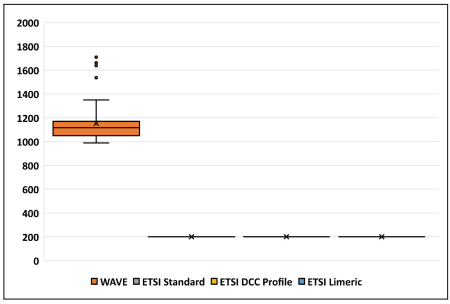
cars receiving beacons from each other. If the vPERRange is set to 200 m, the channel load decreases because of the higher number of counted cars within range and so the generation rate increases. That leads to less sent beacons and thus to a smaller channel load.



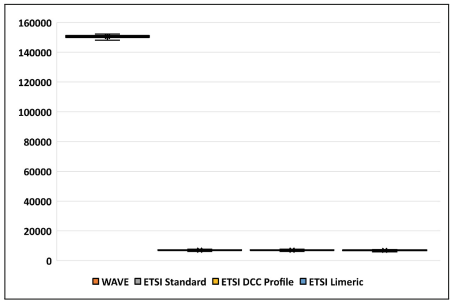
(a) Generation rate in seconds



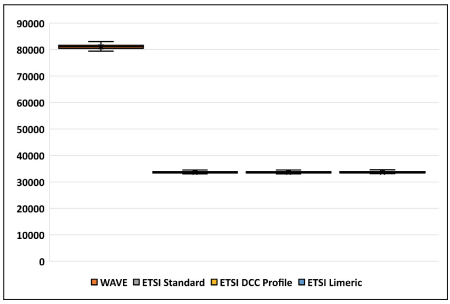
(b) Channel load in %



(c) Sent Beacons

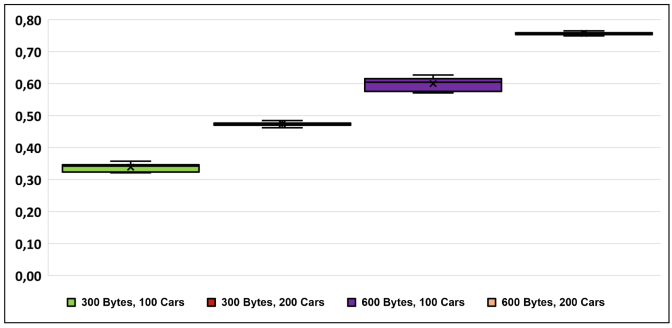


(d) Received Beacons

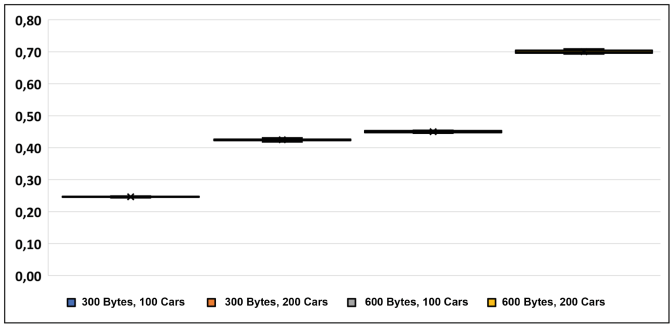


(e) Packet collisions

Fig. 3. Simulation results of the *traffic jam* scenario with 600 bytes beacon size and a simulation time of 200 s and 200 cars



(a) *vPERRange* is 100m



(b) *vPERRange* is 200m

Fig. 4. Impact of changing *vPERRange* on the channel load in the *traffic jam* scenario

5 Conclusion and Future Work

Within the scope of this study we compared the single-hop beaconing in combination with congestion control of the standards IEEE WAVE and ETSI ITS-G5. By implementing the algorithms within the VEINS framework we compared the performance based on three traffic scenarios.

IEEE WAVE shows similar results in all three scenarios. Due to a short generation rate (almost exclusively 100 ms) there will be sent many more beacons than with ETSI ITS-G5. This leads to many more received beacons. However, this mechanism doesn't control the channel load effectively and tends to congest the radio channel. Furthermore, we found out that the generation rate, which is freshly calculated after sending a beacon, only depends on the number of cars counted in the range of 100 m, although the cars also receive beacons from cars which are further away. Setting the value of the parameter *vPERRange* to 200 m showed a reduction of the channel load.

On the part of ETSI ITS-G5 we looked into the generation of CAMs in combination with the DCC. We considered three variations of the DCC parameters: standard, DCC profile and a dynamic approach with LIMERIC. All three variations behave contrarily to IEEE WAVE. Because the generation of CAMs is

depending on the current driving state, ETSI ITS-G5 sends less and receives less beacons. This leads to much less channel load and less packet loss. The differences between all three ETSI ITS-G5 variations are insignificant.

For future studies it may be beneficial to look into mobile communications within the Car2X-context and VEINS, respectively. In [13] an adaption of VEINS with LTE has already been developed. The forthcoming 5th generation of mobile communication (5G) is going to fulfill lower latency and it is supposed to support many more connections between devices as with LTE. Extending VEINS by 5G-Features is necessary to compare the performance with IEEE 802.11p WLAN standards, IEEE WAVE and ETSI ITS-G5, which were part of this study, and LTE, respectively.

References

1. ETSI draft TS 102 687 v1.1.2: Intelligent transport systems (ITS); decentralized congestion control mechanisms for intelligent transport systems operating in the 5 GHz range; access layer part, January 2014
2. ETSI EN 302 637-2 v1.3.2: Intelligent transport systems (ITS); vehicular communications; basic set of applications; part 2: specification of cooperative awareness basic service.www.etsi.org/standards-search#search=ETSIEN302637-2V1.3.2
3. ETSI TS 102 687 v1.1.1: Intelligent transport systems (ITS); decentralized congestion control mechanisms for intelligent transport systems operating in the 5 GHz range; access layer part, July 2011. http://www.etsi.org/deliver/etsi_ts/102600_102699/102687/01.01.01_60/ts_102687v010101p.pdf
4. ETSI TS 102 724: Intelligent transport systems (ITS); harmonized channel specifications for intelligent transport systems operating in the 5 GHz frequency band. http://www.etsi.org/deliver/etsi_ts/102700_102799/102724/01.01.01_60/ts_102724v010101p.pdf
5. IEEE 802.11p-2010: IEEE standard for information technology- local and metropolitan area networks - specific requirements - part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 6: wireless access in vehicular environments. http://standards.ieee.org/findstds/standard/802.11p-2010.html
6. SAE j2735_201603: Dedicated short range communications (DSRC) message set dictionary. http://standards.sae.org/j2735_201603/
7. SUMO - simulation of urban mobility. http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/
8. What is OMNeT++? https://omnetpp.org/intro
9. SAE j2945/1_201603: On-board system requirements for V2V safety communications, 30 March 2013. http://standards.sae.org/j2945/1_201603/
10. Autolitano, A., Campolo, C., Molinaro, A., Scopigno, R.M., Vesco, A.: An insight into decentralized congestion control techniques for vanets from ETSI TS 102 687 v1.1.1. In: 2013 IFIP Wireless Days (WD), pp. 1–6 (2013)
11. Bansal, G., Cheng, B., Rostami, A., Sjoberg, K., Kenney, J.B., Gruteser, M.: Comparing LIMERIC and DCC approaches for vanet channel congestion control. In: 2014 IEEE 6th International Symposium on Wireless Vehicular Communications (WiVeC 2014), pp. 1–7 (2014)

12. Eckhoff, D., Sofra, N., German, R.: A performance study of cooperative awareness in ETSI ITS G5 and IEEE WAVE. In: 10th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2013), pp. 196–200. IEEE, Banff (2013)
13. Hagenauer, F., Dressler, F., Sommer, C.: Poster: a simulator for heterogeneous vehicular networks. In: 2014 IEEE Vehicular Networking Conference (VNC), pp. 185–186. IEEE (2014)
14. Kenney, J.B.: Dedicated short-range communications (DSRC) standards in the United States. Proc. IEEE **99**(7), 1162–1182 (2011)
15. Kenney, J.B., Bansal, G., Rohrs, C.E.: LIMERIC: a linear message rate control algorithm for vehicular DSRC systems. In: Proceedings of the Eighth ACM International Workshop on Vehicular Inter-networking, VANET 2011, pp. 21–30. ACM, New York (2011). http://doi.acm.org/10.1145/2030698.2030702
16. Lorenzen, T.: Performance analysis of the functional interaction of awareness control and DCC in vanets. In: 2016 IEEE 84th Vehicular Technology Conference (VTC-Fall), pp. 1–7 (2016)
17. Marzouk, F., Zagrouba, R., Laouiti, A., Muhlethaler, P.: An empirical study of unfairness and oscillation in ETSI DCC. In: International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks, PEMWN 2015, Hammamet, Tunisia (2015). https://hal.archives-ouvertes.fr/hal-01211466
18. Sommer, C.: Veins - the open source vehicular network simulation framework. http://veins.car2x.org/
19. Sommer, C., Dressler, F.: Vehicular Networking. Cambridge University Press, Cambridge (2015)
20. Yang, S., Kim, H.: Configuring ETSI DCC parameters for better performance and stability. Electron. Lett. **52**(8), 667–669 (2016)

Practical QoE Evaluation of Adaptive Video Streaming

Sebastian Surminski(✉), Christian Moldovan, and Tobias Hoßfeld

Chair of Modeling of Adaptive Systems,
University of Duisburg-Essen, Essen, Germany
sebastian.surminski@uni-due.de

Abstract. Video streaming is an increasingly popular service on the Internet. In HTTP adaptive video streaming (HAS), the video is played while being downloaded, and the quality is selected according to the available bandwidth. Due to this, variations in the transmission affect the playback. The quality of the playback can be rated by technical parameters, which can be grouped by the term 'Quality of Service' (QoS), like the video quality, the number and duration of stallings or the time until the video starts playing. These metrics differently influence the user experience.

Up to now, no widely accepted model for the Quality of Experience (QoE) for HAS exists. Therefore, we use two conceptually different models and investigate their impact on the resulting QoE. To do so, we use a typical video player, namely the Shaka Player, that can be embedded into websites, and change its buffer configuration. The observed data is then used to evaluate the quality of experience (QoE), combining it into a single 'Mean opinion score' (MOS). It can be shown, that, with limitations, these methods can be suited for QoE evaluation.

Keywords: Adaptive video streaming · Quality of Experience

1 Introduction

Video streaming has gained a lot of popularity in the Internet during the last two decades. Not only has the consumed video traffic grown on the Internet, but also have the user expectations for high video quality. In order to be able to stream videos in high resolutions even in overloaded mobile networks, video service providers have started to implement methods, that adapt the video bit rate to the available goodput. In HTTP adaptive streaming (HAS), the video player downloads video segments into his buffer, until enough content is available. If the amount of content stored in the buffer is decreasing and drops below a certain threshold, the video is requested in a lower bit rate by the player [16]. If the amount of content in the buffer reaches a high level, the video may be requested in higher quality again. Since many videos are abandoned by users during replay [3], the buffer capacity is limited to avoid downloading too much video content

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 283–292, 2018.
https://doi.org/10.1007/978-3-319-74947-1_20

that is discarded in an abandonment event. This way video service providers are able to avoid wasting resources, i.e. avoid transmission of unwatched video content. This paper is an extension of our previous experimental study [15] in which we investigated the influence of different buffer configurations on the playback behavior of two different HAS players in real commuting scenarios. This led to important results regarding the impact on the duration and number of stalling events, as well as wasted traffic. In this paper, we want to go one step beyond our previous analysis of QoS and also investigate the Quality of Experience (QoE). While the QoS includes objective metrics, the QoE is defined as the subjective 'degree of delight or annoyance' experienced by a user [1]. In video streaming, the impact of a quality degradation on the QoE if usually determined with the help of user studies from which QoE models can be deducted. We investigate synthetic QoE models were several factors are combined into an additive or multiplicative QoE model [8]. We developed a testbed which allows us to run HAS video players, which can be embedded in websites, a popular example is the YouTube player. The bandwidth available for video transmission can be limited according to real bandwidth scenarios. Using this setup, we tested the Shaka player with different buffer configurations to investigate in how far typical, simple QoE metrics can be used to evaluate video playback. To do so, we compared the QoE results with a detailed analysis of the playback of the video.

The remainder of this paper is structured as follows. In Sect. 2 background and related work on HTTP adaptive video streaming is discussed. Section 3 presents the QoE models that are used in the experiments investigated in this paper. Section 4 outlines the methodology and discusses the results of our experiments. In Sect. 5, the paper is concluded and an outlook on future work is provided.

2 Background and Related Work

In contrary to traditional video transmission, apart of the video encoding, also the adaptation strategy, has a massive influence on the quality of the video, that the user receives. The adaptation strategy decides, which parts of the video are downloaded in which quality at which time. Especially in mobile scenarios, where the bandwidth changes over time, the adaptation logic therefore has a large influence on the video quality [11].

When designing an adaptation strategy, a compromise between different objectives has to be found. For example, a short initial waiting time, this means the time until the video actually starts playing, can result in stalling events, because the buffer is insufficiently filled, and the player runs out of video to be played. At the same time, the played quality should be maximized, which then might lead to frequent quality switches, because of variances in the bandwidth.

Traditionally, technical parameters, summarized as 'Quality of Service' (QoS) are used to monitor and describe service quality. The focus of this evaluation is the user's experience. User studies lead to the development of models for

the Quality of Experience (QoE), where users rate typically impairments of the service by their perceived level of annoyance. A broad overview about QoE modeling in HAS is given in [13]. The QoE models that we used in this work are explained in detail in Sect. 3.

Realistic QoE models allow an automated evaluation, which then can be used in practice to improve service levels according to the perception of the users. Especially the upcoming of so-called cloud services, where more and more functionalities and applications are moved into data centers, shows the possibilities for QoE-management [5]. An understanding of QoE does not only allow to improve the perceived service quality, but also saving money by adapting the service level to the actual users' needs, e.g. response or loading times.

3 QoE Model

The QoE does not describe technical properties of a service, but solely the delight or annoyance experienced by a user when using a service. However, technical aspects of a service may have an impact on the user's QoE. QoE models allow a holistic view on services, and help to understand the users' perception. This user-centric approach is adopted in order to allow optimization of the aspects that really count for the users instead just of technical QoS parameters, for example in cloud computing. This is especially challenging because with QoE the overall user perception is the key metric for managing collaboration between different providers [5].

There are different metrics how to measure and quantify the user experience which are often applied in user studies. A popular method to express the QoE is a Mean Opinion Score (MOS) with ratings from 1 ('Bad') to 5 ('Excellent'). In detail, these numbers are related as follows: 5 – 'Excellent', 4 – 'Good', '3' – Fair, '2' – Poor, and '1' – Bad [10]. Another method is to let users rate the 'acceptability', this means in the case of video streaming, if the video quality is acceptable or if they do not accept it [14]. There are many more interesting QoE metrics beyond the MOS like QoE quantiles or ratio of users accepting good or better quality, see [6] for a discussion on relevant QoE metrics. However, in this paper we are using the MOS value, as we can rely on subjective studies reporting the MOS.

This work's goal is to determine QoS parameters on application layer via experiments, while varying the QoS on network layer. In contrast to analytic models and simulations, the experimental approach returns the most sophisticated results, but it is also very time consuming. After that, the obtained QoS parameters have to be evaluated with the use of user-study-based models that map them to QoE values. Generally, it is difficult to correctly interpret and weigh these QoS values accordingly, especially considering the users perception. Therefore, we focus on the MOS, as it allows to directly quantify the QoE.

For the determination of the QoE, we consider the following two key performance indicators, as these are the main influence factors [5]:

– *stalling events*, as stalling has a high impact on the overall QoE [5] and
– *initial waiting time*, because this directly correlates with the buffer size, which is in the focus of this work. Depending on the usage scenario, e.g. when the user is browsing for a video, this parameter gets a higher importance for the user [2].

3.1 Basic QoE Models for Adaptive Video Streaming

We use the following two functions, which return the MOS depending on stalling events and initial waiting time. The first function returns the QoE depending on the number and duration of stalling events:

$$Q_{\text{Stalling}}(N, L) = a \cdot e^{-(b \cdot L + c) \cdot N} + d \tag{1}$$

where L is the average length of the stalling events in seconds, and N is the frequency of stalling events normalized according to [7]. For evaluation, we used the parameters $a = 3.5$, $b = 0.15$, $c = 0.19$ and $d = 1.5$ as proposed in [4].

The second function, which determines the MOS solely depending on the initial waiting time:

$$Q_{\text{Waiting}}(D) = -a \log_{10}(D + b) + c \tag{2}$$

where D is the initial waiting time in seconds and $a = 0.963, b = 5.381, c = 5$ as suggested by [7]. This leads to a MOS of 4.3 in case of an initial waiting of one seconds, 4.0 with two seconds, and 3.6 in case of five seconds, respectively.

3.2 Combining QoE Models

It is still under discussion how multiple QoE values can be combined. The two most simple approaches are the additive and the multiplicative QoE model [8]. Both models are conceptually different, the multiplicative model is suited, when there is a dominating factor, while the additive model averages the factors. For comparison, we will apply both and compare the results.

For general purpose weighted additive models have been suggested in [8]. An additive model generally sums up the input values. The result can be adapted with weighting values $w_i \geq 0$ and $\sum_i w_i = 1$. A general formula therefore is given as

$$Q_{\text{add}} = \sum_{i=1}^{n} w_i Q_i(x_i)$$

In our case this leads to the following function:

$$Q_{\text{add}}(D, L, N) = 0.5 \cdot Q_{\text{Stalling}}(N, L) + 0.5 \cdot Q_{\text{Waiting}}(D) \tag{3}$$

The multiplicative model for $Q_i \in [0, 1]$ is given as:

$$Q_{\text{mul}} = \prod_{i=1}^{n} Q_i(x_i)$$

Since $Q_{\text{Stalling}}(L, N)$ and $Q_{\text{Waiting}}(T_0)$ are MOS values on a $[1, 5]$ scale, we use the equation

$$Q_{\text{mul}}(T_0, L, N) = \frac{(Q_{\text{Stalling}}(N, L) - 1) \cdot (Q_{\text{Waiting}}(D) - 1)}{4} + 1 \tag{4}$$

for the multiplicative model, where both MOS values Q_{Stalling} and Q_{Waiting} are previously reduced to values ≤ 1 and the result then transformed to comparable MOS values $\in [1, 5]$. Although both models allow to introduce weighting factors for the components, we treat them even, as there are no general concrete recommendations for them. Please note that we are not evaluating video quality and switches. But the approach can be extended to cover these aspects.

4 Results

As stated before, this work is an extension of a previous paper [15], where the bandwidth wastage was in focus. There, a detailed analysis of the players' playback behavior, e.g. quality switches (adaptations), stalling events and their durations and initial waiting times can be found. The focus of this work is on the evaluation of the QoE, that was introduced in Sect. 3. Before we present the results, we outline the methodology.

4.1 Methodology

In [15], we developed a setup that allows us to test HTML video players under predefined bandwidth conditions, while monitoring their behavior. The bandwidth is controlled by bandwidth traces, which reflect real-world scenarios (Bus, car, ferry, metro, train, and tram). These were provided by [12] using a notebook and a 3G modem for measuring download speed, and a GPS module for localization. We used a test video with a length of about 10 min. In every bandwidth scenario, the video was repeated between 19 and 133 times. Using the players' API, we collected the following information, which then was used to determine the quality of the playback of the video.

- Initial waiting time
- Number and duration of stalling events
- Resolution of the video and adaptation events
- Buffer level

From these Quality of Service (QoS) parameters, we calculated the MOS values using the models presented in Sect. 3, considering the initial waiting time and the number and duration of the stalling events. The results of the experiments are evaluated in the following, thereby it is distinguished between the different bandwidth scenarios and buffer configurations. A buffer configuration determines the buffer size and the rebuffering goal. The buffer size is the maximum length of the video, that can be stored in the buffer. The rebuffering goal defines the minimum buffer level, which needs to be reached, before the player starts playing

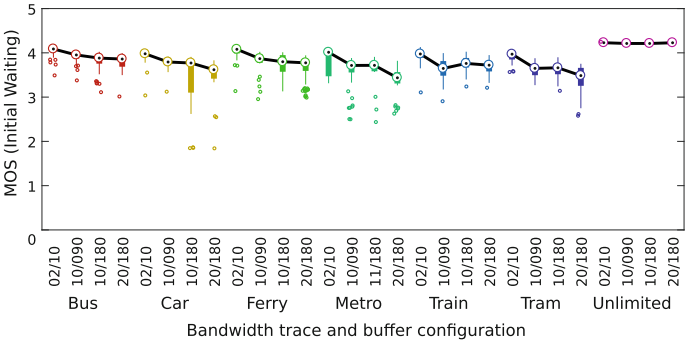


Fig. 1. Shaka player: QoE resulting from initial waiting time.

the video. The bandwidth scenario identifies the bandwidth trace, that was used to limit the bandwidth between video player and -source. As all experiments are run in real-time, we considered four different configurations:

- Buffer size 10 s, rebuffering goal of 2 s (Shaka players' default configuration)
- Buffer size 90 s, rebuffering goal of 10 s
- Buffer size 180 s, rebuffering goal of 10 s
- Buffer size 180 s, rebuffering goal of 20 s

This way, we can compare the effect of different buffer sizes and rebuffering goals.

The boxplot in Fig. 1 shows the QoE resulting from the initial waiting time. On the x-axis, the different buffer configurations can be found, grouped by the bandwidth scenario. The first number denotes the rebuffering goal, the second, larger number is the buffer size. The QoE, represented by the MOS, can be found on the y-axis. As it can be expected, a smaller rebuffering goal leads to a shorter initial waiting time, resulting in a better QoE.

But this lower buffer level, when the player starts to play makes the playback more volatile regarding bandwidth changes, which can result in stalling, when the buffer runs empty and the player has to stop. In Fig. 2 the QoE resulting from stalling is plotted. Therefore, the number and their average duration during one playback of the video is evaluated. Like in the figure before, on the horizontal axis the different bandwidth scenarios and buffer configurations, while on the vertical axis the MOS are given. It can clearly be seen that the smallest buffer size has the worst QoE, the larger buffer leads to a significantly better rating. For comparison, the case with unlimited bandwidth has no stalls and therefore gets a perfect rating with a MOS of 5.

For an overall evaluation, where both the initial waiting and the stalling are considered, these two models have to be combined. To do so, there are different methods, which were presented in Sect. 3.2, namely the additive and the multiplicative model. In Fig. 3 the MOS of the different scenarios and buffer configurations ratings are combined using the additive model. The multiplicative model is used in Fig. 4. For simplicity, we weigh initial waiting and stalling evenly.

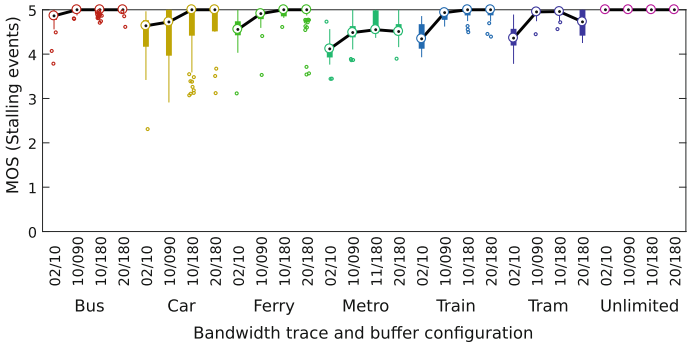


Fig. 2. Shaka player: QoE resulting from number of stalling events and their average duration.

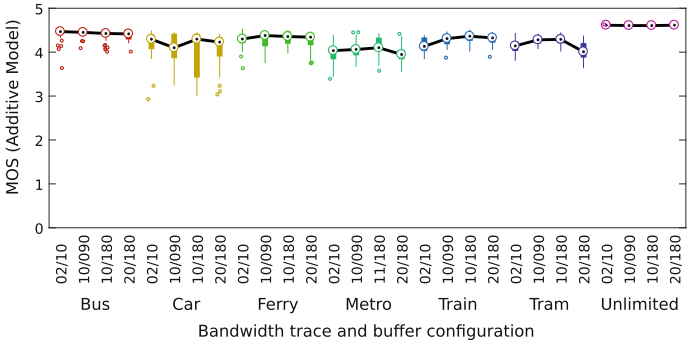


Fig. 3. Shaka player: combined QoE using the additive model.

In practice, this has to be adapted to the use case. If users tend to browse more, skip large parts in the videos and abandon the playback, the initial waiting time is more important, while when users select a video and watch it for a longer time, the effect of stalling becomes more important.

In most bandwidth scenarios, the buffer configuration with a 10 s rebuffering goal and a buffer size of 180 s has the best MOS rating. Only in the 'bus' scenario, which has the highest average bandwidth, the configuration with the smallest buffer size (10 s) has a slightly better rating, because in this specific scenario the initial waiting time lowers the high rating of the stalling events, because in this scenario only little stalling occurs.

This evaluation using the QoE also matches the results of previous evaluations in [15], where the playback statistics were manually interpreted. This shows, that the QoE model can be used in practice for evaluation of adaptive video streaming. The automatic interpretation and reduction of complex statistics to a single MOS value allows it to include QoE evaluation and monitoring into practical scenarios.

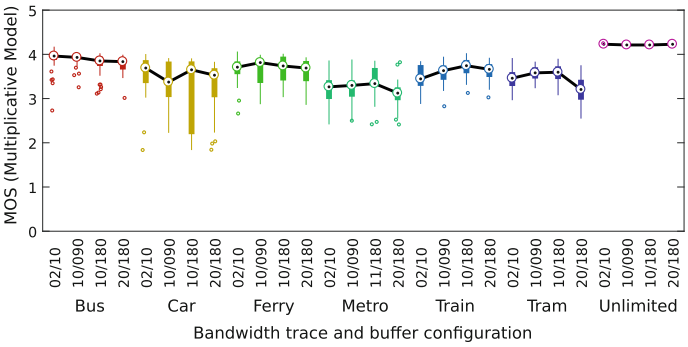


Fig. 4. Shaka player: combined QoE using the multiplicative model.

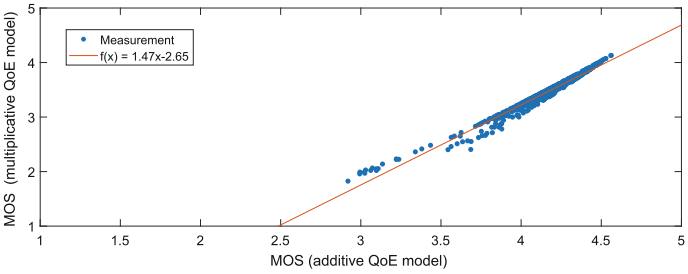


Fig. 5. Shaka player: This figure shows the QoE for the additive (x-axis) and multiplicative model (y-axis). In this evaluation, there is a linear relationship between both models (mean squared error: 0.0043741). In general, the rating of the multiplicative model is lower than the additive model.

In general, the multiplicative model tends to have lower ratings, and the variance is higher compared to the additive model. But, the result is strongly correlated, as can be seen in the scatter plot in Fig. 5, where the mean MOS rating using both models are compared. The x-axis shows the average MOS using the additive model, the y-axis the average MOS with the multiplicative model. Each dot represents a combination of bandwidth scenario and buffer configuration. As the dots lie on a straight line, this means there is a linear relationship between both models.

We can conclude, that both models are equally suited for QoE evaluation, although both models are conventionally different. In the practically relevant scenarios, the correlation between both models is linear. Since the weighing factors to differentiate stalling and initial waiting considering real usage still have to be determined, the different level of MOS ratings are negligible. Also, the rating of stalling in videos longer than 60 s has to be investigated, to compare videos of different length. With this linear model, long videos have a clear advantage over short videos, because a single stalling in a short video is much worse than in a

longer video. Additionally, if a video is longer, the buffer has a higher average fill level [15], so that a longer bandwidth degradation can be compensated.

5 Conclusion and Outlook

In this paper, we used a QoE model to rate the playback of HAS video players. We compared these to extensive manual interpretation. This shows, that the QoE metric is suited in practice to evaluate and rate the video playback by its Quality of Service (QoS), thus technical playback properties. So, this can replace the extensive analysis of different QoS metrics, for example stallings, quality or initial waiting. QoE is a holistic approach to rate the overall user experience. Apart of typical QoS metrics, it could also include the users' device type, as this also has significant impact [9].

The testbed that was presented in this paper can be used to test video players, that can be embedded into websites and provide a JavaScript API. So, also other video players can be tested, which may come with different adaptation or buffering strategies. Future work could also consider the video quality and switches. These methods can be extended to allow QoE management in real-time and thus monitoring of customers of video streaming services or website visitors. This can lead to a higher service quality by distributing load, adapting the playback to the users, and improve adaptation algorithms and real-time monitoring. Providers of video streaming platforms could include QoE monitoring into existing monitoring solutions, for early detection of quality impairments.

An open issue is how to treat longer videos. The QoE model used in this paper has been developed using videos with a duration of 30 s, and is it not validated for longer videos. We normalized the results of the stalling to a video of 30 s, but without limitation this method can only be used to compare videos of about the same length. It favors longer videos compared to short video clips. Investigating this poses a challenge as long-term user-studies are difficult to implement, as longer experiments take much more time to be performed and are too long for typical lab tests. Additionally, during these tests, there are much more influencing factors than during shorter tests, which make it difficult to distinguish them.

References

1. Brunnström, K., Beker, S.A., De Moor, K., Dooms, A., Egger, S., Garcia, M.N., Hossfeld, T., Jumisko-Pyykkö, S., Keimel, C., Larabi, M.C., et al.: Qualinet white paper on definitions of quality of experience (2013)
2. Chen, L., Zhou, Y., Chiu, D.M.: Video browsing-a study of user behavior in online VoD services. In: 2013 22nd International Conference on Computer Communications and Networks (ICCCN), pp. 1–7. IEEE (2013)
3. Finamore, A., Mellia, M., Munafò, M.M., Torres, R., Rao, S.G.: Youtube everywhere: Impact of device and infrastructure synergies on user experience. In: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, IMC 2011, pp. 345–360. ACM, New York (2011). http://doi.acm.org/10.1145/2068816.2068849

4. Hoßfeld, T., Seufert, M., Sieber, C., Zinner, T.: Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming. In: 2014 Sixth International Workshop on Quality of Multimedia Experience (QoMEX), pp. 111–116 (2014)
5. Hoßfeld, T., Egger, S., Schatz, R., Fiedler, M., Masuch, K., Lorentzen, C.: Initial delay vs. interruptions: Between the devil and the deep blue sea. In: 2012 Fourth International Workshop on Quality of Multimedia Experience (QoMEX), pp. 1–6. IEEE (2012)
6. Hoßfeld, T., Heegaard, P.E., Varela, M., Möller, S.: QoE beyond the MOS: an in-depth look at QoE via better metrics and their relation to MOS. Qual. User Exp. **1**(1), 2 (2016)
7. Hoßfeld, T., Moldovan, C., Schwartz, C.: To each according to his needs: Dimensioning video buffer for specific user profiles and behavior. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 1249–1254. IEEE (2015)
8. Hoßfeld, T., Skorin-Kapov, L., Heegaard, P.E., Varela, M., Chen, K.T.: On additive and multiplicative QoS-QoE models for multiple QoS parameters. In: Proceedings of the 5th ISCA/DEGA Workshop on Perceptual Quality of Systems PQS 2016. ISCA (2016)
9. Kara, P.A., Bokor, L., Sackl, A., Mourão, M.: What your phone makes you see: Investigation of the effect of end-user devices on the assessment of perceived multimedia quality. In: 2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX), pp. 1–6 (2015)
10. International Telecommunication Union: P.800: Methods for subjective determination of transmission quality (1996)
11. Müller, C., Lederer, S., Timmerer, C.: An evaluation of dynamic adaptive streaming over HTTP in vehicular environments. In: Proceedings of the 4th Workshop on Mobile Video, MoVid 2012, pp. 37–42. ACM, New York (2012). http://doi.acm.org/10.1145/2151677.2151686
12. Riiser, H., Vigmostad, P., Griwodz, C., Halvorsen, P.: Commute path bandwidth traces from 3G networks: Analysis and applications. In: Proceedings of the 4th ACM Multimedia Systems Conference, MMSys 2013, pp. 114–118. ACM, New York (2013). http://doi.acm.org/10.1145/2483977.2483991
13. Seufert, M., Egger, S., Slanina, M., Zinner, T., Hoßfeld, T., Tran-Gia, P.: A survey on quality of experience of HTTP adaptive streaming. IEEE Commun. Surv. Tutorials **17**(1), 469–492 (2015)
14. Song, W., Tjondronegoro, D.W.: Acceptability-based QoE models for mobile video. IEEE Trans. Multimedia **16**(3), 738–750 (2014)
15. Surminski, S., Moldovan, C., Hoßfeld, T.: Saving bandwidth by limiting the buffer size in HTTP adaptive streaming. In: MMBnet 2017 - Proceedings of the 9th GI/ITG Workshop "Leistungs-, Verlässlichkeits- und Zuverlässigkeitsbewertung von Kommunikationsnetzen und Verteilten Systemen", pp. 5–21 (2017)
16. Wamser, F., Casas, P., Seufert, M., Moldovan, C., Tran-Gia, P., Hossfeld, T.: Modeling the youtube stack: From packets to quality of experience. Comput. Netw. **109**(Part 2), 211–224 (2016). http://www.sciencedirect.com/science/article/pii/S1389128616300925

Tool Papers

A Domain-Specific Language and Toolchain for Performance Evaluation Based on Measurements

Freek van den Berg[1(✉)], Jozef Hooman[2,3], and Boudewijn R. Haverkort[1]

[1] DACS, University of Twente, Enschede, The Netherlands
{f.g.b.vandenberg,b.r.h.m.haverkort}@utwente.nl
[2] ICIS, Radboud University Nijmegen, Nijmegen, The Netherlands
[3] Embedded Systems Innovation (ESI) by TNO, Eindhoven, The Netherlands

Abstract. This tool paper presents iDSL, a language and a fully automated toolchain for evaluating the performance of service-oriented systems. In this work, we emphasize the use of a *high-level domain specific language* that is tailored to be understood by system designers and domain experts, a transformation into an underlying process algebra which contains latency distribution functions based on *real measurements* for calibration, and the *integration of analysis tools* under the hood. Altogether, the approach delivers intuitive, visual results.

1 Motivation

Embedded systems are computer systems that have a dedicated function within a larger system, often with real-time constraints [19]. Hence, their performance is vital. However, good performance is hard to achieve, because embedded systems come with increasingly heterogeneous, parallel and distributed architectures and may comprise many product lines and different configurations.

Here, we consider service-oriented systems [10–15], a subclass of embedded systems, which: (i) provide services to their environment, accessible via so-called requests; (ii) each service request leads to one response; (iii) service requests are functionally isolated from each other; but, (iv) may affect each other's performance by competing for the same resource in the service-oriented system.

We propose a performance evaluation framework that can be used to *evaluate* the performance of service-oriented systems based on real *measurements* for calibration (Contribution $\mathcal{C}1$). We realize this framework via iDSL, which comprises the domain-specific, *high-level* iDSL language (Contribution $\mathcal{C}2$) to model service-oriented systems and the iDSL toolchain (Contribution $\mathcal{C}3$) to evaluate the performance of these systems in a fully automatic fashion. This approach separates the description of the user concerns from the solution approach, in accordance with the Declarative Performance Engineering (DPE, [16]) approach.

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 295–301, 2018.
https://doi.org/10.1007/978-3-319-74947-1_21

2 The High-Level iDSL Language

The iDSL language [10–15] has been developed to model service-oriented systems. It is tailored to be used and understood by system designers and experts in the service-oriented systems domain, in line with $\mathcal{C}2$. Figure 1 depicts the six high-level concepts of the iDSL language, as follows. A service system (Fig. 1-C) provides services to consumers in its environment. A consumer can send a request for a specific service at a certain time, after which the system responds with some delay. A service is implemented using a process (A), resources (B) and a mapping. A process decomposes high-level service requests into atomic tasks, which are each assigned to a resource in the mapping. Resources are capable of performing one atomic task at a time, in a certain amount of time. When multiple services are invoked, their resource needs may overlap, causing contention and making performance analysis harder. A scenario (D) consists of a number of invoked service requests over time to observe specific performance behavior of the system. A study (E) evaluates a selection of systematically chosen scenarios to derive the system's underlying characteristics. Finally, measures of interest (F) define what performance metrics to obtain, given a system in a scenario.

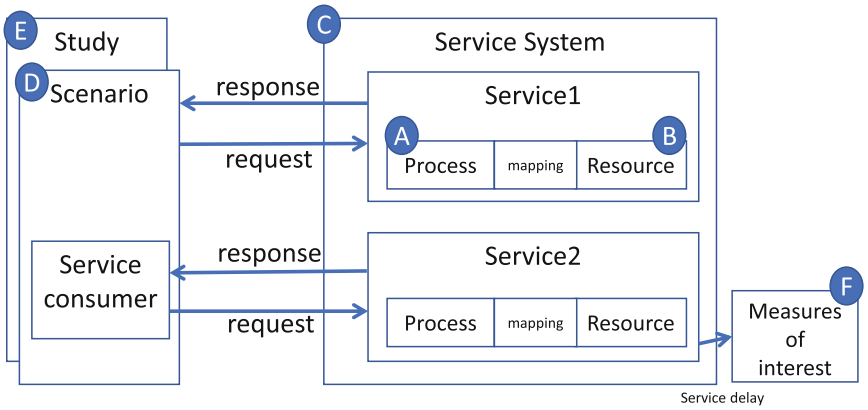


Fig. 1. The meta model of the iDSL language

For illustration, Table 1 provides an example iDSL language instance of a medical imaging system [14, Sect. 3], as follows. The process contains a sequence of the processes "image_pre_processing", "image_processing" and "image_post_processing". In turn, process "image_processing" decomposes into "motion_compensation", "noise_reduction" and "contrast". Each atomic process has a load, an amount of work. The resource contains a CPU with rate 2, i.e., it can process 2 loads per time unit, and a GPU with rate 5. The system combines the process and resource, and has a mapping to connect atomic tasks to resources. The scenario encompasses two streams of requests for the only service. Both streams have fixed inter-arrival times of 400.clearpage One stream has an

Table 1. An example service-oriented system, modeled using the iDSL language

(a) Process

```
Section Process
   ProcessModel image_processing_application
      seq image_processing_seq {
         atom image_pre_processing load 50
         seq image_processing { atom motion_compensation load 44
                     atom noise_reduction load uniform(80:140)
                     atom contrast load 134 }
      atom image_post_processing load 25 }
```

(b) Resource

```
Section Resource
   ResourceModel image_processing_PC decomp
      image_processing_decomp { atom CPU rate 2, atom GPU rate 5 }
```

(c) System

```
Section System
   Service image_processing_service
      Process image_processing_application
      Resource image_processing_PC
      Mapping assign {(image_pre_processing,CPU)(noise_reduction,CPU)
   (motion_compensation,CPU)(contrast,CPU)(image_post_processing,GPU) }
```

(d) Scenario

```
Section Scenario
   Scenario image_processing_run
      ServiceRequest image_processing_service at time 0, 400, ...
      ServiceRequest image_processing_service
         at time dspace("offset"), (dspace("offset")+400), ...
```

(e) Study

```
Section Study Scenario image_processing_run
   DesignSpace ("offset" "0" "20" "40" "80" "120" "160" "260")
```

(f) Measure

```
Section Measure
   Measure ServiceResponseTimes using 1 run of 250 requests
   Measure ServiceResponseTimes absolute
```

(g) Process with an injected EDF

```
Section Process
   ProcessModel normal_U100_O10_n100  palt { 2 atom load 91
      1 atom load 92 1 atom load 93 2 atom load 95 5 atom load 96
      9 atom load 97 9 atom load 98 15 atom load 99 15 atom load 100
      15 atom load 101 9 atom load 102 7 atom load 103 5 atom load 104
      3 atom load 105 2 atom load 107 }
```

initial delay of 0. The initial delay of the other is determined by an offset parameter, which is a variable that is defined in the so-called design space of the study. Finally, the measure contains two measures of interest referring to performance evaluation.

3 The Integrated iDSL Toolchain

In this section, we discuss the iDSL toolchain which ranges from creating an iDSL language instance to generating performance artifacts, in line with $\mathcal{C}3$.

Creating the performance model involves the conjoint modeling by a modeler and analyzer of a *case study* in the iDSL language. A modeler determines how the system behaves and generates a system model, i.e., a process, resource and system (cf. Fig. 1-A, B and C). The analyzer determines system usage and creates a study, i.e., scenario, study and measure (cf. Fig. 1-D, E and F).

During the modeling process, the Eclipse Integrated Development Environment [2] is used to support the user. This environment enables, among others, syntax highlighting, code completion, and "input validation", e.g., checking the code for invalid references, unused objects and ambiguous definitions. Also warnings and information boxes are displayed, e.g., when the design space is too large.

Under the hood, the iDSL grammar has been defined using the Xtext framework [18]. The toolchain functionality is programmed in the Xtend language [17].

In the following, we briefly describe the four main activities that constitute the performance analysis toolchain of iDSL.

Process Measurements. Measurements are performed on a real system and injected the into the iDSL model for calibration [15, Sect. 3]. The text-processing tool AWK [1] is used to facilitate this.

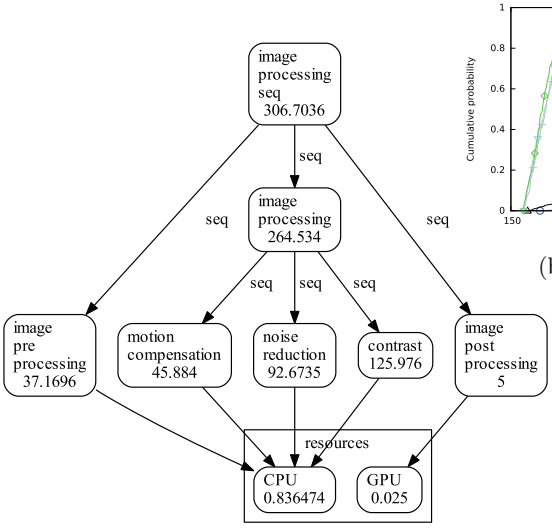
1. Perform measurements on a real system [15, Sect. 3.1].
2. Create Gantts: group measurements into execution times [15, Sect. 3.2].
3. Generate Empirical Distribution Functions (EDFs) [15, Sect. 3.3].
4. Inject the EDFs of step 3 into the IDSL model via a model transformation: represent EDFs as probabilistic alternatives (PALT, [4]) constructs, in line with $\mathcal{C}1$. For illustration, we have drawn 100 numbers from a normal distribution ($\mu = 100$, $\sigma = 10$) [7] representing measurements. Table 1g then shows the resulting EDF in iDSL. For instance, "2 atom load 91" means that the 100 drawn numbers contain 2 times value 91.

Model Simplification. iDSL determines whether the model can practically be evaluated [12, Sect. 4.3]. If not, it is simplified via a transformation, as follows.

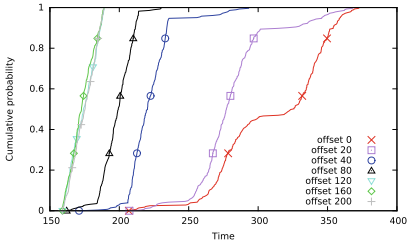
1. Cluster similar measurements in each generated EDF [12, Sect. 4.1].
2. Increase the time unit of all time occurrences in the model [12, Sect. 4.2].

Model evaluation is delegated to Modest [4].

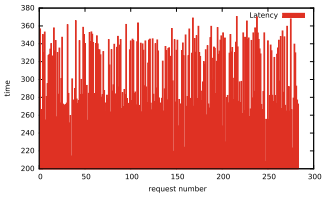
1. Create Modest models: transform iDSL into Modest [11, Sect. 4.3]
2. Evaluate the Modest models for performance using the Modest toolset.
   (a) Discrete-event simulation: yields average latencies [14, Sect. 4.2]:
   (b) Timed Automata (TA)-model checking: a binary search for absolute bounds [14, Sect. 4.2].
   (c) Probabilistic Timed Automata (PTA)-model checking: an iterative algorithm in which cumulative latency probabilities are computed one at a time [13, Sect. 4].
   (d) Efficient PTA-model checking: a carefully constructed combination of the aforementioned techniques [12, Sect. 6].
3. Parse results: parse the Modest results into high-level iDSL results.



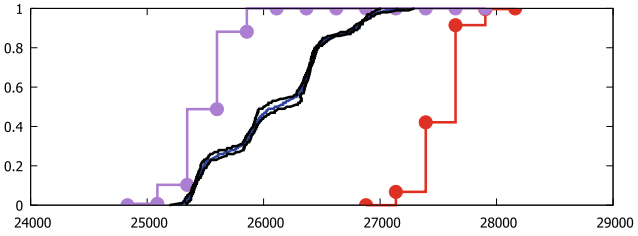
(a) A latency breakdown chart (offset=0)



(b) Multi-design latency CDF



(c) A latency bar graph (offset=0)



(d) The lower (in purple) and upper bound CDF (in red), the simulation avg. (in blue), and $\alpha = 0.95$ CI (in black)

Fig. 2. Four ways of representing latencies, generated from the iDSL code (Color figure online)

Create visualizations turns the parsed results into intuitive graphs.

1. Latency breakdown chart (see Fig. 2a): displays the structure of a service, i.e., the underlying processes and resources, and its dynamics, i.e., process latencies and resource utilizations.
2. Multi-design latency Cumulative Distribution Function (CDF, see Fig. 2b): provides latency CDFs for multiple designs in one graph to easily determine the effect of design decisions.
3. Latency bar chart (see Fig. 2c): shows the subsequent latency times of a service which provides insight in jitter, i.e., the variation of latencies.
4. Latency CDF (see Fig. 2d): provides a lower (purple) and upper bound (red) CDFs whose difference is the result of how nondeterminism is resolved.

Figure 2a–c are based on discrete-event simulations, and Fig. 2d on PTA-model checking. Figure 2a is made by GraphViz [3], the others by GNUplot [6].

4 Background

iDSL is different from tools such as the Modest toolset [4], Storm [8], UPPAAL [9] and PRISM [5]. Where the latter deliver relatively generic, widely-applicable languages, instead, iDSL provides a *domain-specific* language ($\mathcal{C}2$) which allows measurements-based calibration ($\mathcal{C}1$), and a fully automated toolchain ($\mathcal{C}3$).

References

1. Andrews, R., Jones, D., Williams, J., Thorson, P., Oliver, G., Costa, D., Le Boeuf, B.: Heart rates of northern elephant seals diving at sea and resting on the beach. J. Exp. Biol. **200**(15), 2083–2095 (1997)
2. Eclipse desktop & web IDEs. https://www.eclipse.org/ide/
3. Graphviz - Graph Visualization Software. http://www.graphviz.org/
4. Hartmanns, A., Hermanns, H.: The modest toolset: an integrated environment for quantitative modelling and verification. In: Ábrahám, E., Havelund, K. (eds.) TACAS 2014. LNCS, vol. 8413, pp. 593–598. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54862-8_51
5. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
6. Racine, J.: GNUplot 4.0: a portable interactive plotting utility. J. Appl. Econometrics **21**(1), 133–141 (2006)
7. RANDOM.ORG. https://www.random.org/gaussian-distributions/
8. Storm Checker. http://www.stormchecker.org
9. Uppsala Aalborg model checker. http://www.uppaal.org/
10. van den Berg, F.: Automated performance evaluation of service-oriented systems. Ph.D. thesis, University of Twente (2017)
11. van den Berg, F., Haverkort, B.R., Hooman, J.: iDSL: automated performance evaluation of service-oriented systems. In: Katoen, J.-P., Langerak, R., Rensink, A. (eds.) ModelEd, TestEd, TrustEd. LNCS, vol. 10500, pp. 214–236. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68270-9_11

12. van den Berg, F., Haverkort, B.R., Hooman, J.: Efficiently computing latency distributions by combined performance evaluation techniques. In: Proceedings of the 9th EAI International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2015, pp. 158–163. ICST (2015)
13. van den Berg, F., Hooman, J., Hartmanns, A., Haverkort, B.R., Remke, A.: Computing response time distributions using iterative probabilistic model checking. In: Beltrán, M., Knottenbelt, W., Bradley, J. (eds.) EPEW 2015. LNCS, vol. 9272, pp. 208–224. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23267-6_14
14. van den Berg, F., Remke, A., Haverkort, B.R.: A domain specific language for performance evaluation of medical imaging systems. In: 5th Workshop on Medical Cyber-Physical Systems. OpenAccess Series in Informatics, vol. 36, pp. 80–93. Schloss Dagstuhl (2014)
15. van den Berg, F., Remke, A., Haverkort, B.R.: iDSL: automated performance prediction and analysis of medical imaging systems. In: Beltrán, M., Knottenbelt, W., Bradley, J. (eds.) EPEW 2015. LNCS, vol. 9272, pp. 227–242. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23267-6_15
16. Walter, J., van Hoorn, A., Koziolek, H., Okanovic, D., Kounev, S.: Asking what?, Automating the how?: The vision of declarative performance engineering. In: Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering, pp. 91–94. ACM (2016)
17. Xtend. https://www.eclipse.org/xtend/
18. Xtext. https://www.eclipse.org/Xtext/
19. Zurawski, R.: Embedded Systems Handbook. CRC Press, Boca Raton (2005)

SLA Tool

Falko Bause and Peter Buchholz(✉)

Department of Computer Science, TU Dortmund, 44221 Dortmund, Germany
{falko.bause,peter.buchholz}@tu-dortmund.de

Abstract. Service level agreements (SLAs) specify commitments between providers and users of services. Concerning quantitative aspects it is common to guarantee response times under a maximal load. Often simulation models are used to validate those guarantees. The SLA calculus offers an efficient technique to establish and validate quantitative aspects of service level agreements. Based on bounds for the load and response times of single services, bounds for the response times of composed services can be calculated. Additionally, the SLA calculus determines bounds for the service capacities which the provider needs in order to fulfill the service level agreements.

This paper describes the *SLA Tool* which supports the calculation of such bounds using results from min/+-algebra. The tool consists of two parts: Octave functions, that implement the operators of the SLA calculus, and a graphical user interface (GUI) which facilitates their use also for users not familiar with the theory. In the following we concentrate on the user interface.

1 Introduction

Service-oriented architectures (SOAs) are normally described with a user-oriented and provider-oriented perspective in mind. With respect to quantitative measures users ask for acceptable response times, which can only be guaranteed by a provider, if the number and the size of arriving jobs is limited. Corresponding specifications are typically part of Service Level Agreements (SLAs) and can be described by bounds, e.g. for loads, service capacities, and response times. Violation of these bounds is often only allowed in a few exceptional cases. For validation it is common to simulate an adequate model of the system which is usually a complex and time-consuming task requiring detailed input data.

An analytical approach for a fast validation which is based on the network calculus [3], is provided by the SLA calculus [2,4,8,10]. The SLA calculus supports the calculation of response time bounds for composed services based on bounds for delays of the individual components and thus allows for the determination of SLAs when orchestrating services. In contrast to other analytical approaches used for SOA analysis as e.g. queueing network analysis [5,6], the SLA calculus does not determine mean values but sharp bounds which is often more interesting in a SOA context. Furthermore, other analysis techniques often require descriptions of arrival and service characteristics as input and derive

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 302–306, 2018.
https://doi.org/10.1007/978-3-319-74947-1_22

response times as output. In contrast, providers and orchestrators of SOAs are generally more interested in the calculation of necessary service capacities given customer related figures.

The SLA calculus aims at answering this kind of analysis problems. A drawback of the calculus is that the mathematical basis is far from being intuitive for software engineers, so that appropriate tool support is required. *SLA Tool* consists of two main parts: A set of Octave functions and a GUI. The Octave functions implement the operators of the SLA calculus and are based on operators from min/+-algebra. The Octave functions use (sequences of) piecewise linear functions as input and output piecewise linear functions as results.

In the following we sketch the GUI of the *SLA Tool* [1] which hides mathematical details of the approach for people not familiar with the theory and makes the approach operational.

2 The SLA Tool

Figure 1 shows the GUI of the *SLA Tool* which offers features to define services and to connect them resulting in a workflow specifying an orchestrated service.

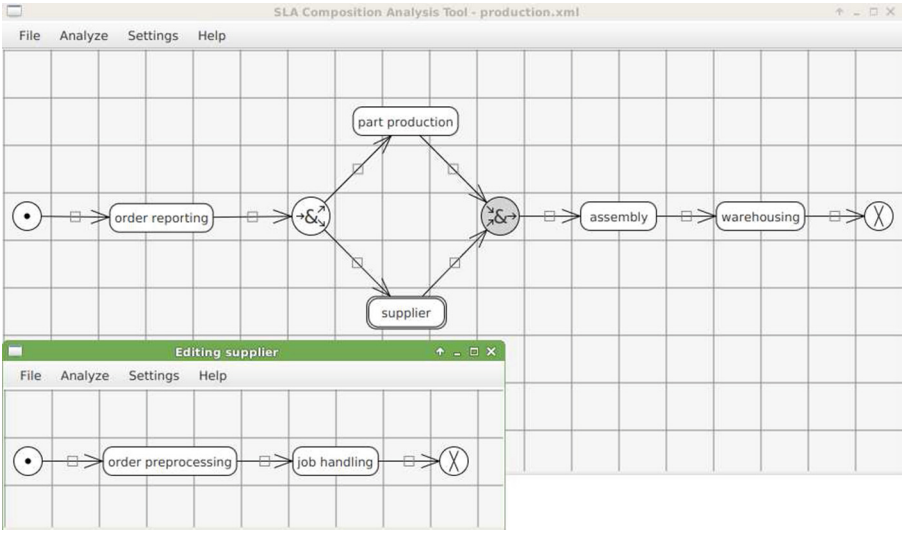


Fig. 1. Hierarchical model description

As common, also *SLA Tool* offers connectors to describe sequential and parallel compositions of services. Additionally, hierarchical model descriptions are supported by refining service descriptions. A composition of services with a single entry and exit point can be defined as a sub-service as e.g. service *supplier* in Fig. 1. Overall a composed service is described by an acyclic graph. For analysis,

bounds for arrivals and delay/response times need to be specified. Any bound is a non-decreasing function B which in *SLA Tool* is specified by a sequence of piecewise linear functions of the form

$$\{[x_1, y_1, s_1], \ldots, [x_i, y_i, s_i], \ldots, [x_n, y_n, s_n]\}, \;\; x_i < x_{i+1}$$

where $B(x_i) = y_i$. s_i specifies the slope of function B in the interval $[x_i, x_{i+1}]$. For example, $\{[0, 1, 3], [1, 4, 2]\}$ specifies a continuous function B where $B(x) = 3x+1$ for $0 \leq x < 1$ and $B(x) = 2(x-1)+4$ for $x \geq 1$. Function B might be discontinuous at x_i which is represented by a vertical line (cf. Fig. 2).

SLA Tool allows the specification of lower and upper bounds. In the following we will restrict our description mostly to upper bounds.

2.1 Input Specification of Bounds

In the world view of *SLA Tool* arrivals are service calls which deliver a portion of load to the service. The size of an arriving service call is measured in some application specific unit like e.g. the number of transactions or the number of bytes to be stored. These arriving load units are limited by arrival bounds.

An upper arrival bound B_A specifies that in an interval of length $x \in \mathbb{R}$ time units at most $B_A(x)$ load units will arrive in order to be compliant with the SLA specification. The service provider then guarantees an upper response time or delay bound B_D specifying that service calls of size x will be processed after at most $B_D(x)$ time units. Given upper bounds for arrivals and response times *SLA Tool* automatically calculates a corresponding lower service bound B_S which specifies that in an interval of x time units the service must be able to handle at least $B_S(x)$ load units.

Fig. 2. Bounds for arrival, delay and service

Figure 2 shows the input specification of a service of the workflow shown in Fig. 1. Vastag [9,10] presents an "onion bucket algorithm" for the computation of arrival and response time bounds from traces, so that the input of the tool can be determined from measurements.

2.2 Analysis Results

At the push of a button *SLA Tool* analyses the complete workflow by calling Octave functions which implement the operators of the SLA calculus [2,8,10]. The result is shown in Fig. 3.

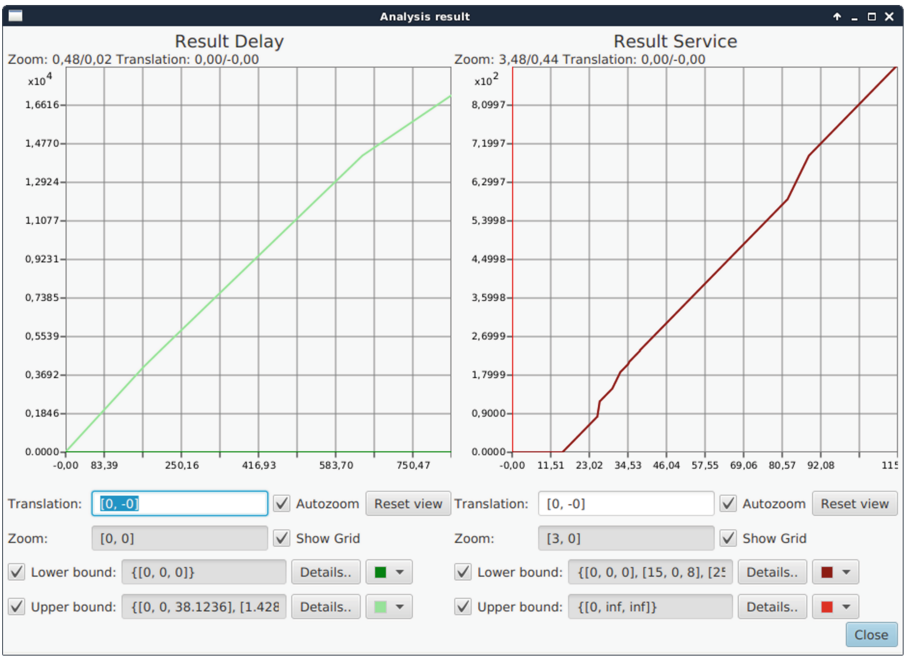


Fig. 3. Analysis results

The delay or response time bound respectively determines the SLA assertion the provider of the orchestrated service is able to guarantee to customers. The upper bound B_D of the delay curve shown in Fig. 3 states that customer calls of size x will be served after at most $B_D(x)$ time units. Similarly, the lower service bound shown in the right part of Fig. 3 specifies that the composed service has at least a capacity of B_S load units, or precisely that in an interval of x time units at least $B_S(x)$ load units can be served.

3 Conclusions

We presented the *SLA Tool* emphasizing its GUI and the necessary user input. The tool helps to obtain sharp bounds for essential performance figures of composed services and in this way supports the specification of SLAs for providers of complex services. In the future we plan to support the execution and evaluation of series of experiments to support optimization (cf. [1]).

SLA Tool is free software, licensed under the Apache License and can be obtained from [7].

References

1. Bause, F., Buchholz, P., May, J.: A tool supporting the analytical evaluation of service level agreements. In: Binder, W., Cortellessa, V., Koziolek, A., Smirni, E., Poess, M. (eds.) Proceedings 8th ACM/SPEC on International Conference on Performance Engineering, ICPE 2017, L'Aquila, Italy, 22–26 April 2017, pp. 233–244. ACM (2017)
2. Buchholz, P., Vastag, S.: Toward an analytical method for SLA validation. Softw. Syst. Model. (to appear)
3. Le Boudec, Jean-Yves, Thiran, Patrick (eds.): Network Calculus: A Theory of Deterministic Queuing Systems for the Internet. LNCS, vol. 2050. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45318-0
4. Buchholz, P., Vastag, S.: An introduction to SLA calculus for the analytical validation of SLAs. Technical report, Informatik 4, TU Dortmund (2015). http://ls4-www.cs.tu-dortmund.de/download/buchholz/sla.pdf
5. Menascé, D.A.: Composing web services: a QoS view. IEEE Internet Comput. **8**(6), 88–90 (2004)
6. Menascé, D.A.: Mapping service-level agreements in distributed applications. IEEE Internet Comput. **8**(5), 100–102 (2004)
7. SLA TOOL. http://ls4-www.cs.tu-dortmund.de/cms/en/research/software/SLA_Tool/index.html
8. Vastag, S.: A calculus for SLA delay properties. In: Schmitt, J.B. (ed.) MMB&DFT 2012. LNCS, vol. 7201, pp. 76–90. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28540-0_6
9. Vastag, S.: Arrival and delay curve estimation for SLA calculus. In: Proceedings of the 2012 Winter Simulation Conference (WSC), Berlin, Germany, pp. 1–12 (2012)
10. Vastag, S.: SLA Calculus. Ph.D. thesis, Department of Computer Science, TU Dortmund (2014)

A Tool for Generating Automata of IEC60870-5-104 Implementations

Max Kerkers[1], Justyna J. Chromik[1(✉)], Anne Remke[1,2], and Boudewijn R. Haverkort[1]

[1] University of Twente, Enschede, Netherlands
m.kerkers@alumnus.utwente.nl,
{j.j.chromik,a.k.i.remke,b.r.h.m.haverkort}@utwente.nl
[2] University of Münster, Münster, Germany

Abstract. Power distribution networks are often controlled using the communication protocol IEC 60870-5-104 (IEC-104). While a specification exists, not every device implementing this protocol, actually follows this specification. We present *mealy104*, a tool that infers finite-state automata from IEC-104 implementations and use it on a real device implementing IEC-104, comparing it to the protocol standard. We use the tool to show that implementations do deviate from the specification.

Keywords: ICS · Power grid · SCADA · Mealy machine · IEC-104

1 Introduction

Implementations of communication protocols should closely follow their specification, as differences or ambiguities might lead to security issues, as recently shown by the vulnerability that was found in the popular Wi-Fi protocol WPA2 [7]. Similar problems might occur with any protocol, if the specification contains ambiguity or if implementations do not follow the standard. Vulnerabilities in industrial control protocols like IEC-104 pose a serious threat to critical infrastructures, such as the power distribution grid. The implementations of industrial control protocols are often not checked against protocol specifications.

To verify whether an implementation follows a specification, both can be represented as finite state machine and then compared. The automaton representing the specification should be part of the standard. The other automaton can be learned from the implementation, e.g., using the tool presented in this paper. It implements a variant of Angluin's $L^\star$ algorithm [1], which produces Mealy machines that can represent more complex behaviour of input/output systems [5]. This algorithm has been applied before, e.g., for determining the correct operation of the ABN Amro e.dentifier2 [2], or implementations of the Transport Layer Security protocol [3]. To the best of our knowledge, we present the first tool to check communication protocols in *SCADA networks* that is made available under the Gnu General Public License. The source code of this tool can be found on GitHub[1].

[1] https://github.com/mkerkers/mealy104.

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 307–311, 2018.
https://doi.org/10.1007/978-3-319-74947-1_23

We propose a tool developed for the automated generation of Mealy machines for implementations of IEC-104 [4], which is crucial for the communication between control and field stations in power distribution in Europe. The tool we developed generates a formal representation from an IEC-104 implementation. We tested three IEC-104 simulators and two real-life devices with our tool [4, Chap. 5]. While **none** of the simulators implemented the protocol according to its specification, the investigated hardware, i.e., Sprecher Sprecon-E-C-92 and Datawatt D05-Lite, only partially matched the specification.

This paper describes the most relevant information about the IEC-104 protocol in Sect. 2 and the tool setup and most significant components in Sect. 3. Finally, Sect. 4 presents a case study on a real-life IEC-104 implementation.

2 SCADA Protocol IEC-104

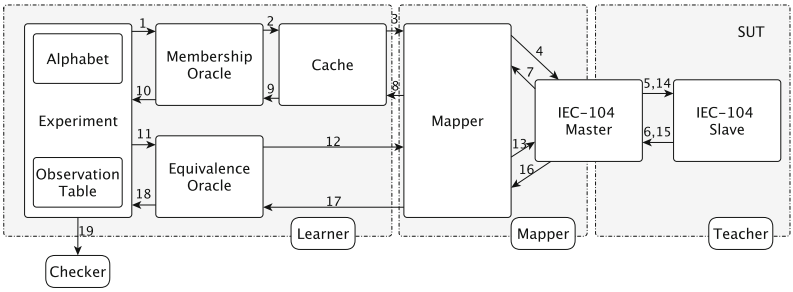
IEC-104 [6] describes two different layers: the Application Protocol Control Information (APCI) layer and the Application Service Data Unit (ASDU) layer. The first runs on top of the TCP layer and has three message formats: (i) unnumbered control functions (U-format), (ii) numbered Supervisory functions (S-format), and (iii) the Information transfer format (I-format). **U-format** messages either (de)activate a connection using STARTDT (start data transfer) and STOPDT (stop data transfer) or test whether a connection is still active using TESTFR (test frame). **I-format** messages transfer data. They use TypeIDs to define what kind of message is sent, using, e.g., General Interrogation (C_IC_NA_1) or Single Command (C_SC_NA_1) numbers that range from 0 to 255. **S-format** messages acknowledge previously received I-format messages.

3 Description of *mealy104*: Components and Set-up

We first describe the main components of the tool before discussing its general setup. As shown in Fig. 1, the learner builds the automaton based on input queries from the alphabet that are translated by the mapper to actual IEC-104 messages. The teacher, i.e., the queried device, answers these request. Once an automaton is constructed, the checker tests it against the protocol specification.

The **Alphabet** implements all IEC-104 message format types: three U-format types, the S-format type, and one for each I-format category. The complete alphabet is available in [4, Appendix A]. The **Learner** is built using the framework LearnLib[2]. For each run, a suitable sub-alphabet is chosen to create automata implementing the $L^{\star}_{M}$ algorithm [5]. The **Mapper** translates between abstract (human-readable) messages on the learner side and actual messages on the teacher side. For every abstract message in the alphabet, the mapper contains an implementation of a concrete message, structured according to the format as defined in the IEC-104 specification. To implement these concrete messages, OpenMUC j60870[3] is used.

[2] http://learnlib.de.
[3] https://www.openmuc.org/iec-60870-5-104/.



1	Learn / Refine Hypothesis	12	Abstract Equivalence Query
2, 3	Abstract Membership Query	13, 14	Concrete Equivalence Query
4, 5	Concrete Membership Query	15, 16	Concrete Equivalence Answer
6, 7	Concrete Membership Answer	17	Abstract Equivalence Answer
8, 9	Abstract Membership Answer	18	Confirmation / Counterexample
10, 11	Hypothesis	19	Final Hypothesis

Fig. 1. Block diagram of the *Mealy104* finite-state automata learner

The **Teacher** is implemented as master using OpenMUC j60870 such that all fields in APDUs are adjustable, and sending and receiving of STOPDT and TESTFR messages is included. The **Checker** tests if the automaton of the implementation matches the automaton deduced from the standard [6], as shown in Fig. 2; it has been built using AutomataLib[4], and traverses both automata using the same inputs, and comparing the outputs.

In more detail, the learner consists of: (i) the experiment, (ii) the membership oracle and cache, and (iii) the equivalence oracle. The experiment uses the membership oracle (Step 1 in Fig. 1) to learn a hypothesis, structured as an Observation Table. During learning, the membership and equivalence oracles send abstract queries to the mapper (Steps 2, 3 and 12), from which they receive abstract answers (Steps 8, 9 and 17). A cache between the membership oracle and the mapper stores each query and its corresponding answer. The mapper translates each abstract query it receives (Steps 3 and 12), into a concrete IEC-104 message which is forwarded to the IEC-104 Master (Steps 4 and 13), and from there to the IEC-104 Client (Subject Under Test). The SUT replies with a concrete answer (Steps 15 and 16), which the Mapper translates into an abstract answer and sends to the oracles (Steps 8 and 17). The membership oracle continues sending queries until the hypothesis is closed and consistent. Then, this learned hypothesis is returned (Step 10) and passed to the equivalence oracle (Step 11), which attempts to find a counterexample (Step 18). A counterexample is added to the Observation Table and the learning is restarted. Without a counterexample, the final hypothesis is transformed into a Mealy machine and passed to the checker (Step 19).

The equivalence oracle first checks for inconsistencies with the cache, then it sends random queries to the mapper, checking if the responses match the hypothesis. The tool is configured to send 1000 random queries to the SUT, and it resets the SUT and the hypothesis to the initial starting position with

[4] https://github.com/LearnLib/automatalib.

probability 1%. These settings provide a traversal that is both extensive and time bound. If a random query contradicts the hypothesis, a counterexample has been found. If none is found after 1000 random queries, the hypothesis is assumed to be confirmed. To compare the hypothesis to the standard specification, the tool contains the standard automaton (cf. Fig. 2) [4, Sect. 3.5].

4 Case Study and Outlook

We tested several simulators of the IEC-104 protocol and two real devices used in the Dutch power distribution system [4, Chap. 5]. The Axon Test Simulator and the Siemens IEC-Test Simulator generated only one state, where all inputs were accepted; the Mitra Software IEC 870-5-104 Simulator generated two states: one required to initiate the connection by sending U[STARTDT] message, and then it accepted any input. Furthermore, the Sprecher Sprecon-E-C-92 did not match the specification. For example in the UNCONFIRMED STOPPED state, according to the specification the connection should be terminated upon receiving an I-format message. Instead, the device keeps accepting incoming I-format messages.

In the following, we present one result of a DataWatt D05-Lite device used as an IEC-104 RTU in a field station. We run the tool for different subsets of the alphabet. For most of the cases, the obtained Mealy machine matches the one provided by the standard. However, one automaton was learned that does not comply to the standard when sending the I-format message for File Select. Figure 2 shows the automaton from the standard, whereas Fig. 3 shows the learned automaton. We used the same name and color for corresponding states in both automata. While they largely overlap, an additional state, indicated by 'X' (colored orange) can be observed in Fig. 3. As the File Select message does not

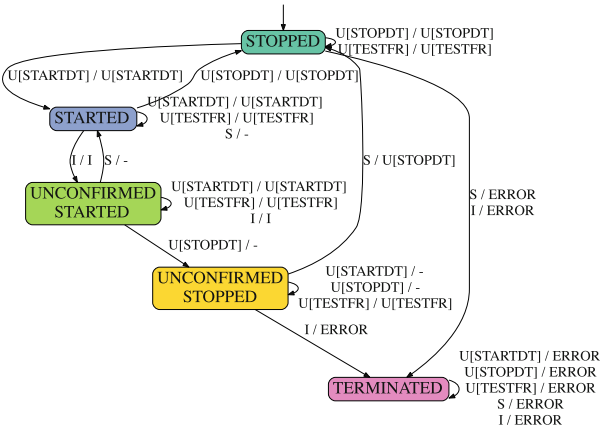


Fig. 2. Automaton derived from the IEC-104 standard for any I-frame (Color figure online)

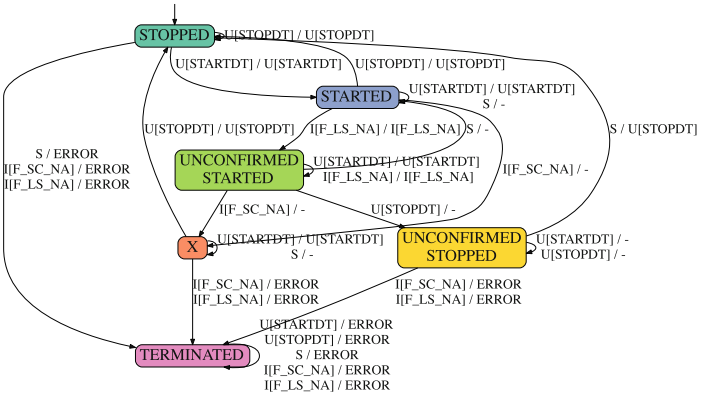


Fig. 3. Automaton learned using alphabet containing File Select I-frame. (Color figure online)

contain a valid address, the RTU is stricter than the standard. It terminates the connection on the next received I-format message, while the standard expects a negative confirmation I-format message. Hence, the behavior specified by the standard is *not fully implemented* by the investigated device.

Outlook. This tool can be used by vendors or users of devices implementing the IEC-104 protocol. The variety in the implementations of the IEC-104 protocol is alarming for both parties. The deviation in the presented implementation was found in a rarely used File Select function. However, such rare scenarios are often exploited [7]. Note that the presented tool can be adjusted to other protocols by adapting its components.

References

1. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75**(2), 87–106 (1987)
2. Chalupar, G., Peherstorfer, S., Poll, E., de Ruiter, J.: Automated reverse engineering using lego. In: Proceedings of the 8th USENIX Workshop on Offensive Technologies, p. 9 (2014)
3. de Ruiter, J., Poll, E.: Protocol state fuzzing of tls implementations. In: USENIX Security Symposium, pp. 193–206 (2015)
4. Kerkers, M.: Assessing the security of IEC 60870-5-104 implementations using automata learning. Technical report, May 2017
5. Shahbaz, M., Groz, R.: Inferring mealy machines. In: Cavalcanti, A., Dams, D.R. (eds.) FM 2009. LNCS, vol. 5850, pp. 207–222. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-05089-3_14
6. TC 57 - Power systems management and associated information exchange. IEC 60870-5-104:2006. Technical report, International Electrotechnical Commission, Geneva (2006)
7. Vanhoef, M., Piessens, F.: Key reinstallation attacks: Forcing nonce reuse in WPA2 (2017)

A Software Tool for the Compact Solution of the Chemical Master Equation

Tuğrul Dayar[1(✉)] and M. Can Orhan[2]

[1] Department of Computer Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey
tugrul@cs.bilkent.edu.tr
[2] Kanava Technologies, 06800 Ankara, Turkey
m.canorhan@gmail.com

Abstract. The problem of computing the transient probability distribution of countably infinite multidimensional continuous-time Markov chains (CTMCs) arising in systems of stochastic chemical kinetics is addressed by a software tool. Starting from an initial probability distribution, time evolution of the probability distribution associated with the CTMC is described by a system of linear first-order ordinary differential equations, known as the chemical master equation (CME). The solver for the CME uses the time stepping implicit backward differentiation formulae (BDF). Solution vectors in BDF can be stored compactly during transient analysis in one of the Hierarchical Tucker Decomposition, Quantized Tensor Train, or Transposed Quantized Tensor Train formats.

Keywords: Continuous-time Markov chain
Chemical master equation · Backward differentiation
Compact vector · Kronecker decomposition

1 The Problem

Letting the initial probability distribution vector of the infinitesimal generator matrix Q underlying a multidimensional continuous-time Markov chain (CTMC) be denoted by $\boldsymbol{\pi}_0$, the transient probability distribution vector $\boldsymbol{\pi}_t \in \mathbb{R}_{\geqslant 0}^{1\times|\mathcal{R}|}$ of Q at time $t \in \mathbb{R}_{\geqslant 0}$ satisfies [12]

$$\frac{d\boldsymbol{\pi}_t}{dt} = \boldsymbol{\pi}_t Q, \quad \boldsymbol{\pi}_t \boldsymbol{e} = 1. \tag{1}$$

Here, $\mathcal{R}$ is the reachable state space of the CTMC and $\boldsymbol{e}$ is a vector of 1's.

When the CTMC arises in the area of systems of stochastic chemical kinetics, (1) is referred to as the chemical master equation (CME) [4]. In this case, there are a finite number of dimensions and transitions, but $\mathcal{R}$ is almost always countably infinite. Therefore, a CTMC $\{S(t), t \geqslant 0\}$ having H dimensions such that $S(t) = (S_1(t), \ldots, S_H(t))$ and $\Pr(S(t) = \mathbf{i}) = \Pr(S_1(t) = i_1, \ldots, S_H(t) = i_H)$

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 312–316, 2018.
https://doi.org/10.1007/978-3-319-74947-1_24

can be used with the state vector $\mathbf{i} = (i_1, \ldots, i_H)$. The state space of dimension h is given by $\mathcal{S}^{(h)} = \mathbb{Z}_{\geqslant 0}$ for $h = 1, \ldots, H$, and when there are no unreachable states, we have $\mathcal{R} = \times_{h=1}^{H} \mathcal{S}^{(h)}$ and K transition classes in which transition class k is represented by the pair $(\alpha_k(\boldsymbol{i}), \boldsymbol{v}^{(k)})$ for $k = 1, \ldots, K$. Here, $\alpha_k(\boldsymbol{i}) \in \mathbb{R}_{\geqslant 0}$ is the transition rate function specifying the transition rate from state $\boldsymbol{i} \in \mathcal{R}$ to state $(\boldsymbol{i} + \boldsymbol{v}^{(k)}) \in \mathcal{R}$ and $\boldsymbol{v}^{(k)} \in \mathbb{Z}^{1\times H}$ is the state change vector specifying the successor state of the transition, with $v_h^{(k)}$ denoting the change in state variable $i_h \in \mathcal{S}^{(h)}$ due to a class k transition [3]. That $\mathcal{R}$ is equal to the product state space is a property of the models under consideration; but, this can be relaxed with the help of a well known hierarchical state space structuring approach [1].

A Kronecker representation for models in this area, which has separable state dependent transition rate functions in the form

$$\alpha_k(\boldsymbol{i}) = \phi_k \prod_{h=1}^{H} \alpha_k^{(h)}(i_h) \ ,$$

can be obtained by letting the transition matrix of dimension h with state space $\mathcal{S}^{(h)}$ for $h = 1, \ldots, H$ and transition class $k = 1, \ldots, K$ be denoted by $Q_k^{(h)} \in \mathbb{R}_{\geqslant 0}^{|\mathcal{S}^{(h)}|\times|\mathcal{S}^{(h)}|}$ and given entrywise as

$$Q_k^{(h)}(i_h, j_h) = \begin{cases} \alpha_k^{(h)}(i_h) & \text{if } j_h = i_h + v_h^{(k)} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i_h, j_h \in \mathcal{S}^{(h)} \ .$$

Then

$$Q = Q_O + Q_D, \quad Q_O = \sum_{k=1}^{K} \phi_k \bigotimes_{h=1}^{H} Q_k^{(h)}, \quad Q_D = -\sum_{k=1}^{K} \phi_k \bigotimes_{h=1}^{H} \mathrm{diag}(Q_k^{(h)} \boldsymbol{e}).$$

Next, we introduce a tool to solve the initial value problem associated with the system of linear first-order ordinary differential equations (ODEs) in (1) [12].

2 A Software Tool

We present a software tool [2] for the transient analysis of countably infinite multidimensional CTMCs introduced in the previous section in a sequential setting. Details regarding the tool may be obtained from its user manual. Time is discretized into smaller time steps and the solver for the CME [10] uses the implicit backward differentiation formulae (BDF). BDF methods are a class of implicit multistep methods to solve stiff ODEs [12]. Stiffness generally manifests itself when reaction rates occur at different time scales, and this is the case for many realistic systems. The o-step BDF method, denoted BDFo, keeps approximations of solutions at o previous time steps and computes the solution at the current time step by solving a linear system. BDFo methods have local truncation error proportional to the oth power of the step size, and therefore, are said to be of order o. The particular solver initializes the first o backward differences

with the embedded Runge–Kutta method due to Fehlberg, written RKFk–1(k), which is an order k method without the error estimate [12].

At each time step, n, the reachable state space, $\mathcal{R}$, is truncated by using a well defined aggregation operator on the prediction vector of BDFo [11] to obtain $\mathcal{R}_n$ [10]. Solution vectors can be stored compactly during transient analysis using one of the Hierarchical Tucker Decomposition (HTD) [5], Quantized Tensor Train (QTT) [8], or Transposed Quantized Tensor Train (QT3) [7] formats. Compact vectors in HTD format can work with a truncated generator matrix represented as a sum of Kronecker products of small molecule matrices, whereas those in QTT/QT3 format can work with a low-rank approximation of the truncated generator matrix in the same format [10].

The solution of the linear system at each time step in BDF is performed by the Jacobi iteration [12] using the Newton-Schulz method [9] to compute reciprocals of diagonal elements of the coefficient matrix for HTD and the density matrix renormalization group (DMRG) method for QTT/QT3 [7]. It is possible to use fixed and adaptive rank control strategies with compact vectors in HTD format. There are HTD_A, HTD_M, and HTD_F variants of the BDFo solver in which (adaptive, adaptive), (fixed, adaptive), and (fixed, fixed) rank bounds are used in (Jacobi, Newton-Schulz) methods [10].

Next we show examples of results that can be obtained with the tool.

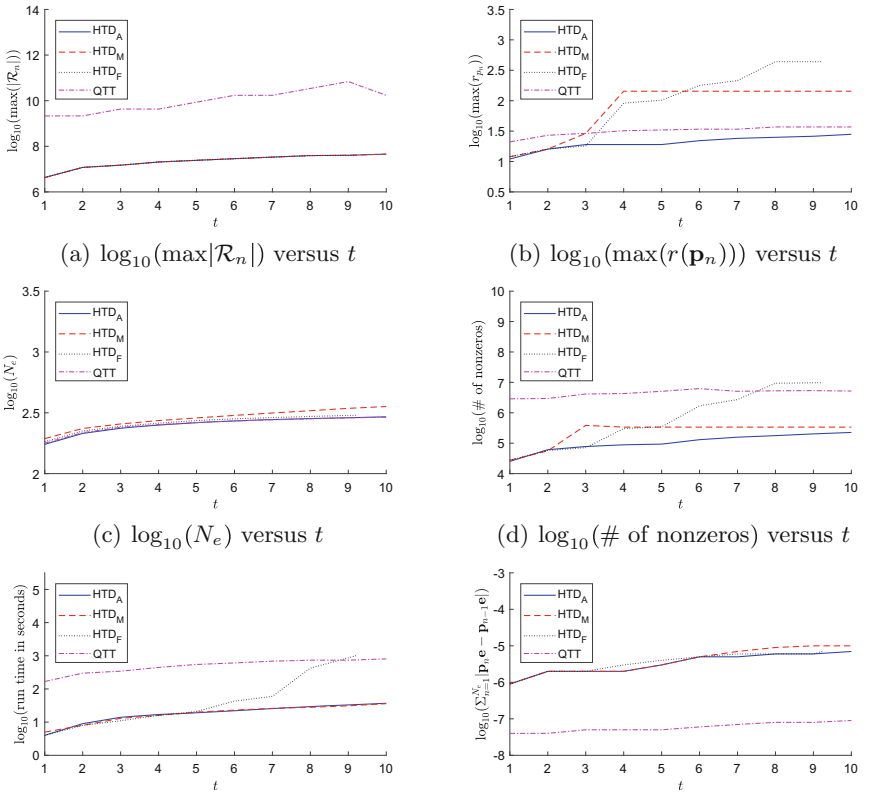
3 An Example

We consider a cascade model [6] that has five molecules each corresponding to a different dimension with the transition classes in Table 1. Here, $H = 5$, $\boldsymbol{i} = (i_1, i_2, i_3, i_4, i_5)$, $K = 10$, a, b, c, $\mu \in \mathbb{R}_{>0}$, and $\mathbf{e}_h$ is the hth principal axis vector. We let $a = 0.7$, $b = 1$, $c = 5$, and $\mu = 0.07$ as in [6].

The cascade model is analyzed using BDF5 with an accuracy tolerance of 10^{-9} and the indicated compact vector formats starting from the initial distribution $\boldsymbol{\pi}_0(10, 10, 10, 10, 10) = 1$ for final time values $t \in \{1, \ldots, 10\}$. A maximum run time of 1,000 seconds is imposed on the experiments performed on an Intel

Table 1. Transition classes of the cascade model

k	ϕ_k	$\alpha_k^{(1)}(i_1)$	$\alpha_k^{(2)}(i_2)$	$\alpha_k^{(3)}(i_3)$	$\alpha_k^{(4)}(i_4)$	$\alpha_k^{(5)}(i_5)$	$\boldsymbol{v}^{(k)}$
1	a	1	1	1	1	1	$\boldsymbol{e}_1^T$
2	μ	i_1	1	1	1	1	$-\boldsymbol{e}_1^T$
3	b	$\frac{i_1}{bi_1+c}$	1	1	1	1	$\boldsymbol{e}_2^T$
4	μ	1	i_2	1	1	1	$-\boldsymbol{e}_2^T$
5	b	1	$\frac{i_2}{bi_2+c}$	1	1	1	$\boldsymbol{e}_3^T$
6	μ	1	1	i_3	1	1	$-\boldsymbol{e}_3^T$
7	b	1	1	$\frac{i_3}{bi_3+c}$	1	1	$\boldsymbol{e}_4^T$
8	μ	1	1	1	i_4	1	$-\boldsymbol{e}_4^T$
9	b	1	1	1	$\frac{i_4}{bi_4+c}$	1	$\boldsymbol{e}_5^T$
10	μ	1	1	1	1	i_5	$-\boldsymbol{e}_5^T$



(a) $\log_{10}(\max|\mathcal{R}_n|)$ versus t

(b) $\log_{10}(\max(r(\mathbf{p}_n)))$ versus t

(c) $\log_{10}(N_e)$ versus t

(d) $\log_{10}(\#$ of nonzeros$)$ versus t

(e) $\log_{10}$(run times in seconds) versus t

(f) $\log_{10}(\sum_{n=1}^{N_e} |\mathbf{p}_n\mathbf{e} - \mathbf{p}_{n-1}\mathbf{e}|)$ versus t

Fig. 1. Various measures associated with BDF5 for the cascade model

Core i7 2.6 GHz processor with 16 Gigabytes main memory under Linux. We let $\mathbf{p}_n$ denote the transient probability distribution vector computed at time step n, $\max(|\mathcal{R}_n|)$ denote the maximum truncated state space size, $\max(r(\mathbf{p}_n))$ denote the maximum rank associated with compact solution vectors, N_e denote the total number of time steps taken up to t (if t is reached within 1,000 seconds), and $\sum_{n=1}^{N_e} |\mathbf{p}_n\mathbf{e} - \mathbf{p}_{n-1}\mathbf{e}|$ express the total state space truncation error, which has been shown to be in the same order as the relative error in the solution. We do not report the results with QT3 since they did not fare well. The results in Fig. 1 indicate that relative errors of at most 10^{-7} and 10^{-5} are obtained respectively with QTT and adaptive rank controlled HTD formats within 1,000 seconds in all problems. Furthermore, memory and time requirements of HTD_A are at least an order of magnitude better than those with QTT.

We depict in Fig. 2 the mean number of molecules when BDF5 with HTD_A is used to analyze the cascade model starting from the initial distribution $\boldsymbol{\pi}_0(0,0,0,0,0) = 1$. We remark that all results are obtained in at most 3,783 seconds with relative errors in $[5 \times 10^{-7}, 10^{-3}]$ using a maximum truncated state space size of $58,786,560$ and a maximum of $2,301,678$ nonzeros.

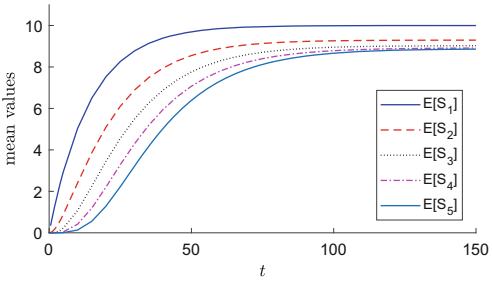


Fig. 2. Mean values with BDF5 using HTD_{A} for the cascade model

Acknowledgement. Part of this work is supported by the Alexander von Humboldt Foundation through the Research Group Linkage Programme. The research of M. Can Orhan is carried out during his PhD studies at Bilkent University and supported by The Scientific and Technological Research Council of Turkey under grant 2211-A. We thank the referees whose comments led to an improved manuscript.

References

1. Buchholz, P., Dayar, T., Kriege, J., Orhan, M.C.: On compact solution vectors in Kronecker-based Markovian analysis. Perform. Eval. **115**, 132–149 (2017)
2. CompactTransientSolver software (2017). http://www.cs.bilkent.edu.tr/~tugrul/software.html
3. Dayar, T.: Analyzing Markov Chains using Kronecker Products: Theory and Applications. Springer, New York (2012). https://doi.org/10.1007/978-1-4614-4190-8
4. Goutsias, J., Jenkinson, G.: Markovian dynamics on complex reaction networks. Phys. Rep. **529**(2), 199–264 (2013)
5. Hackbusch, W.: Tensor Spaces and Numerical Tensor Calculus. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28027-6
6. Hegland, M., Burden, C., Santoso, L., MacNamara, S., Booth, H.: A solver for the stochastic master equation applied to gene regulatory networks. J. Comput. Appl. Math. **205**(2), 708–724 (2007)
7. Kazeev, V., Khammash, M., Nip, M., Schwab, C.: Direct solution of the chemical master equation using quantized tensor trains. PLoS Comput. Biol. **10**(3), e1003359 (2014)
8. Khoromskij, B.N.: $O(d\log N)$-Quantics approximation of $N - d$ tensors in high-dimensional numerical modeling. Constructive Approximation **34**(2), 257–280 (2011)
9. Kressner, D., Tobler, C.: htucker – A Matlab toolbox for tensors in hierarchical Tucker format. Technical Report 2012–02, Mathematics Institute of Computational Science and Engineering, Lausanne (2012)
10. Orhan, M.C.: On the Numerical Analysis of Infinite Multi-dimensional Markov Chains. PhD Thesis, Department of Computer Engineering, Bilkent University, Ankara (2017)
11. Shampine, L.F., Reichelt, M.W.: The MATLAB ODE suite. SIAM J. Sci. Comput. **18**(1), 1–22 (1997)
12. Stewart, W.J.: Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Princeton (1994)

Logical PetriNet A Tool to Model Digital Circuit Petri Nets and Transform them into Digital Circuits

Christoph Brandau(✉) and Dietmar Tutsch(✉)

University of Wuppertal, Rainer-Gruenther-Str. 21, 42119 Wuppertal, Germany
{brandau,tutsch}@uni-wuppertal.de

Abstract. This paper introduces a new tool to design digital circuits with Petri nets. It describes the functionality of the tool Logical PetriNet (LPN). It discusses the modeling of Digital Circuit Petri Nets (DCPN). In addition, the possibilities of layout optimization of the DCPN are shown and export opportunities are described. The implemented analysis methods are described in a separate section.

1 Introduction

Nowadays, associated with the continuous increase of the complexity of digital circuits, new approaches and tools supporting their design must be introduced. In this paper we offer a tool to generate digital circuits from Petri net descriptions. These behavioral descriptions of digital circuits with Petri nets can be transformed into a hardware description language like VHDL. To reach this goal we extend the standard Petri nets with additional elements.

The motivation is to design a digital system using a graphical description. Petri nets provide many methods to analyze the modeled nets and which can be used to create safety relevant circuits. The modeled Petri nets can be transformed into combinatorial and sequential circuits. Sequential circuits will be split into synchronous and asynchronous circuits. The resulting circuit will be determined through the analysis in the transformation process. The modeled DCPN will be transformed entirely by using the reachability graphs and other strategies [1].

First, the Petri net model of Logical PetriNet is presented. The regular places are extended with two new types: To represent digital circuits with Petri nets, inputs and outputs must be defined, to get access to the circuit from outside. These are, firstly, the input places, which are represented as black circles with a green filling. On the other hand, these are the output places presented also as black circles, but with a yellow filling. DCPN uses immediate and timed transitions, where the timed transitions are deterministic. This transition type is implemented, because we want to describe switching times in digital circuits and to use these transitions to create synchronous circuits.

For readability a division of Petri nets into subnets is desirable to obtain hierarchical modeling. Regarding a connection of multiple items within a net,

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 317–321, 2018.
https://doi.org/10.1007/978-3-319-74947-1_25

it is a necessary that subnets provide multiple input and output arcs. For our approach, the existing subnet types are not sufficient because multiple inputs and outputs to and from the subnet are needed, so two separate types have been introduced. The subnets are divided in subplaces and subtransitions. In order to prepare the components of a subnet to be available to the outside a new element is introduced. It is called netconnector. For each input and output arc from or to the subnet a netconnector within the subnet is created. This element is given the name of the element, from which the arc leads into or out of the subnet.

Subplaces are used to represent a subnet whose arcs lead from or to transitions and subtransitions. Subplaces replace places in case that this part of the net should be more detailed. The symbol in the Petri net corresponds to a double-lined circle. As a further kind of subnet, Petri nets have been expanded with subtransitions. The presentation is a double-lined rectangle. Subtransitions have an additional property in DCPN. They can be used to save Petri nets or parts and to reuse them in newly modeled nets. For this purpose, we created a library to add created nets in a simple way to new nets. For a detailed description of the new Petri net type, the interested reader is referred to [1] and [2].

2 Implementation

The tool Logical PetriNet (LPN) is written in Java. The graphical user interface (GUI) shown in Fig. 1 is based on Java Swing components. The GUI and the functionality are separated, so the GUI can be adapted to a wide range of target domains, like other Petri net definitions.

The Digital Circuit Petri Net (DCPN) descriptions are saved in PNML [3] files. PNML uses XML to describe the elements of a Petri net. The generation or editing of Petri nets can be done easily by scripts or manually besides using the GUI. Logical PetriNet has a fully defined PNML schema. This allows consistency checks to loaded DCPN and the user can be pointed to wrongly modeled nets.

Figure 1 shows the graphical user interface from Logical PetriNet. The toolbar is placed on the right side and can be filled individually with the required elements. In the default setting all elements for modeling a DCPN are deposited. The Petri net elements placement can be optimized by using several algorithms from the KIELER framework [4], which we have modified to get better results.

At the bottom of the GUI a panel is located which displays notifications. These are used to display errors in modeling or when a loaded file does not exist. The menu is integrated as a ribbon and can be minimized if more space is required for modeling purposes.

In the left panel of the tool, an additional area can be displayed, which contains pre-assembled Petri nets. They can be added to the Petri net design in work when needed. Here also newly created nets can be saved in order to use them again in other DCPNs. It serves as a library for further designs.

The main area of the GUI includes the area for modeling a DCPN. Here, the area is divided into the current area for modeling on the left side. Elements can be placed, moved or deleted and can be connected with each other using arcs.

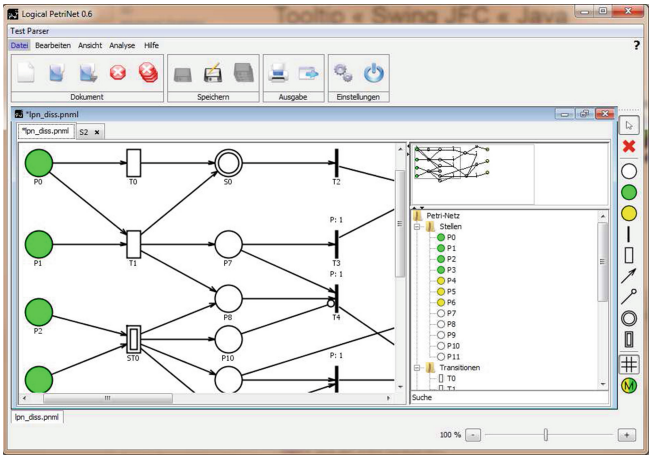


Fig. 1. GUI of the tool "Logical PetriNet" to create, simulate and analyze DCPNs

Newly modeled arcs will be checked regarding their validity, since the same types of items must not be connected. Furthermore, it is not allowed to connect places with subplaces or transitions with subtransitions. A minimap is placed in the upper right which can be used to navigate quickly and easily through the DCPN. Below the minimap a tree view with the elements of the current DCPN is displayed. The subdivision is done by element type. There is also a search field for locating elements faster in large DCPN. A zoom functionality and a minimap is implemented to deal comfortably with larger DCPN.

The DCPN can be exported as vector graphic (SVG), portable network graphic (PNG) or JPEG. Furthermore, an export to Latex is also implemented by using the package Tikz. In addition, reachability graphs and the results of the simulation can be exported. For this purpose all of the aforementioned image formats are available. The markings of the reachability graph and the results of the simulation can be further exported to Latex and comma-separated values (CSV).

3 Functionality

This tool can be used to design DCPN in graphical description. A zoom functionality and a minimap is implemented to deal comfortably with larger DCPN. In addition, the DCPN can be automatically rearranged graphically wherein input places are located on the left and output places are located on the right.

As a further component, the so-called token game is implemented to test the behavior of a modeled DCPN. This simulates the behavior of the net by firing enabled transitions. The token game can be started from the current marking given by the user. This game can be executed manually or automatically. In manual simulation, all enabled transitions flash and can be double clicked. The new

marking will be calculated and then the newly enabled transitions will be identified. Then, the user can fire one of the next enabled transitions. The automated token game is separated into full and partial automation. Partial automation fires transitions automatically until more than one transition is simultaneously enabled. Then the user has to select which transition fires. In fully automatic mode the firing of the transitions is carried out automatically. When multiple transitions are enabled, a firing transition is chosen with the built-in random function from Java. The probability of every single firing transition depends on the random function. The random function can be changed simply by editing one line in the source code. The token game will be used for testing the modeled DCPN by the designer. In the transformation process, a detailed analysis takes place if the net has more than one active transition. This case will be dealt with in the general structure analysis step (see below).

Furthermore, a reachability graph for the current DCPN can be determined. This graph is the main ingredient for the transformation. The graph describes all reachable markings from an initial marking. This graph will be created by firing all enabled transitions individually. Then, the new marking is put in the reachability graph, if this marking does not yet exist in the graph. An arc between the old and the new marking will be created. This routine is running until all enabled transitions from every marking in the graph are fired. The created reachability graph can now be used for further analysis or it can be used for the transformation into a digital circuit.

The transformation process from a DCPN to a digital circuit can take place in two different ways. First, the transformation of the entire DCPN can be performed by transforming each subnet of the DCPN by starting with the innermost subnets. This is because the above lying nets in the hierarchy use the circuits of the internal nets. On the other hand, a breakup of the hierarchy can be performed to produce an overall circuit from the entire DCPN. To resolve the subnets, they will be reintegrated into the main net.

The modeling of the proposed system takes place as a DCPN. Here, the properties from [1] are used. This is followed by the verification of the model to detect erroneous or contradictory characteristics of the DCPN and to identify these faults in the net. Further strategies will be used to identify potential improvements in the net. Elements without an impact to the behavior of the circuit will be removed in this optimization step. Furthermore, this step also applies to the detection and grouping of redundant elements.

The next step is a general structural analysis, in which the breakdown by circuit type must be carried out. These types are combinational logic and sequential logic. Faulty modeling can also be detected in this step, because a comprehensive analysis of the system takes place. An incorrect modeled net can be a net with two conflicting transitions for example if the conflict leads to different states in the DCPN. This step is the central step in this chain, because the transformation starts and will be finished within the next step. The analysis consists of strategies to create reachability graphs, input driven reachability graphs [1], finding cycles in the net and the graphs, checking for termination and defined states.

More detailed information to the strategies can be found in [1]. The transformation of DCPN to the hardware description VHDL follows in which the interface of the complete net is created. The interface from a DCPN is represented by all input and output places. They will be used as inputs and outputs of the resulting circuit. Furthermore, the description of the behavior or structure is created depending on the detected circuit type. This will be followed by the adaptation of the target architecture. Examples for the resulting digital circuits are shown in [1]. After the transformation is completed, a validation of the circuit can be started. It will be achieved by a comparison of the simulation results between the DCPN model and the resulting digital circuit. The simulation of the DCPN is carried out by an event-based simulation because every firing transition is a single event.

4 Conclusion

This paper presented Logical PetriNet, a Digital Circuit Petri Net modeling and simulation tool. Places, transitions, subplaces, and subtransitions can be modeled with LPN. These components can be linked with arcs to create working systems.

Our current research is aimed at extending the tool to establish more strategies in the tool. We want to implement a method to transform existing digital circuits into DCPN to analyze them with the known methods. The target is to validate all methods implemented in the LPN. For this, we transform the digital circuit into a DCPN and use the given strategies to rebuild a digital circuit. The behavior of the previous circuit will be compared to the transformed circuit.

The tool is tested with nets up to 500 elements in total (places, transitions) and with up to 750 arcs. For bigger nets, there is the possibility to split the net into subnets. These subnets are analyzed separately, so larger nets can be analyzed. To the best of the authors' knowledge, Logical PetriNet is the first tool that can handle Digital Circuit Petri Nets with a convenient graphical user interface.

References

1. Brandau, C., Tutsch, D.: A new method to transform petri nets to digital circuits using input-driven reachability graphs. In: ESM 2017 - 31th European Simulation and Modeling Conference, pp. 301–307 (2017)
2. Brandau, C., Potthoff, N., Tutsch, D., Lepich, T.: Digital circuit petri nets: a new petri net type to describe and transform digital circuits for product safety engineering. In: The 6th IEEE International Conference on Consumer Electronics - Berlin, pp. 271–275 (2016)
3. PNML - Petri Net Markup Language. www.pnml.org
4. Christian-Albrechts-Universität zu Kiel, KIELER Project. rtsys.informatik.uni-kiel.de/confluence/display/KIELER

ClassCast: A Tool for Class-Based Forecasting

Florian Heimgaertner(✉), Thomas Sachs, and Michael Menth

Chair of Communication Networks, University of Tuebingen,
Sand 13, 72076 Tuebingen, Germany
{florian.heimgaertner,menth}@uni-tuebingen.de,
thomas.sachs@student.uni-tuebingen.de

Abstract. We present ClassCast, a tool for class-based forecasting. It partitions an input time series into class-specific time series. It uses conventional prediction methods for each of these time series and maps class-specific values to classified future time indices. It is a simple means to account for non-linear factors in time-series for the purpose of prediction.

Keywords: Forecasting · Prediction · Tool

1 Introduction

Various approaches have been discussed for load forecasting in the domain of electrical energy [1,2]. Especially methods based on machine learning, such as artificial neural networks [3] or support vector machines [4], have attracted interest. However, when looking at time series y_j of energy consumption of administrative buildings, a clear dependency on working and non-working days and on the weather can be observed. Therefore, a simple forecast method accounting for these influencing factors can outperform sophisticated forecast methods neglecting that influence. This basic forecasting concept is known as similar-days method [5]. A very simple forecast is the average energy consumption specific for working and non-working days. The average prediction may be further adapted to cold and warm working and non-working days. We propose class-based forecasting as a generalization of the similar-days method. Cold/warm and working/non-working are an example for a classification c_j of the time index j. Besides basic calendar information, additional data in the time series that is known for both the past and the future can be used for classification, e.g., scheduled operating times of specific devices. Essentially, we compute class-specific means $\overline{Y}(c)$ with $c \in \mathcal{C}$ and $\mathcal{C}$ being the class range. These class-specific means may be used for prediction $y_j = \overline{Y}(c_j)$ of future time indices j which are classified as c_j.

This class-based forecast method can be adapted to more complex forecasting methods. We exemplify this for linear regression. It predicts future values y_j with a dependence on a regressor x_j: $y_j = \beta_0 + \beta_1 \cdot x_j$. Class-specific forecast implies that linear regression is applied to a class-specific subsets of the time series with $(y_j^c) = (y_j)_{j:c_j=c}$ for $c \in \mathcal{C}$. Based on those time series, class-specific

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 322–326, 2018.
https://doi.org/10.1007/978-3-319-74947-1_26

forecast models $\{(\beta_0^c, \beta_1^c) : c \in \mathcal{C}\}$ are derived for each class $c \in \mathcal{C}$ which predict $y_j = \beta_0^{c_j} + \beta_1^{c_j} \cdot x_j$ for $j : c_j = c$.

We have developed ClassCast [6], a tool for class-based forecast supporting prediction based on mean values and linear regression.

This work is structured as follows. Section 2 provides a high-level description of ClassCast. Section 3 explains the concept of the tool and describes the classification and forecasting mechanism. In Sect. 4, we present implementation details and the user interface of ClassCast. Section 6 concludes the paper.

2 Tool Description

The input of ClassCast is an annotated time series y_j. The data series consists of past values of the *dependent variable*, i.e., the variable to be forecasted. Annotations are values of one or multiple *independent variables* x_j^i, $0 \leq i \leq n$ that are supposed to influence the values of the dependent variables. An example for a dependent variable is the total energy consumption while independent variables could be day of week, time of day, outside temperature, or the state of a monitored device.

The second part of the input is the future time series. The future time series consists of values for the independent variables but does not contain values for the dependent variable. The independent variables of the future time series are used to forecast values of the dependent variables using the approaches presented in Sect. 3.

Figure 1 depicts the operation of ClassCast. Past time series $y_j x_j^i$ and future time series x_j^i are supplied to the tool. For each independent variable of the input time series, the user selects whether it is discarded, used as classifier, or (if applicable) as regressor. This selection determines the forecasting method. If one or multiple independent variables are selected as regressors, linear regression is used for forecasting. Otherwise, class-specific averages are used for forecasting.

The future values for the dependent variables y_j are computed based on the combination of values for the independent variables x_j^i of the future time series. The forecast can be saved to a file or visualized for further analysis.

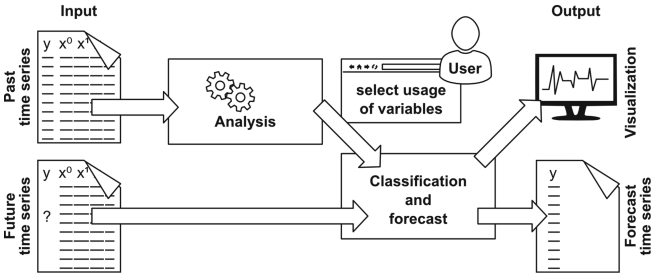


Fig. 1. Operation of ClassCast.

3 Concept

The past data set for the forecast consists of the time series $(y_j)_{j=s,\dots,-1}$ where y_j are the dependent variables and j are the time series indices. The index $s < 0$ denotes the start index of the time series and is negative as it relates to the past. Additionally, the input data set contains n series of independent variables $\left(x_j^i\right)_{j=s,\dots,-1}^{i=0,\dots,n-1}$. The future data set starts at the index 0 and contains values for the independent variables $\left(x_j^i\right)_{j=0,\dots,m}^{i=0,\dots,n-1}$ where m denotes the forecasting horizon. ClassCast uses information from the past and future data set to compute the forecast time series $(y_j)_{j=0,\dots,m}$.

The basic concept of ClassCast is classifying time series entries according to the values of the independent variables. The independent variables are elements of an n-dimensional annotation space $\mathcal{X} = \mathcal{X}_0 \times \ldots \times \mathcal{X}_{n-1}$. A function $f : \mathcal{X} \mapsto \mathcal{C}$ maps the annotations to a q-dimensional class space $\mathcal{C} = \mathcal{C}_0 \times \ldots \times \mathcal{C}_{q-1}$. The mapping function f of our tool is limited in the way that any annotation dimension is mapped to at most one class dimension or not used for the mapping. Furthermore, any class dimension is determined by exactly one annotation dimension so that $q \leq n$ holds. We use the mapping function f to calculate class-specific model parameters and to determine the class of future independent variables x_j as a base for class-based forecasting. Class-specific means are calculated by

$$\overline{Y}(c) = \frac{1}{|\mathcal{J}_c|} \cdot \sum_{j \in \mathcal{J}_c} y_j, \text{for } c \in \mathcal{C}, \mathcal{J}_c = \{j : f(x_j) = c\}$$

If at least one independent variable is selected as regressor, the past time series is partitioned into class-specific subsets $(y_j^c) = (y_j)_{j:f(x_j)=c}$. Linear regression can be used to compute regression parameters β_0, β_1 for each class-specific partition.

Independent variables can be given as symbolic or numeric values. The scale of measure [7] for the independent variables is derived from the input data type. Symbolic values (e.g., "Monday", "Tuesday",...) are interpreted as nominal scale, integers as ordinal scale, and real numbers as interval scale. By default, ClassCast uses independent variables on nominal and ordinal scale as classifiers and independent variables on interval scale as regressors. If an independent variable containing real numbers is manually selected as classifier, classes need to be defined based on intervals.

4 Implementation

ClassCast is implemented in Java and features a graphical user interface (GUI) based on the JavaFX [8] framework. Input data series are processed in comma-separated values (CSV) format. The tool automatically detects separators (comma, semicolon, tab), line break encoding, and input data types.

Values for independent variables can be given as strings, integer numbers, or floating point numbers. Strings are interpreted as nominal scale, integers as

ordinal scale, and floating point numbers as interval scale. The first line of the input data file is interpreted as heading and is used for visualization only.

If the past data set is insufficient for forecasting, ClassCast presents a dialogue to the user for selecting an independent variable to discard. E.g., if the past data series is a time series recorded in winter, a forecast for summer cannot be computed using class-specific means without discarding the independent variables representing the weather.

ClassCast can be used both as an interactive tool with a GUI and as a non-interactive command line tool. In non-interactive mode, a CSV file is given as command line argument. Configuration like interval size for floating point classifiers can be given as option. The forecast data series is written to standard output in the same CSV format as used for the input data series.

5 Forecast Validation

ClassCast implements a very generic forecast approach. The forecast quality depends on the availability of sufficient data for the input time series, the existence of a strong dependence of dependent variables y_j on independent variables x_j^i, and the selection of appropriate classifiers. Therefore, ClassCast includes an interactive validation feature. For validation, the input data series is split in two parts. The first part is used to forecast the second part of the data series. The forecast can be visually compared to the actual measurement values and the mean square error can be computed to quantify the forecast quality.

Figure 2 shows a screenshot of the interactive validation, with an electrical load time series sampled from a school building in Germany. The forecast is computed as class-specific means with day of week and time of day as classifiers.

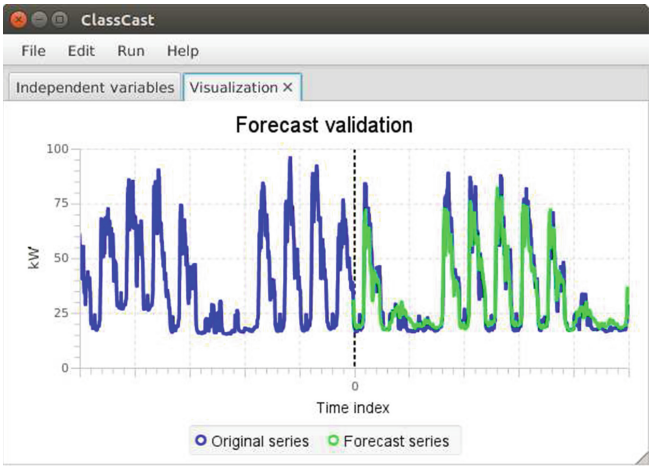


Fig. 2. Validation of an electrical load forecast. (Color figure online)

The measured loads are plotted in blue, the forecast for the second part of the time series is plotted in green.

6 Conclusion

In this paper, we presented ClassCast, a simple tool for class-based forecasting. ClassCast classifies entries of time series according to the values of the independent variables and predicts future values for dependent variables using class-specific means or linear regression. The conception of ClassCast was motivated by energy load forecasting. However, ClassCast is agnostic to input data semantics. Therefore, it can be applied for forecasting time series in other domains under the following two conditions. First, time indices can be classified based on annotated information (independent variables). Second, this annotated information has a major impact on the values of interest (dependent variables).

Acknowledgement. The research leading to these results received funding from the German Ministry for Economic Affairs and Energy under the ZIM programme (Zentrales Innovationsprogramm Mittelstand), grant reference no. 16KN039521. The authors alone are responsible for the content of this paper. The authors thank Prof. Joachim Grammig for fruitful discussions and helpful advice.

References

1. Feinberg, E.A., Genethliou, D.: Applied Mathematics for Restructured Electric Power Systems: Optimization, Control, and Computational Intelligence. Springer, New York (2005). https://doi.org/10.1007/b101578. pp. 269–285
2. Campbell, P.R., Adamson, K.: Methodologies for load forecasting. In: International IEEE Conference on Intelligent Systems (2006)
3. Park, D.C., El-Sharkawi, M., Marks, R., Atlas, L., Damborg, M.: Electric load forecasting using an artificial neural network. IEEE Trans. Power Syst. **6**(2), 442–449 (1991)
4. Chen, B.J., Chang, M.W., et al.: Load forecasting using support vector machines: a study on EUNITE competition 2001. IEEE Trans. Power Syst. **19**(4), 1821–1830 (2004)
5. Mu, Q., Wu, Y., Pan, X., Huang, L., Li, X.: Short-term load forecasting using improved similar days method. In: IEEE PES Power and Energy Engineering Conference (APPEEC) (2010)
6. Heimgaertner, F., Sachs, T., Menth, M.: ClassCast (2018). https://github.com/uni-tue-kn/classcast
7. Stevens, S.S.: On the theory of scales of measurement. Science **103**(2684), 677–680 (1946)
8. Oracle Corporation: JavaFX API Documentation (2015). https://docs.oracle.com/javase/8/javafx/api/

Collider – Parallel Experiments in Silico

Dimitri Scheftelowitsch(✉)

Informatik IV, TU Dortmund, Dortmund, Germany
dimitri.scheftelowitsch@cs.tu-dortmund.de

Abstract. Large-scale software experiments are a ubiquitous feature of research. For example, performance evaluation of algorithms implies testing said algorithms on a large number of test cases. We provide a software framework which helps performing experiments on large parameter spaces, benefits from multi-core architectures, and saves generated results in a machine-readable format for future post-processing.

Keywords: Performance evaluation · Experiments · Software tools

1 Introduction

Modern research relies on large amounts of computing power in order to perform experiments on and with software. The SETI@home project [2] is a famous example for parallelizing computational tasks, but also algorithmic research requires to test practical performance of algorithm design ideas by benchmarking said algorithms on a set of instances. Another example are software simulations [6], where several different runs of a model have to be evaluated. In most cases this is done via a loop over the space of inputs, application of several successive processing stages, and, finally, aggregation of the produced data.

The increasing availability of multi-core computers (and increasing parallel processing power in hardware installations) motivates parallelizing the experiments in order to utilize the available hardware efficiently and perform more experiments in less time, as the different experiments can be run independently from each other. However, even if the programming language allows to parallelize loops in a machine-independent way, writing ultimately the same infrastructure code which implements loops and saving partial results for every experiment setup makes the experiment code "write-only" with the obvious downsides.

Here, we propose a tool that allows one to define the individual stages of a software experiment, run it, utilizing all available cores, and save intermediate and final results for future post-processing, decoupling the experiment from the software which is being experimented on. In detail, the user provides a *parameter space* of *values*, i.e., objects that are to be processed, a set of individual *stages* which describe the units of computation such as instance generation, application of a specific data analysis method etc., and, optionally, a post-processing routine which analyzes the resulting data set.

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 327–331, 2018.
https://doi.org/10.1007/978-3-319-74947-1_27

The tool itself is written in Python [5] with the help of the Pandas [1] library. The main purpose behind COLLIDER is to provide a maintainable infrastructure framework that takes care of the experimental process.

1.1 Previous Work

There exist several tools and frameworks which serve similar purposes.

- *jug* [3] is a Python tool and framework that allows one to parallelize tasks for a large number of inputs. Similar to COLLIDER, *jug* has the ability to re-run sub-sequences of tasks on a given subset of the parameter space. In contrast to COLLIDER, *jug* is Python-native and does not natively provide means to call other executables (which, however, can be implemented by the user).
- *BOINC* [2] is a general-purpose framework for distributed computing in large-scale networks. Its most known application is the SETI@home project in which any volunteer can provide her computing power for the search of extraterrestrial life in astronomic observation data. BOINC schedules and distributes computational tasks over a variety of client platforms, however, it requires a dedicated server installation and allows one to use a global-scale infrastructure of computers with limited access rights. COLLIDER is a lightweight alternative for less computation-intensive tasks which can be performed on one, or, in future, on a limited number of computers which the experiment owner can access at least with user-grade privileges.
- For large-scale computing systems, *grid schedulers* [4] such as *Open Grid Scheduler* which manage computing tasks exist. They provide multi-user features at the expense of, again, more infrastructure and system administrator-grade access rights.

2 Architecture

COLLIDER provides an infrastructure to perform software experiments, or, in general, apply several successive *stages*, i.e. atomic units of computation on a space of parameters $\mathcal{C}$, while additionally saving intermediate results in order to be able to rerun (or restart from) a specified stage. In order to run COLLIDER, the user must specify a parameter space by providing its individual components and a list of stages $f_1, \dots, f_m$. For example, if she wants to apply the experiment on the Cartesian product space $\mathcal{C} = \{1, \dots, 20\} \times \{\text{red}, \text{green}, \text{blue}\}$, then it is sufficient to specify the individual components, $\{1, \dots, 20\}$ and $\{\text{red}, \text{green}, \text{blue}\}$ along with functions $f_1, \dots, f_m$ which operate on $\mathcal{C}$. COLLIDER then converts the given sets into the desired parameter space and initializes the computation.

COLLIDER uses n available CPU cores for the following classes of subprocesses.

- $n - 2$ *workers* which execute the stages,
- 1 *queue manager* which schedules tasks,
- 1 *result manager* which stores the results of the individual stages.

The multiprocessing architecture is centered around three queue mechanisms that govern the execution of the individual stages on values. The first queue, Q_v, contains the jobs that have to be done. A job is a tuple (v, i, s) containing a *value* $v \in \mathcal{C}$, output from previous stages for this value s, and the index of the next stage i. The second queue, Q_r, contains the job results, that is, tuples (v, i, r) that contain a value $v \in \mathcal{C}$, a stage i, and the results $r = f_i(v)$ of the computation. The purpose of Q_r is to inform the queue manager that a task is done. The third queue, Q_s, contains results (in the same format as in Q_r) that have to be stored into a table-like Pandas [1] data frame.

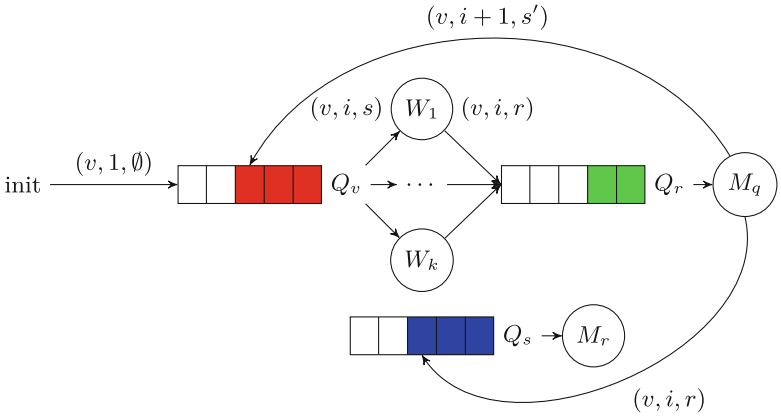


Fig. 1. Schematic view of the queues and the processes involved in the computation (Color figure online)

The detailed function of the queues is explained as follows. In the beginning, the main process pushes all tasks $(v, 1, \emptyset)$ for the first stage into Q_v. The workers compete for the next task (v, i, s) in Q_v and execute the computation or, if a result already exists and a re-computation is not required, look up the precomputed result; when the computation is complete, the worker writes the results $(v, i, s' = s \uplus r)$ to Q_r. The queue manager takes the next tuple (v, i, s') from Q_r, pushes the job for the next stage $(v, i+1, s')$ onto Q_v and the results (v, i, r) onto Q_s. In parallel to that, the result manager reads items (v, i, r) from Q_s and stores them into a Pandas data frame.

The complete process is visualized in Fig. 1. Individual processes are displayed as circles. Workers are designated by $W_1, \ldots, W_k$ (where $k = n - 2$), the queue and the result managers are designated by M_q resp. M_r.

2.1 API

The COLLIDER library provides an API that enables the user to specify the individual stages of experiment, and therefore, the individual measurements.

In order to use the API, the user must import the `collider` Python module and instantiate subclasses of the `Stage` class. The list of these instances can then be passed on to the `run_experiments` function which takes the following arguments and runs then the experiments in the provided order.

- A dictionary that describes and names the dimensions of the parameter space,
- a list of stages, i. e., instances that implement the `Stage` interface,
- a history of previous results (possibly empty),
- a predicate that tells for which stages and values a re-execution is required.

`run_experiments` returns a Pandas data frame which can be analyzed further.

The API provides several ready-made `Stage` subclasses that implement calls to foreign executables and several predicates that allow one to re-execute an individual stage, all experiments, all experiments beginning with some specified stage, or only those experiments which have not yet been run.

2.2 Executable

COLLIDER also provides an executable that runs experiments operating from a description in a configuration file. In this file, the user has to provide the individual stages by naming the executables that have to be called and their arguments (which may be the results of previous stages), the parameter space, and, optionally, a Python function `postprocess(data, values)` that accepts the Pandas data frame with the collected data and the dictionary of input values. The syntax of the configuration file is identical to Python syntax.

3 Conclusion and Outlook

This work presents the COLLIDER tool for describing and running experiments in silico. Currently, the tool is capable of running an experiment for all input values on all available CPU cores on a computer and save the results for further analysis.

We believe there are several direction to improve COLLIDER. Up to now, COLLIDER runs only on one machine. This can be improved on by introducing clients which can process jobs from the queue on other machines on the network, parallelizing the task further. Another direction of improvement is a more complex dependency tracking mechanism: Re-evaluation should automatically force additional re-evaluation only of dependent stages.

References

1. Pandas: Python Data Analysis Library. Online (2012). http://pandas.pydata.org/
2. Anderson, D.P.: BOINC: a system for public-resource computing and storage. In: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, GRID 2004, pp. 4–10. IEEE Computer Society, Washington (2004)

3. Coelho, L.P.: Jug: a task-based parallelization framework (2008). https://jug.readthedocs.io/en/latest/. Accessed 18 Sep 2017
4. Gradwell, P.: Overview of Grid Scheduling Systems. Department of Computer Science, University of Bath. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93.150&rep=rep1&type=pdf
5. Python Software Foundation: Python 3 documentation (2001). https://docs.python.org/3/. Accessed 6 Nov 2017
6. Varga, A., Hornig, R.: An overview of the OMNeT++ simulation environment. In: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems and Workshops, Simutools 2008, pp. 60:1–60:10. ICST, Brussels, Belgium (2008)

FunSpec4DTMC – A Tool for Modelling Discrete-Time Markov Chains Using Functional Specification

Frederik Hauser(✉), Dominik Krauß(✉), and Michael Menth(✉)

Chair of Communication Networks, University of Tuebingen,
Sand 13, 72076 Tuebingen, Germany
{frederik.hauser,menth}@uni-tuebingen.de,
johannes-dominik.krauss@student.uni-tuebingen.de

Abstract. We present a tool for the analysis of finite discrete-time Markov chains (DTMCs). As a novelty, the tool offers functional specification of DTMCs and implements forward algorithms to compute the stationary state distribution x_s of the DTMC or derive its transition matrix P [19]. In addition, we implement nine direct and indirect algorithms to compute various metrics of DTMCs based on P including an algorithm to determine the period of the DTMC. The tool is intended for both production purposes and as platform for teaching the functional specification of DTMCs. It is published under GPLv3 [3] on Github [2].

1 Introduction

Discrete-time Markov chains (DTMCs) are a widely applied concept for system modelling. Typically, DTMCs are defined by a stochastic matrix P that holds probabilities for transitions among system states. The vector x_n describes the state distribution of a system after n transitions. Consecutive distribution vectors x_n are calculated by $x_{n+1} = x_n \cdot P$. The stationary state distribution fulfills $x_s = x_s \cdot P$. It reflects the average state distribution after multiple transitions and is a useful base for the derivation of further specific performance metrics. Theoretical background of DTMCs is described in [21,22]. There are many scientific analysis tools [6,13,14,16–18,23] and libaries for the field of teaching [6,20]. All utilize the transition matrix P as the base for analysis.

In this work, we present a tool for modelling DTMCs with a finite state space using the novel functional specification suggested in [19]. We introduce the functional specification by an example. We consider a two-dimensional constraint random walk on a grid with coordinates (a, b) and integer values $a \in \{A_{min}, ..., A_{max}\} = \mathcal{A}$ and $b \in \{B_{min}, ..., B_{max}\} = \mathcal{B}$. The walk starts at position (A_{min}, B_{min}). In any transition, the position may change horizontally by integer values $\mathcal{H} = \{H_{min}, ..., H_{max}\}$ and vertically by $\mathcal{V} = \{V_{min}, ..., V_{max}\}$, all with equal probability. We consider the system with $A_{min} = B_{min} = 1$, $A_{max} = B_{max} = 3$, $H_{min} = V_{min} = -1$, and $V_{max} = H_{max} = 1$. It can be modelled by a two-dimensional state space $\mathcal{X} = \mathcal{A} \times \mathcal{B}$ and a factor space

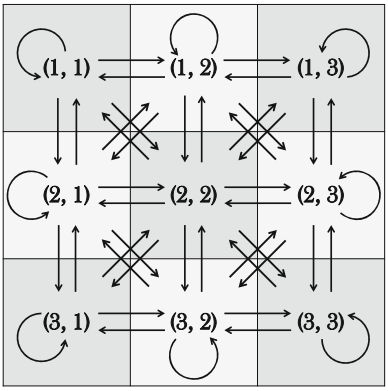
R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 332–337, 2018.
https://doi.org/10.1007/978-3-319-74947-1_28

$\mathcal{Y} = \mathcal{H} \times \mathcal{V}$ with $\mathcal{H} = \mathcal{V} = \{-1, 0, 1\}$. The random variables $X_n = (A_n, B_n) \in \mathcal{X}$ describe the position of the walk after n transitions. Given a random factor for the move $Y = (H, V) \in \mathcal{Y}$, the next position $(A_{n+1},\ B_{n+1}) \in \mathcal{X}$ of the walk is determined by

$$A_{n+1} = min(A_{max}, max(A_{min}, A_n + H)) \tag{1}$$
$$B_{n+1} = min(B_{max}, max(B_{min}, B_n + V)). \tag{2}$$

This is a stochastic recursive equation that constitutes the state transition function of the system $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{X}$. Together with the distribution y of the factor space $\mathcal{Y}$ this function constitutes a DTMC [19]. A so-called forward algorithm may be used to compute consecutive state distributions x_n based on this description without the use of a transition matrix P. The transition matrix P can be derived by a similar algorithm so that other analytical methods can be applied to it.

Figure 1(a) shows all states of the random walk with potential transitions. Figure 1(b) illustrates the state transition matrix. The stationary state distribution yields $x_s(i) = \frac{1}{9}$ for any $i \in \mathcal{X}$.



(a) State space and potential transitions.

	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,3)	(3,1)	(3,2)	(3,3)
(1,1)	$\frac{4}{9}$	$\frac{2}{9}$	0	$\frac{2}{9}$	$\frac{1}{9}$	0	0	0	0
(1,2)	$\frac{2}{9}$	$\frac{2}{9}$	$\frac{2}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	0	0	0
(1,3)	0	$\frac{2}{9}$	$\frac{4}{9}$	0	$\frac{1}{9}$	$\frac{2}{9}$	0	0	0
(2,1)	$\frac{2}{9}$	$\frac{1}{9}$	0	$\frac{2}{9}$	$\frac{1}{9}$	0	$\frac{2}{9}$	$\frac{1}{9}$	0
(2,2)	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
(2,3)	0	$\frac{1}{9}$	$\frac{2}{9}$	0	$\frac{1}{9}$	$\frac{2}{9}$	0	$\frac{1}{9}$	$\frac{2}{9}$
(3,1)	0	0	0	$\frac{2}{9}$	$\frac{1}{9}$	0	$\frac{4}{9}$	$\frac{2}{9}$	0
(3,2)	0	0	0	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{2}{9}$	$\frac{2}{9}$	$\frac{2}{9}$
(3,3)	0	0	0	0	$\frac{1}{9}$	$\frac{2}{9}$	0	$\frac{2}{9}$	$\frac{4}{9}$

(b) State transition matrix P.

Fig. 1. Random walk example.

The functional specification might appear more complex, but provides many benefits. First, it allows intuitive modelling of systems with event-triggered state transitions. Events are modelled by factors whose probabilities are described by the factor distribution. The system's state transition in case of particular events is described by the transition function. Therefore, the functional specification is close to the system's behaviour which facilitates modelling of complex systems with even multi-dimensional state spaces. Second, the functional specification allows the modelling of very large DTMCs. The conventional specification requires the transition matrix P which scales quadratically with the number of states. For very large DTMCs, the resulting size of P may be so

large (multiple Terabytes) that DTMC analysis based on P may become infeasible. Sparse matrix representation may reduce memory requirements, but its effectiveness depends on the specific use case. With the functional specification, the transition matrix is not needed for the analysis of the DTMC and memory requirements are reduced to the state and factor distribution. The memory requirement for the state transition function is generally small. In [19] further optimization methods are described to speed up the convergence of the consecutive state distributions based on the functional description.

FunSpec4DTMC implements the functional specification and the forward algorithm to calculate consecutive state distributions x_n and the transition matrix P. Besides, the tool offers various direct and iterative computation methods to calculate metrics for DTMCs that are based on P. In particular, the period of finite DTMCs can be derived and methods for output visualization are provided.

The paper is structured as follows. In the next section, we present the core idea and features of FunSpec4DTMC. Section 3 describes the architecture and implementation.

2 Tool Description

FunSpec4DTMC consists of a library implementing the functionality and a graphical user interface (GUI) that allows users to analyse DTMCs in an interactive process. The four phases of FunSpec4DTMC's analysis process for DTMCs are depicted in Fig. 2.

In the first phase (I), the DTMC model is defined by the user. DTMCs can be either defined in a GUI-based input dialogue or imported from JSON project files. As an example for intuitive modelling of DTMCs, our tool offers a system specification dialogue for a $GI^{[GI]}/D/1 - Q_{max}$ queuing system.

In the second phase (II), the DTMC model input is parsed and validated against mistakes in the specification, e.g., state probability vectors that do not sum up to 1. In addition, aspects of the DTMC model such as the initial state vector can be visualized.

In the third phase (III), metrics for DTMCs are calculated. If the DTMC is defined in a conventional way using the transition matrix P, multiple direct and iterative computation algorithms can be applied. The former are accurate and fast but require a large amount of memory. Examples are the Gaussian elimination algorithm and the inverse iteration. The latter requires less memory but lots of iterations to compute the stationary state distribution with high accuracy. The tool offers the following iterative methods to approximate the stationary state distribution x_s:

- DTMC random walk (simulation)
- calculation of the limiting distribution ($\lim_{n\to\infty} x_n$) (applicable to aperiodic DTMCs)
- matrix powering ($\lim_{n\to\infty} P^n$) (applicable to aperiodic DTMCs and to DTMCs with a period of 2^n, $n \in \mathbb{N}_0$)

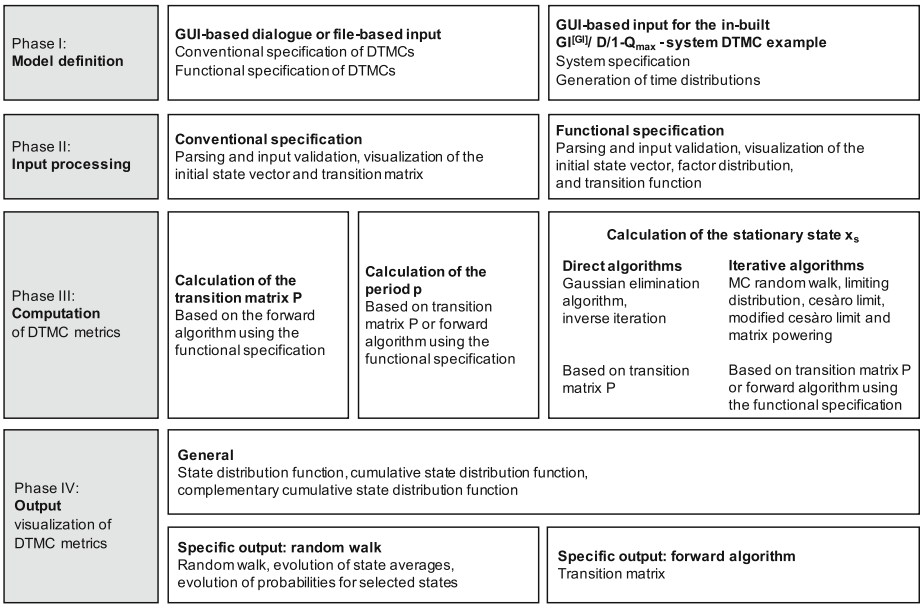


Fig. 2. Four phases of FunSpec4DTMC's analysis process for DTMCs.

- calculation of the Cesàro limit $(\lim_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} x_i)$
- modified calculation of the Cesàro limit $(\lim_{n\to\infty} \frac{1}{p} \sum_{n\leq i<n+p} x_i)$ as introduced in [19]. To that end, the tool analyzes transition structures of the DTMC and computes its period p.

With a functional specification of a DTMC, the tool computes consecutive state distributions x_n and DTMC simulations without the state transition matrix P and uses for this purpose the forward algorithm or just the state transition function f, respectively. Moreover, the state transition matrix P can be derived from the functional specification based on another forward algorithm [19].

In the fourth phase (IV), the output of the DTMC analysis can be visualized. That includes the visualization of general metrics, e.g., the stationary state distribution, and the visualizations of particular computation algorithm specifics, e.g., the random walk.

3 Architecture and Implementation

The architecture of FunSpec4DTMC is based on the model-view-controller (MVC) pattern that separates its functionality from the GUI. We designed an object-oriented class hierarchy and applied design patterns, e.g., the observer or strategy pattern [15], and language constructs, e.g., the signal-and-slot approach [10].

We chose Python in version 3.6.3 [8] as programming language. We use Matplotlib [4] to generate plot figures and SciPy [11] to import common distributions. To apply SciPy's continuous distributions on DTMCs, we implemented mechanisms for discretization and normalization. We implemented the GUI using PyQt5 [7], the Python bindings to the widely applied GUI framework Qt [9]. It is platform-independent and allows the creation of more advanced graphical surfaces compared to simple approaches like Tkinter [12]. Python is an interpreted programming language, i.e., source code is translated at runtime. To compensate performance drawbacks, computational-intense functions are implemented in C and called at runtime. External libraries like NumPy [5] adopt this principle and use efficient implementations, e.g., for vector-matrix and matrix-matrix multiplications. We used the Cython [1] extension to implement the forward algorithm's interleaved loops and the model-specific transition functions. Cython allows to implement CPU-intensive modules as C-extensions in a Python-like syntax with additional annotations such as static type declarations. Afterwards, the Cython source code is transformed into C code, compiled, and called at runtime from within Python. Our tool automatically integrates the model-specific transition function defined by the users into the forward algorithm that is a static part of the tool. Analyzing large DTMCs is limited by the memory on the host system. The functional specification may mitigate but not solve the memory problems that arise with a large number of states. We applied the memory-to-disk swapping mechanism of NumPy so that computing data is stored in a file on the hard disk which can be accessed in small segments.

Within the tool's GUI, users can define multiple projects. For each project, multiple DTMCs can be specified either in the functional or conventional specification within an interactive dialogue or by importing a JSON file. DTMC models and the output of the calculation algorithms can be visualized in multiple plot views. Projects can be stored as files in JSON format and be imported.

References

1. Cython: C-Extensions for Python. http://cython.org/. Accessed 6 Nov 2017
2. Github: FunSpec4DTMC. https://github.com/uni-tue-kn/funspec4dtmc. Accessed 6 Nov 2017
3. GNU General Public License 3. https://www.gnu.org/licenses/gpl-3.0.en.html. Accessed 6 Nov 2017
4. Matplotlib 2.1.0. https://matplotlib.org/. Accessed 6 Nov 2017
5. NumPy - Scientific Computing with Python. http://www.numpy.org/. Accessed 6 Nov 2017
6. PyPI: discreteMarkovChain. https://pypi.python.org/pypi/discreteMarkovChain. Accessed 6 Nov 2017
7. PyQt5. http://www.numpy.org/. Accessed 6 Nov 2017
8. Python 3.6.3. https://www.python.org/downloads/release/python-363/. Accessed 6 Nov 2017
9. Qt. https://www.qt.io/. Accessed 6 Nov 2017
10. Qt5: Signals & Slots. http://doc.qt.io/qt-5/signalsandslots.html. Accessed 6 Nov 2017

11. SciPy. https://www.scipy.org/. Accessed 6 Nov 2017
12. tkinter. https://docs.python.org/3.6/library/tkinter.html. Accessed 6 Nov 2017
13. Benoit, A., Brenner, L., Fernandes, P., Plateau, B., Stewart, W.J.: The Peps software tool. In: Kemper, P., Sanders, W.H. (eds.) TOOLS 2003. LNCS, vol. 2794, pp. 98–115. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45232-4_7
14. Bini, D.A., et al.: Structured Markov chains solver: software tools. In: Proceedings of the Workshop on Tools for Solving Structured Markov Chains (SMCtools 2006). ACM (2006)
15. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley, Boston (1995)
16. Hermanns, H., Joubert, C.: A set of performance and dependability analysis components for CADP. In: Garavel, H., Hatcliff, J. (eds.) TACAS 2003. LNCS, vol. 2619, pp. 425–430. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36577-X_30
17. Katoen, J.P., et al.: The Ins and Outs of the probabilistic model checker MRMC. In: Proceedings of the 6th International Conference on the Quantitative Evaluation of Systems (QUEST 2009). IEEE Computer Society Press (2009)
18. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
19. Menth, M.: Description and analysis of Markov chains based on recursive stochastic equations and factor distributions. World J. Model. Simul. **7**(1), 3–15 (2011)
20. Spedicato, G.A., Kang, T.S., Yalamanchi, S.B., Yadav, D.: The markovchain Package: A Package for Easily Handling Discrete Markov Chains in R. https://cran.r-project.org/web/packages/markovchain/vignettes/an_introduction_to_markovchain_package.pdf
21. Stewart, W.J.: Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Princeton (1994)
22. Stewart, W.J.: Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling. Princeton University Press, Princeton (2009)
23. Timmer, M., Katoen, J.-P., van de Pol, J., Stoelinga, M.I.A.: Efficient modelling and generation of Markov automata. In: Koutny, M., Ulidowski, I. (eds.) CONCUR 2012. LNCS, vol. 7454, pp. 364–379. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32940-1_26

Model-Based System Design and Evaluation of Image Processing Architectures with SimTAny Framework

Anna Deitsch(✉) and Vitali Schneider(✉)

Department of Computer Science 7,
Friedrich-Alexander-University of Erlangen-Nuremberg,
Martensstr. 3, 91058 Erlangen, Germany
{anna.deitsch,vitali.schneider}@fau.de

Abstract. Becoming a ubiquitous part of a huge number of various applications, image processing algorithms and underling architectures have to meet many different requirements. Some have real-time performance constraints combined with demands on efficient implementation for limited or various hardware resources. This poses particular challenges for design, implementation, and evaluation of efficient image processing systems. In this paper, we present a model-based approach to address these issues using our framework *SimTAny*. Founded on the standard modeling language UML, we propose the *UML Image Proccessing Language (UIPL)* to facilitate expressing image processing application algorithms directly in UML, which is especially beneficial for rapid modeling. With the help of *SimTAny*, such design models can be simulated in order to investigate the performance of a modeled system, to determine optimal design solutions, and to validate the required properties. We extend *SimTAny* to enable the generation of efficient implementation code of image processing algorithms for different target architectures. The code generated is then directly integrated in the simulation environment to increase the accuracy of our performance evaluations.

Keywords: Model driven engineering · UML · SysML · MARTE
Image processing applications · High-level synthesis

1 Introduction

In recent years, image processing applications have become increasingly popular in various areas of research, industry and also in the private sector. Mobile devices, like for example unmanned aerial vehicles (UAVs) applied for tracking of moving vehicles on roads, are often equipped with applications which perform complex image processing algorithms on-board, dealing with a considerable amount of sensor data. The design of such systems mainly relies on the following challenging issues: first, how to deal with heterogeneity and ensure sufficient performance; second, how to abstract implementation details in order to

R. German et al. (Eds.): MMB 2018, LNCS 10740, pp. 338–342, 2018.
https://doi.org/10.1007/978-3-319-74947-1_29

manage their complexity; third, how to refine these abstract representation in order to produce efficient implementations.

Recently, the *SimTAny* framework was proposed to cope with the above issues. This framework aims at enabling the performance investigation and validation of hardware/software architectures by simulating the UML-based system specification models and executing test cases at early engineering stages [1]. A notable special feature of *SimTAny* and its underlying modeling approach is that they are solely based on common standards.

Due to the high heterogeneity of platform components and the complexity of algorithms, the practical applicability of the UML-based design flows is still a difficult question, particularly for the image processing domain. As a remedy, in this work, we aim to exploit domain specific knowledge directly in a UML conform modeling library in order to benefit from rapid system description through providing useful modeling elements often required in practice. We also suggest a design flow which considers different abstraction levels with refinement stages [2]. The suggested design flow is accompanied by our modeling library that allows for seamless application of common, UML-based modeling standards for system specification, while reducing the complexity of the modeling process.

Furthermore, we improve upon a recently introduced *SimTAny* framework with techniques which allow the designers to automatically generate code for both simulation and implementation from high-level system specifications of image processing algorithms. In order to enable the generation of efficient implementation code for different target architectures without cumbersome modeling of platform specific details, we apply the HIPA^{cc} framework [3]. Based on a domain-specific language (DSL) embedded into C++, this framework offers a source-to-source compiler which translates algorithms defined in this high-level DSL to a highly optimized target-specific implementations. With *SimTAny* it is now possible, to generate such DSL code for HIPA^{cc} from UML-based specification models. Additionally, techniques have been applied in *SimTAny* which allow integration of the generated implementations into the simulation environment for refined simulation of the overall system.

The remainder of this paper is structured as follows: Sect. 2 describes our contributions to facilitate the suggested design process. Section 3 explains the implemented evaluation process.

2 Modeling of Image Processing Systems

As mentioned in Sect. 1, creating formal specifications of image processing algorithms takes a huge amount of effort. Hence, we enhanced the framework with a concept of viewpoints and view-specific wizard support. As already described in [2], we defined the general concerns to be covered by each viewpoint. For the corresponding modeling activities, appropriate diagram types and stereotypes from UML profiles have been associated. In the high-level specification phase the requirements and functional system model have to be captured using SysML modeling language [4]. The system model is progressively refined capturing functional (behavior and structure) and non-functional (timing, power

consumption, and etc.) properties with the help of MARTE [5] stereotypes. As soon as a sufficient level of details is achieved, the architecture modeling level phase can start. During this phase, we focus on detailed considerations of the system architecture and on the preparation of the code generation phase. Based on the concepts provided by SysML and MARTE, we refine the high-level system components in terms of hardware/software application components. Thereby, the provided extensions are primarily adapted for the modeling of image processing systems.

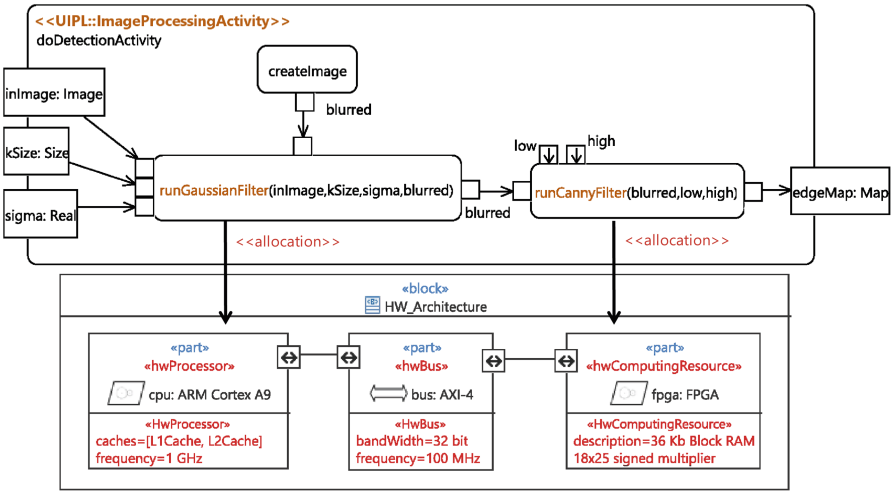


Fig. 1. Description of an image processing behavior using UML activity diagram and *UIPL* library

In general, any complex image processing algorithm can be described as an image processing pipeline, which consists of several filter operators concatenated in a pipeline structure. Depending on the memory access scheme, we distinguish between a point, local, and global filter operators [6]. In this work, we propose the *UIPL* to enable representing image processing applications directly in UML and refining them with image processing offerings captured by dedicated UML profiles, like the one depicted in Fig. 1. The *UIPL* is realized as an extension to UML that comprises a modeling library for expressing image processing pipeline and a set of profiles for writing then with image processing offerings. The modeling library provides predefined elements for describing artifacts related to the image processing domain, like buffering, collections structure as well as generic data types. The library also contains components which describe the structure and behavior of certain filter operators, like threshold or sort operations, applied in a wide range of local filter operators. The intended purpose of *UIPL* is to express image processing pipelines by common UML modeling concepts and to enable the writing of such models with concrete image processing offerings. This writing is

achieved by applying a dedicated *UML* profile to a model expressed with the help of the *UIPL Library*. The overall set of stereotypes encompass image processing filter operator's offerings such as computation cycles, number of instructions, and memory usage. They are important for performance analysis issues and the possible design choices provided in the later steps of the design process. In particular, we aim at providing an efficient and rapid modeling of image processing algorithms, exploiting the benefits of wizard-based user guidance [2]. After the specification of the image processing pipeline is obtained, the next step involves the evaluation of the application description on the target hardware platform.

3 Model-Based System Evaluation

In order to enable the evaluation of image processing algorithms for different target hardware platforms at the modeling level, we combine *SimTAny* with the HIPA^{cc} framework. The framework consists of a DSL that is embedded into C++ and a source-to-source compiler. Exploiting the compiler, image filter descriptions written in DSL code can be translated into multiple target languages such as CUDA, OpenCL or GPUs. In our work, we generate such DSL code for HIPA^{cc} from UML-based specification of image processing filter operators. Moreover, we apply in *SimTAny* the *Pointer to IMPLementation (PIMPL)* programming technique [7] to integrate generated implementations from HIPA^{cc} into the simulation engine OMNeT++ [8]. The proposed design flow is depicted in Fig. 2. The output data collected during the simulation can be directly analyzed in our framework, in the first instance, without prior external analysis tools being required. Therefore, *SimTAny* provides a dedicated perspective for analysis where the simulation results can be imported and visualized [1].

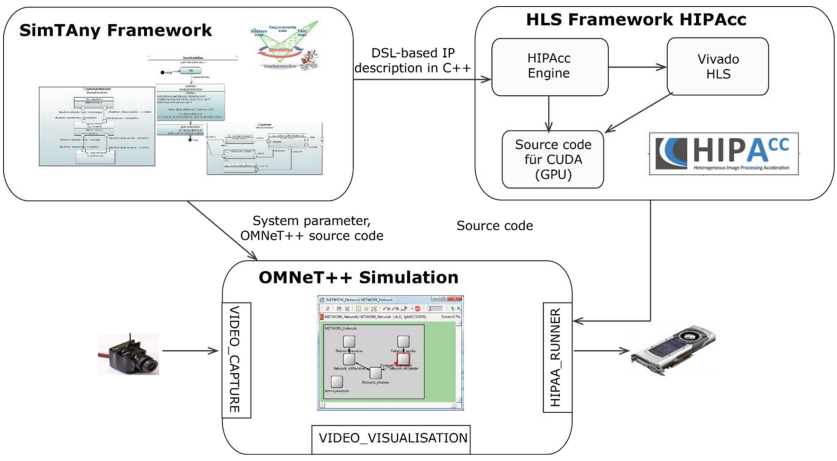


Fig. 2. Design flow of the proposed combination of *SimTAny* and HIPA^{cc}.

4 Conclusions

In this work, we demonstrated how the *SimTAny* framework can be extended for rapid and efficient design of image processing systems. Therefore, we enhanced the framework with a concept of viewpoints and view-specific wizard modeling. Thereby, we introduced the *UIPL* modeling language which permits easy capturing of image processing aspects at the model level. Additionally, we extend *SimTAny* to enable the generation of efficient implementation code of image processing algorithms for different target architectures.

References

1. Schneider, V., Deitsch, A., Dulz, W., German, R.: Combined simulation and testing based on standard UML models. In: Fiondella, L., Puliafito, A. (eds.) Principles of Performance and Reliability Modeling and Evaluation. SSRE, pp. 499–523. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30599-8_19
2. Deitsch, A., Schneider, V., Kane, J., Dulz, W., German, R.: Towards an efficient high-level modeling of heterogeneous image processing systems. In: Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative (DEVS 2016), Pasadena, CA, USA, April 2016
3. Membarth, R., Reiche, O., Hannig, F., Teich, J., Korner, M., Eckert, W.: HIPA^{cc}: a domain-specific language and compiler for image processing. IEEE Trans. Parallel Distrib. Syst. **27**(1), 210–224 (2016)
4. Object Management Group (OMG), SysML Systems Modeling Language (2012). http://omg.org/spec/SysML
5. Object Management Group (OMG), UML Profile for MARTE Modeling and Analysis of Real-Time and Embedded Systems (2011). http://omg.org/spec/MARTE
6. Yupatova, A., Schneider, V., Dulz, W., German, R.: Test-driven agile simulation for design of image processing system. In: Proceedings of 16th International Conference on Advances in System Testing and Validation Life Cycle (VALID 2014). IARIA, October 2014
7. Sutter, H.: Exceptional C++: 47 Engineering Puzzles, Programming Problems, and Solutions. Addison-Wesley Longman Publishing Co., Inc., Boston (2000)
8. OMNeT++ Network Simulation Framework. http://omnetpp.org
9. Reiche, O., Özkan, M., Membarth, R., Teich, J., Hannig, F.: Generating FPGA-based image processing accelerators with Hipacc. In: Proceedings of the International Conference on Computer Aided Design (ICCAD), pp. 1012–1019. IEEE

Author Index